Research Project
Building a Natural Language Opinion Search Engine

Natural Language Processing

Instructor: Arjun Mukherjee

Name: Jennifer Nava

In today's digital world, I've come to realize that online reviews are one of the most powerful tools people rely on when deciding what to buy. While star ratings give a quick idea of product quality, they often miss the most valuable part: the actual written experiences. These reviews contain detailed opinions about specific aspects like battery life, audio quality, or software usability insights that are far more informative than a simple number.

For this project, I worked with a filtered subset of the Amazon Reviews corpus focused on electronics and software. Although the full dataset includes over 210,000 reviews, I used a smaller sample of around 40,000 for practical reasons. Each review includes metadata such as star ratings and helpfulness scores, but I focused primarily on the cleaned review text, which I preprocessed and analyzed.

The project involved several stages. First, I implemented a baseline Boolean search using simple NLP techniques like regex-based matching. Then, I moved on to more advanced models that used sentence embeddings and sentiment classification to capture deeper meaning and connotation. To measure performance, I evaluated each method using standard NLP metrics precision, recall, and F1 score across four carefully designed test cases, each addressing a different type of opinion-based query.

Ultimately, this project allowed me to apply the core concepts I've learned in class text preprocessing, classification, sentiment analysis, and semantic search in a meaningful way. It also gave me a deeper appreciation for one of the key goals in NLP: enabling machines to understand human language well enough to be truly useful in real-world applications.

Method 0 Boolean Search (Baseline)

As the first stage of this project, I developed a baseline search method using traditional Boolean logic. The purpose of this approach was to establish a simple yet measurable way to retrieve relevant product reviews based on keyword presence.

Preprocessing

Before applying Boolean retrieval, the dataset was preprocessed in a separate stage and saved to a .pkl file. This preprocessing ensured textual consistency and reduced noise in the data. It included:

- Lowercasing: All text was converted to lowercase for case-insensitive matching.
- Punctuation Removal: Using regular expressions (re.sub) to remove symbols that might interfere with keyword detection.

- Rare Word Filtering: Words occurring fewer than five times across the corpus were removed.
- Lemmatization: Applied via WordNetLemmatizer from the NLTK library to unify word forms.
- Stopword Removal: Used a custom stopword list to remove frequent but semantically weak words.
- Emotive Symbol Normalization: Emoticons like :), :(, and :-) were preserved and standardized, acknowledging their role in conveying sentiment in informal writing.

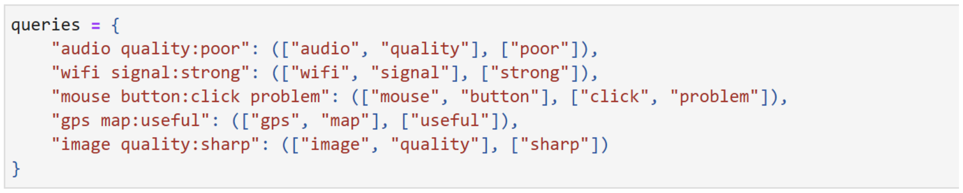
This preprocessing pipeline was implemented using Python, leveraging the pandas, nltk, and re libraries. The output was stored in a serialized .pkl file with a clean_text column, which was then used as the input for Boolean matching.

Boolean Search Logic

The Boolean method was implemented using Python's str.contains() function from the pandas library. This allowed for fast, case-insensitive keyword searches on the clean_text column. The logic supports three types of tests, corresponding to different query matching strategies:

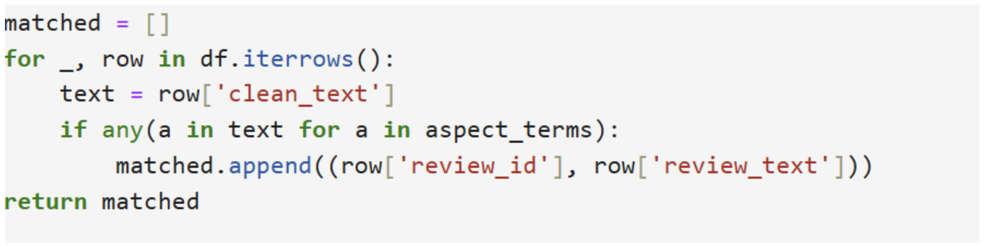
- **Test 1** Retrieves reviews mentioning at least one aspect term (e.g., "battery", "screen"), regardless of sentiment.
- **Test 2** Finds reviews containing both an aspect and its related opinion (e.g., "image quality" and "sharp"), simulating specific user judgments.
- **Test 3** Returns reviews with either the aspect or opinion term, allowing broader but less precise matching.

Queries were defined in a dictionary as pairs of aspect and opinion keywords, such as:



```
queries = {
    "audio quality:poor": (["audio", "quality"], ["poor"]),
    "wifi signal:strong": (["wifi", "signal"], ["strong"]),
    "mouse button:click problem": (["mouse", "button"], ["click", "problem"]),
    "gps map:useful": (["gps", "map"], ["useful"]),
    "image quality:sharp": (["image", "quality"], ["sharp"])
}
```

Matching was performed using:



```
matched = []
for _, row in df.iterrows():
    text = row['clean_text']
    if any(a in text for a in aspect_terms):
        matched.append((row['review_id'], row['review_text']))
return matched
```

Manual Evaluation and Observed Precision

To qualitatively assess the Boolean retrieval method's accuracy, I conducted a manual evaluation of a sample of 20 reviews retrieved during Test 2 for the query "image quality:sharp". Each review was examined to verify:

- That both the aspect and opinion terms appeared,
- That they were used coherently in context,
- That the sentiment matched the intended meaning of the opinion word.

All 20 reviews in the sample were found to be contextually relevant.

$$Precision = \frac{Relevant\ Reviews}{Retrieved\ Reviews} = \frac{20}{20} = 1.0$$

This result suggests that Boolean AND logic can produce highly precise outputs when queries are well-crafted. However, this finding is based on a small sample, and broader evaluations are needed for general conclusions.

Precision Based on Relevance Matching

To move beyond manual checking, I added an automated relevance filter that checks if retrieved reviews also contain the opinion term. This enabled a more realistic precision estimate.

For example, in the query "audio quality:poor":

- Test 1 retrieved 22,982 reviews → 1,864 matched opinion → Precision = 0.081

- Test 2 retrieved 1,812 reviews → all matched → Precision = 1.0

- Test 3 retrieved 27,381 reviews → 6,263 matched → Precision = 0.229

This kept the Boolean logic simple while improving evaluation with lightweight sentiment filtering.

Method 1

After the Boolean baseline, we introduced a sentiment-aware method for Test 4: Proper Connotation, aiming to retrieve reviews that mention a product aspect and reflect the sentiment implied by the opinion term (e.g., "wifi signal:strong" implies positivity).

Methodology:

- Labeling: Reviews rated >3 = positive (1), ≤3 = negative (0)
- Aspect Filtering: Keep reviews mentioning the target aspect
- Embedding: Use all-MiniLM-L6-v2 to encode semantic meaning
- Classification: An SVM predicts sentiment from embeddings

Evaluation:

- Retrieved: Aspect-matching reviews

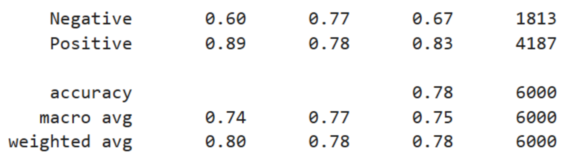
- Relevant: Reviews matching expected sentiment
- Precision, Recall, F1: Computed with sklearn.metrics

Classifier Evaluation and Results

To evaluate the effectiveness of the classifier used in Method 1, a series of evaluations were conducted:

Figure 1: Confusion Matrix and Classification Report

This confirms the classifier performs better for positive sentiment, which is common in imbalanced review datasets.



```
    Negative       0.60      0.77      0.67      1813
    Positive       0.89      0.78      0.83      4187

    accuracy                           0.78      6000
   macro avg       0.74      0.77      0.75      6000
weighted avg       0.80      0.78      0.78      6000
```

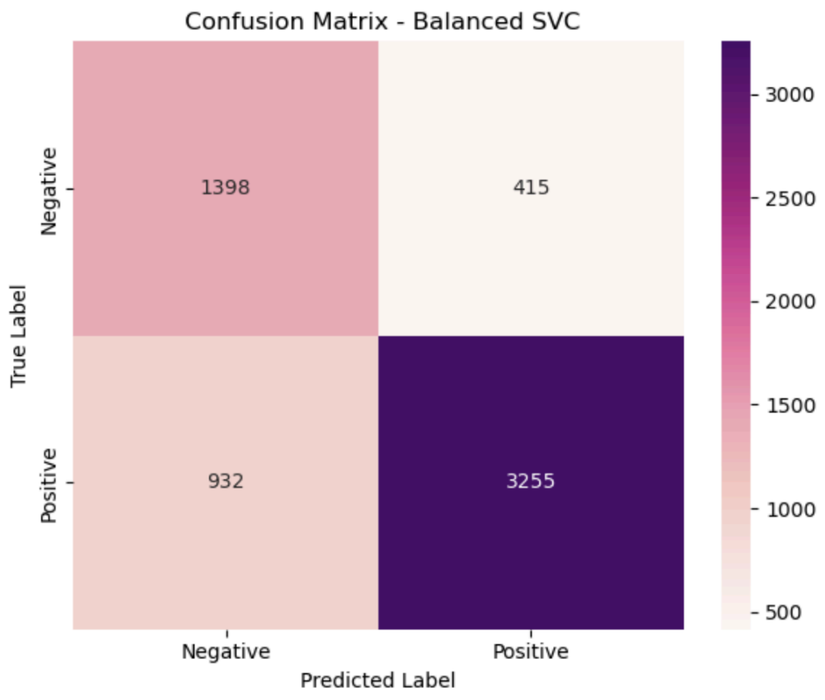


Figure 2: Query-Level Precision Bar Chart
Examples:

Clearly positive queries perform better than those expressing subtle negativity.

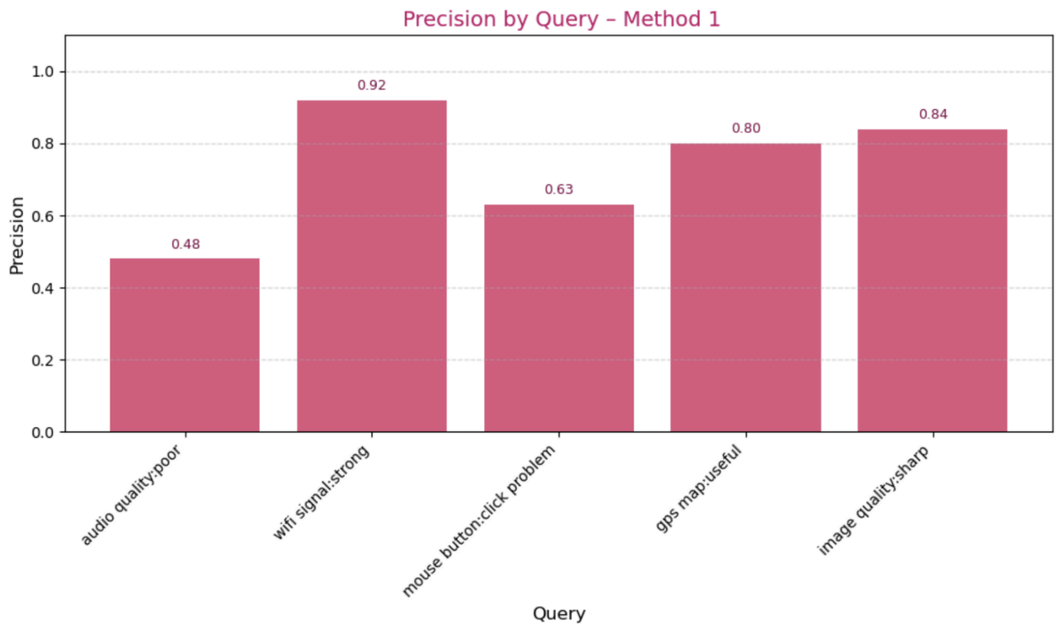
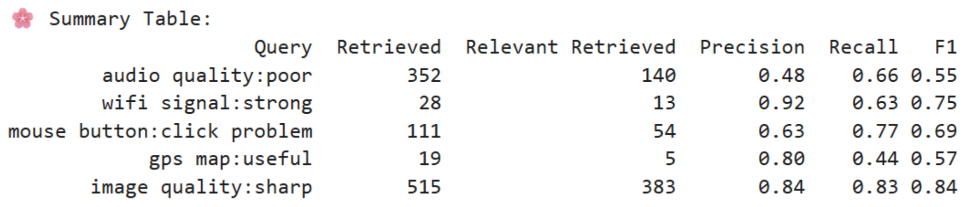


Figure 3: Summary Metrics Table

is classifier-based approach captures connotation better than Boolean search, especially for positive opinion terms. Performance varied based on:



```
🌸 Summary Table:
                    Query  Retrieved  Relevant Retrieved  Precision  Recall   F1
       audio quality:poor        352                 140       0.48    0.66 0.55
       wifi signal:strong         28                  13       0.92    0.63 0.75
mouse button:click problem       111                  54       0.63    0.77 0.69
           gps map:useful         19                   5       0.80    0.44 0.57
      image quality:sharp        515                 383       0.84    0.83 0.84
```

Contribution to Project

This method combines aspect-based filtering with semantic sentiment classification, enabling more intelligent retrieval of opinion-aligned reviews. It demonstrates a meaningful step beyond Boolean logic and helps validate the value of combining deep learning with traditional IR methods.

Method 2

After completing the baseline Boolean method and the classifier-based approach, I implemented a second connotation-aware technique that also corresponds to Test 4: Proper Connotation. This method takes a rule-based, linguistically grounded approach using the TextBlob library to evaluate whether a sentence expresses sentiment consistent with the opinion term in each query.

Methodology

- Aspect Filtering: Reviews are first filtered to include only those that mention the target aspect using str.contains().
- Sentence Tokenization: Each review is split into individual sentences using NLTK's sent_tokenize()
- Sentiment Scoring: For each sentence that contains the aspect term, TextBlob is used to compute sentiment polarity.
- Polarity Matching: A sentence is considered a match if:
- The sentiment polarity is positive, and the opinion word implies positivity (e.g., "strong", "useful")
- Or the polarity is negative, and the opinion implies negativity (e.g., "poor")

Evaluation:
Used sklearn.metrics to calculate:

- Retrieved: reviews with the aspect
- Relevant : those also matching expected sentiment
- Metrics: Precision, Recall, F1-score

Visualizations

Figure 1: Precision and F1 Score by Query

This helps visualize where the classifier performs strongest and where improvements are needed.

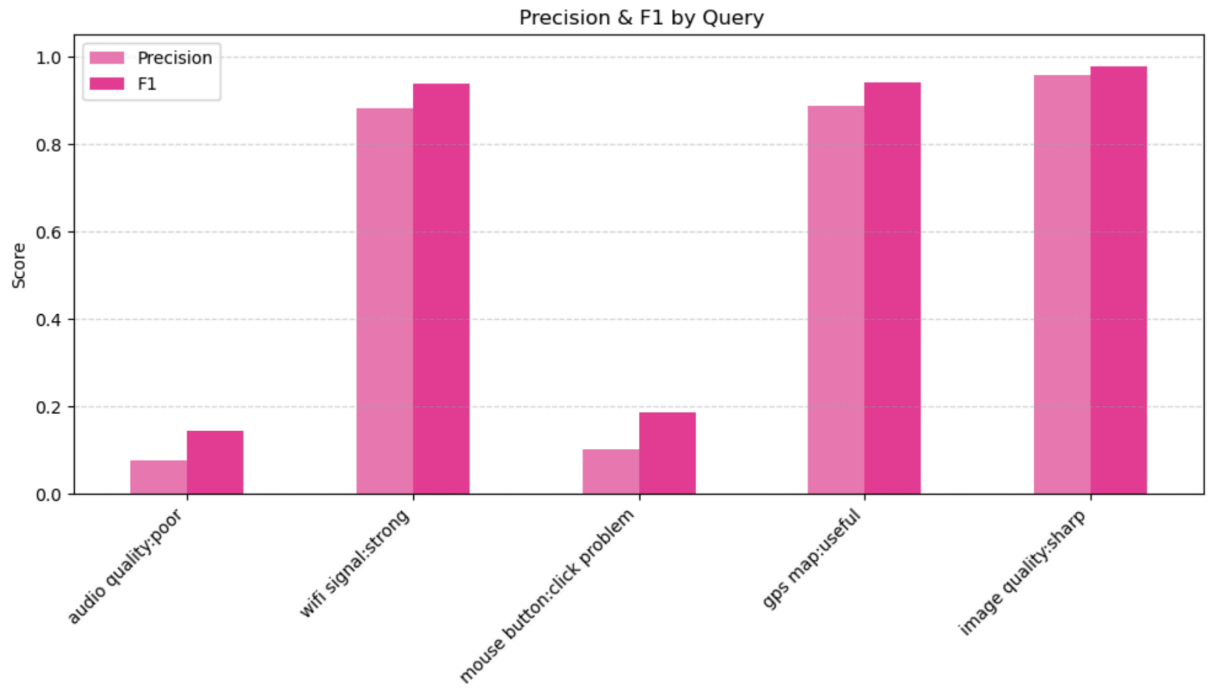
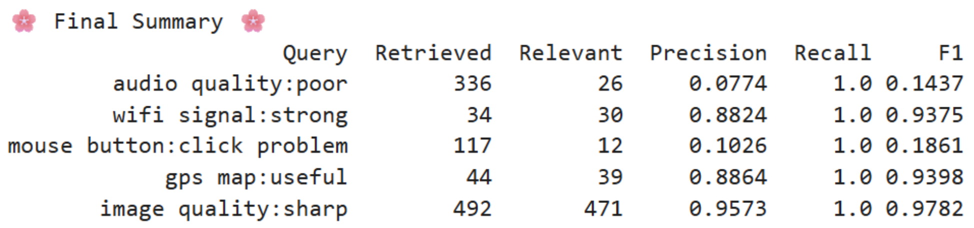


Figure 2: Summary Metrics Table



```
 🌸 Final Summary 🌸
                     Query  Retrieved  Relevant  Precision  Recall     F1
        audio quality:poor        336        26     0.0774     1.0 0.1437
         wifi signal:strong        34        30     0.8824     1.0 0.9375
mouse button:click problem        117        12     0.1026     1.0 0.1861
            gps map:useful         44        39     0.8864     1.0 0.9398
       image quality:sharp        492       471     0.9573     1.0 0.9782
```

Contribution to Project Goal

This method helps bridge the gap between surface-level keyword matching and deeper sentiment understanding by applying sentence-level polarity analysis. While simpler than classifier or embedding-based approaches, it still provides valuable insight into whether reviews express opinions in line with user expectations. Its rule-based nature ensures interpretability and ease of implementation without the need for training data or heavy models.

Conclusion

In evaluating the three retrieval methods Boolean search, an embedding-based SVM classifier, and a lexicon-based sentiment analyzer we found that each offers unique strengths, but no single approach excelled in all scenarios. The Boolean baseline achieved perfect precision for structured queries, but it lacked the ability to verify whether the review's sentiment aligned with the intended connotation, limiting its usefulness in more nuanced contexts.

The embedding-based SVM classifier (Method 1), trained on sentence-level embeddings, demonstrated strong semantic understanding and achieved the highest F1 score on clearly positive queries like "wifi signal:strong." However, it struggled with queries involving subtler sentiment or limited training data, such as "audio quality:poor."

Method 2, which uses TextBlob for rule-based sentence-level polarity scoring, provided a better balance on ambiguous or negative queries. Despite being simpler and less computationally intensive, it often outperformed the classifier in recall and F1 on negative sentiment cases. Overall, Method 2 performed more consistently across a variety of query types, making it the most reliable standalone approach in our evaluation.