Research Project

Building a Natural Language Opinion Search Engine

Natural Language Processing

Instructor: Arjun Mukherjee

1. Introduction:

In the modern era of technology, user reviews play a critical role in influencing consumer decisions about online shopping. This project focuses on developing a search engine that retrieves relevant product reviews based on specific aspects and opinions. By incorporating Natural Language Processing (NLP) techniques, the system ensures a higher relevance of retrieved documents compared to traditional search methods. In the process, we perform a practical task where you want to search for relevant and coherent opinions as well as training the machine in an efficient way where it can understand the language instead of just understanding the code queries.

The primary goal is to improve the precision as well as determine the pros and cons through retrieved results between advanced NLP searching methods. This project evaluates three approaches: a Boolean search which is our baseline ground truth, a Boolean search with augmented review ratings, and finally an embedding-based search. With the use of an inverted index which we have gone through this course, we will boost the process and make the runtime more efficient with less time consumption as well as help with better accuracy. Through this project, we will make use of the text-mining method and determine their performance. These methods are tested on the Amazon reviews dataset, which offers a real-world representation of diverse customer sentiments and product feedback. Preprocessing steps, including text cleaning, tokenization, stopword removal, lemmatization, and inverted index construction, were critical in preparing the dataset for these methods

2. Methodology:

This section outlines the approaches used in this project, from preprocessing the dataset to implementing and evaluating multiple search methods. Each method aims to improve the relevance and precision of the retrieved reviews.

2.1 Dataset and Problem Definition:

The dataset consists of Amazon product reviews, containing the following key attributes:

- Review_id: indicating the ID of each review which we will be looking at mostly on our output results
- Product_id, customer_id, review_title, review_written_date, customer_name, review_from_title, helpful_count, out_of_helpful_count, number_of_comments amazon_verified_purchase, amazon_vine_program_review, review_with_metadata: these data helps identify the review more but in fact I will not go in-depth for these because our work did not make much use of them.
- Customer_review_rating (0-5): numeric rating provided by users which we will be using for our second method.
- Review_text: the main resource we will be analyzing in our baseline which will then be contributed and developed with all of our methods. It is used to form and separate for the inverted-index build.

Problem Identification:

Traditional keyword-based search methods struggle to retrieve relevant results. For example, a query like wifi signal: strong, may fail to capture reviews discussing "great connectivity" or "excellent reception." This project addresses the following challenges:

1. Semantic Understanding: Capturing relationships between synonyms and related terms.
2. Contextual Relevance: Ensuring retrieved reviews match both the aspect (e.g., "wifi") and opinion (e.g., "strong").
3. Efficiency: Balancing computational cost with precision and recall.

2.2 Preprocessing:

Effective preprocessing was crucial for ensuring meaningful results from the implemented search methods. In this we will build the inverted_index through dictionary data structure which we will then use for all the methods. The steps included:

Text Cleaning: Removed punctuation, and special characters, and normalized text to lowercase. Replaced emojis (e.g., :-) and :-( ) with sentiment placeholders like positive_smiley and negative_smiley.

Tokenization: Split review text into individual tokens using NLTK's tokenizer.
Stopword Removal: Removed frequent but non-informative words (e.g., "the", "and", "is") using an expanded stopword list, including domain-specific stopwords.
Lemmatization: Reduced tokens to their base forms (e.g., "running" → "run") using WordNet Lemmatizer, ensuring consistency across similar word forms.
Inverted Index Construction: Created a posting list mapping each token to its occurrences across reviews, including Review IDs and Sand tar Ratings Cleaned review text

2.3 Search Methods

2.3.1 Baseline: Boolean Search

The Boolean search method used an inverted index to retrieve reviews that matched the exact tokens in the query. It is the foundation approach in information retrieval systems. It uses Boolean logic (AND, OR) to identify documents that contain specified query terms. In this project, operates on an inverted index, a data structure mapping each term to its occurrences across the dataset. We then parse the query and split it into individual tokens and finally retrieve posting lists for each token from the inverted index. OR retrieves reviews containing any query terms AND comes with reviews containing all query terms.

Theoretical Advantages:

- Fast and efficient for exact matches.
- Easy to implement with minimal computational resources.

Theoretical Limitations:

- Cannot capture semantic relationships (e.g., synonyms or paraphrased expressions).
- Results often include irrelevant matches due to a lack of context and analysis depth.

2.3.2 Boolean_Rating Search

Enhancement:

Extends Boolean search by incorporating review star ratings to align with the sentiment of opinion terms as an additional requirement condition:

- Positive Opinions (e.g., "great") → Star rating > 3.
- Negative Opinions (e.g., "poor") → Star rating ≤ 3

Use the same algorithm as we did for Boolean search to retrieve initial matches. After that we then filter reviews based on star ratings to match query sentiment

Theoretical Strength:

- Improved precision by filtering irrelevant results.
- Useful for sentiment-aware retrieval tasks.

Theoretical Weaknesses:

- Relies on coarse-grained ratings, which may not capture nuanced sentiments.

2.3.3 Embedding-Based Search

This method leverages word embeddings (e.g., GloVe) to compute semantic similarity between query terms and review content. It works still use the value from inverted-index list then to analyze based on its functions. This approach identifies synonyms and contextually similar terms then use the condition we have to filter out our results.

Algorithm:

1. Embedding Construction:
   - Compute mean embeddings for query aspects and opinions.
   - Compute mean embeddings for each review text.
2. Similarity Calculation:
   - Use cosine similarity to compare query embeddings with review embeddings.
3. Result Filtering:
   - Retain reviews with similarity scores above a specified threshold.
   - As we increase the threshold, the accuracy of the retrieved results gets better, however, we need to make sure that if we increase it by a lot, it can potentially lose some important information and results.

Theoretical Strengths

- Captures semantic relationships (e.g., "bad" ≈ "poor").
- Flexible and effective for paraphrased or contextually similar queries.

Theoretical Weaknesses

- Computationally expensive for large datasets.
- Results depend on embedding quality and coverage.

3. Results:

For baseline: We will run the code to retrieve all 3 methods (OR, AND , mix of OR AND) boolean for 5 queries: audio quality: poor; wifi signal: strong; mouse button:c lick problem; gps map: useful; image quality: sharp. We then will just focus on 1 of them as a pivot for each query which is the OR query, so that we can then compare them with the boolean_rating method and the embedding method we are also doing so that we will lower the chance of missing out on some of them in case the AND AND produce an empty result set since OR seems to be pushing out the higher amount of reviews_id between queries. For the Ret, I used the total results I have for my OR queries, however, for Rel I pick out randomly 20 outputs, which I will also used for Precision based on how many #Rel out of that 20.

Key Observations:

Baseline Boolean Search

- Strengths:
  - High recall: Captures all possible matches by exact keyword search.
  - Simple and computationally efficient.
- Weaknesses:
  - Very low precision: Retrieves many irrelevant reviews.
  - No semantic understanding, leading to keyword mismatches.

Boolean + Rating Search

- Strengths:
  - Balances precision and recall by filtering results based on star ratings.
  - Aligns with the sentiment indicated by the opinion terms.
- Weaknesses:
  - Relies on coarse-grained ratings, which may miss some relevant context.
  - Still unable to capture semantic relationships between words.

Embedding-Based Search

- Strengths:
  - High semantic relevance: Captures synonyms and related terms (e.g., "strong" = "powerful").
  - Flexibility in matching paraphrased or loosely related content.
- Weaknesses:
  - Computationally intensive for large datasets.
  - The result set size depends on the threshold, and lower thresholds can introduce noise.

We then have the precision trends:

The baseline seems to have low precision for most of the query which the majority of the results were irrelevant , boolean_rating will be between medium but the highest at 13/20, then come to embedding-based will have between medium and high with 100% precision at the image quality: sharp query with 0.8 threshold, but surprisingly gets to 2/20 at the wifi signal: strong but can potentially get higher as we increase the threshold value. The OR condition for our queries actually make our Ret comes out to be the highest in numbers, however the Rel and Prec actually gets a bit more tricky and not 100% accurate because we are manually picking up 20 randomly out of thousands of results, which then potentially miss out on the important information but it somehow reflects how "random" the results can be! However, in case if we change the threshold for the embedding method, as we increase it, the amount of retrieved reviews is get significantly smaller. As for query image quality: sharp, as I use threshold 0,5 , I receive 23577 reviews, but if I use 0.8, it comes down to 1 and surprisingly the precision is 1/1 here.
The report for all the numbers after I run my queries (the output .pkl files are also included in the submitted folfer).

Query	Baseline (Boolean)			Method 1 (Boolean rating)			Method 2 (embedding)		
	# Ret.	# Rel.	Prec.	# Ret.	# Rel.	Prec.	# Ret.	# Rel.	Prec.
audio quality:poor	25243	9	9/20	8.457	5	5/20	24.736	7	7/20
wifi signal:strong	6844	9	9/20	4877	9	9/20	6776	2	2/20
mouse button:click problem	33871	7	7/20	29	8	8/20	33768	13	13/20
gps map:useful	8010	12	12/20	5407	13	13/20	7752	11	11/20
image quality:sharp	23936	5	5/20	16979	4	4/20	1 (with threshold of 0.8)	1	1/1

Discussion:

Retrieving relevant reviews from unstructured text is a challenging problem due to the flexibility of human language. Synonyms, paraphrasing, and contextual shifts make it hard to get exact matches, making semantic understanding a crucial component of effective retrieval systems.

The Baseline Boolean method is computationally efficient but fails to capture semantic relationships, leading to low precision. Boolean + Rating Search addresses this by aligning results with another rating condition, but this condition also limits its flexibility. Embedding-Based Search is theoretically the most precise, leveraging cosine similarity to capture the results better.

Conclusion:

Based on theoretical and practical performance:

1. Embedding-Based Search is the best method for precision-critical tasks due to its semantic capabilities.
2. Boolean + Rating Search is a strong alternative for sentiment-aware retrieval with moderate computational demands.
3. Baseline Boolean Search is suitable for exploratory tasks where high recall is prioritized over precision.