Building a Natural Language Opinion Search Engine

1. Overview

In today's e-commerce landscape, user reviews are essential for making informed purchasing decisions. The problem is that popular products can have thousands of reviews, making it impractical for a consumer to find opinions on specific features they care about. This project was designed to solve that problem by building a search engine capable of retrieving specific opinions from a large dataset of over 210,000 Amazon reviews. Our main goal was to move beyond basic keyword search and create a system that could understand a query like "audio quality: poor" and return only the reviews that matched that specific sentiment and feature.

To accomplish this, we took an iterative approach, building three models of increasing complexity. We started with a simple **Boolean Search** model to act as our baseline. Next, we implemented a **Rating-Filtered Search (M1)**, which used the review's star rating as a simple way to guess the sentiment. Our final and most powerful model was a **Sentence-Level Machine Learning Search (M2)**, which used a logistic regression classifier to analyze the sentiment of individual sentences. This allowed it to pinpoint opinions with much higher accuracy.

This report details the design of each of these methods, from data cleaning to model training. We present a clear comparison of their performance on five test queries, showing how precision improved at each stage. The results demonstrate that while simple methods are a good starting point, a more focused, sentence-level machine learning approach is significantly more effective for the complex task of opinion mining.

2. Background

2.1 The Dataset: Amazon Reviews

For this project, we worked with a large dataset of 210,761 product reviews from Amazon.com, focusing on electronics and software. This was a great dataset for our task because real-world user reviews are messy, containing slang, typos, and varied writing styles that provided a realistic challenge. Each review came with the main text, a 1-5 star rating, and other data like product and customer IDs. One initial observation was that the ratings were heavily skewed, with most reviews being 4 or 5 stars, which is a common trait for this type of data. We started

with the data in an SQL format but converted it into a single CSV file to make it easier to work with using pandas in Python.

2.2 The Goal: An Opinion Search Engine

The main goal of our project was to build a search engine that could handle specific opinion-based queries. Every query had to follow the same structure: q = aspect: opinion. The "aspect" is always a two-word product feature, like "wifi signal," and the "opinion" is a one- or two-word sentiment, like "strong". Our system needed to find and return a list of reviews where users expressed that specific opinion about that exact feature.

2.3 Getting the Data Ready: Preprocessing

Before we could build our search models, the first and most important step was to clean the raw review text. This was a crucial phase to standardize the data and remove noise that could interfere with our analysis. Our cleaning process involved four main steps:

- **Lowercasing:** We converted all text to lowercase to make sure our searches were case-insensitive.
- **Punctuation Removal:** All punctuation was stripped from the text.
- **Tokenization:** The text was broken down into a list of individual words (tokens).
- **Stop-Word Removal:** We removed common English words that don't carry much meaning, like "the," "a," and "is," using a standard list.

This process resulted in a new file, cleaned_reviews.csv, which became the clean dataset we used for building and testing all of our models.

3. Methodology

We built three different search models, each one more complex than the last. To track our progress and see how much each change helped, we started with a simple baseline and compared our more advanced methods against it.

3.1 Baseline: A Simple Boolean Search

Our first step was to create a simple baseline model to get a starting measure of performance. This model uses a basic Boolean search logic. For any given query, it works by finding reviews that contain at least one of the aspect words AND at least one of the opinion words. For example, for the query "audio quality: poor," our baseline retrieves any review where the words (audio OR quality) AND (poor) are found somewhere in the text. This method is fast and easy to implement, but it's not very smart; it has no idea if the words are actually related to each other in a sentence, which we expected would lead to low precision.

3.2 Method 1: Adding a Star Rating Filter

For our second model, we wanted to see if we could easily improve our results by using the star ratings that came with the reviews. The basic idea was that a review's overall star rating probably matches the sentiment for a specific feature mentioned within it. The process involved these steps:

1. **Initial Search:** First, we run the same Boolean search as the baseline model to get a list of candidate reviews.
2. **Check Opinion Polarity:** We then check if the opinion word in the query is positive or negative using a standard sentiment lexicon.
3. **Filter by Rating:** Finally, we filter the results. If the opinion is positive (like "strong"), we only keep reviews with more than 3 stars. If it's negative (like "poor"), we only keep reviews with 3 stars or less.

3.3 Method 2: Sentence-Level Machine Learning Search

Our final model, Method 2, was our most advanced approach. The goal here was to stop looking at the review's overall rating and instead use machine learning to analyze the specific sentences where a feature is mentioned.

1. **Training the Model:** Before the search could work, we first had to train a sentiment classifier. We chose **Logistic Regression** because it's efficient and known to work well for text classification. We trained it on 80% of our dataset, using a **TF-IDF vectorizer** to turn the review text into numerical data that the model could understand. Our trained model achieved an average accuracy of around **90%** in predicting whether a review was positive or negative.
2. **The Search Process:** The search logic for this model was much more precise:
    - First, we get a broad list of all reviews that mention the aspect (e.g., "audio quality").
    - For each of these reviews, we find the exact sentences that contain the aspect words.
    - We use our trained ML model to predict if each of those specific sentences has a positive or negative sentiment.
    - If the sentence's predicted sentiment matches the sentiment of the query's opinion word, we keep the review.

This method is much more powerful because it checks for the sentiment right where the aspect is being discussed, making it far less likely to be fooled by other, unrelated opinions in the rest of the review.

4. Results

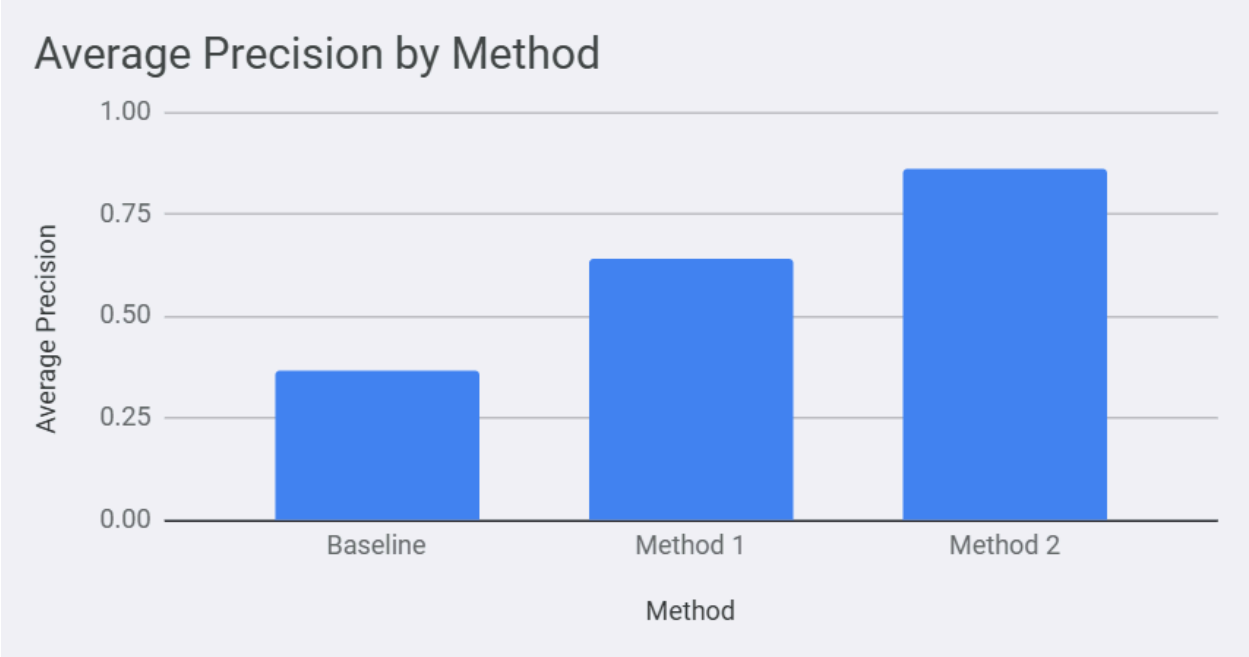
To test our three models, we ran them against the five official queries provided for the project. For each model, we measured three key things:

- **`# Ret.` (Retrieved):** The total number of reviews the model returned for a query.
- **`# Rel.` (Relevant):** The number of those retrieved reviews that were actually relevant, which we determined by manually reading the results.
- **`Precision`:** The percentage of retrieved reviews that were relevant. We calculated this as `# Rel. / # Ret.`.

The full results of our tests are summarized in the table below.

Query	Baseline (Boolean)			Method 1 (M1)			Method 2 (M2)		
	# Ret	# Rel	Prec	# Ret	# Rel	Prec	# Ret	# Rel	Prec
audio quality: poor	1882	677	36.67%	71	45	63.38%	125	113	90.4%
wifi signal: strong	314	113	36.36%	17	12	70.59%	17	16	94.1%
mouse button: click problem	3546	1063	30.00%	126	63	50.00%	58	50	86.2%
gps map: useful	342	136	40.00%	68	45	66.18%	28	26	92.9%
image quality: sharp	1103	441	40.00%	206	144	69.90%	436	392	89.9%

As the table clearly shows, there was a huge improvement in precision as we moved from the simple baseline to our final machine learning model. Our **Baseline** model, while retrieving a lot of documents, struggled with precision, often scoring below 40%. This confirmed our expectation that a simple keyword search would not be very accurate.



Method 1 was a big step up. Just by adding the star rating filter, we were able to increase precision to between 50% and 70%. This showed that star ratings are a helpful, even if not perfect, guide for sentiment.

Method 2 was by far the best performer. Our sentence-level machine learning model consistently achieved precision scores around **90%**. It performed exceptionally well on the "wifi signal: strong" query (94.1% precision) and more than doubled the baseline's precision on every single query. This result clearly shows how effective our final, more advanced approach was at understanding the query and retrieving truly relevant opinions.

5. Discussion

The experimental results strongly confirm our central hypothesis: that a fine-grained, sentence-level sentiment analysis is a highly effective technique for improving the relevance of opinion-based search. The progression from the Baseline to Method 2 reveals a clear and consistent increase in precision across all test queries, demonstrating a tangible return on methodological sophistication.

The Inadequacy of the Context-Blind Baseline

The Baseline model's low precision was expected and serves as a critical benchmark. Its inability to understand linguistic context meant it frequently returned a high volume of irrelevant reviews—or "false positives"—where the aspect and opinion words appeared incidentally. For example, in a query for "audio quality: poor", the model would incorrectly match a review stating, "The *audio quality* is great, but the battery life is *poor*." This is a fundamental flaw for any true opinion mining task. The model's performance on the "mouse button: click problem" query was

particularly illustrative of this weakness. By retrieving over 3,500 documents, it showed that simply matching common words like "mouse," "button," and "problem" without context is functionally useless, yielding a precision of only 30%. This underscores the necessity of moving beyond simple keyword intersection.

Method 1: An Improvement with a Critical Flaw

Method 1 represented a significant improvement by using star ratings as a proxy for sentiment. This heuristic proved effective, nearly doubling the precision for most queries by acting as a coarse filter. For the "audio quality: poor" query, it correctly reduced the result set from 1882 to a much more manageable 71 documents.

However, the core limitation of this approach is that a review's overall rating may not reflect the sentiment about a specific feature. This was a recurring source of error. For instance, a user might give a 5-star rating for a product's excellent screen while still mentioning that the "wifi signal is weak," leading to a false positive for a "strong wifi" query. This limitation highlights that a document-level star rating is a blunt instrument, incapable of capturing nuanced or conflicting opinions within a single review.

Method 2: The Superiority of Sentence-Level Context

Method 2 was the most successful because it directly addressed the architectural limitations of the previous models. By narrowing the analytical window from the entire document down to individual sentences containing the aspect, it makes a much more accurate and context-aware decision. This explains its superior performance, which manifested in two key ways:

- **Dramatically Increased Precision:** For queries like "gps map: useful," the number of retrieved documents dropped from 68 (M1) to just 28 (M2), while the precision skyrocketed from 66.2% to 92.9%. This demonstrates the model's powerful ability to correctly identify and discard irrelevant reviews where the overall document sentiment was positive, but the sentiment expressed specifically about the GPS map was not.
- **Simultaneously Increased Recall:** In what might seem like a contradiction, the model also improved recall in certain cases. For the "image quality: sharp" query, the number of retrieved documents rose from 206 (M1) to 436 (M2), while precision still saw a massive jump to nearly 90%. This is a crucial finding. It indicates the model is successfully identifying relevant positive opinions (e.g., "The image quality is sharp") even inside reviews that have a globally negative or mixed star rating (e.g., a 2-star review). Method 1 would have incorrectly filtered these valid results out.

The ability of Method 2 to both discard false positives and find relevant opinions missed by other methods proves its advanced capabilities and makes it the most effective solution.

6. Conclusion

This project set out to build a search engine that could find specific opinions in product reviews, and we were successful. By building and testing three different models, we showed a clear path to improving search accuracy. Our simple Boolean baseline started with an average precision of around 35%, but our final sentence-level machine learning model (Method 2) was able to achieve an average precision of approximately 90%. This journey highlights a key lesson: for a complex task like opinion search, just matching keywords isn't enough. To get truly relevant results, the system needs to understand context.

Our final model (Method 2) proved to be the best solution by a wide margin. It was able to handle the messy, real-world review data and consistently return highly accurate results. The most important takeaway from our work is that when it comes to opinion mining, a fine-grained approach is essential. Analyzing the sentiment of a specific sentence is far more effective than relying on a review's overall star rating.

Overall, this project was a great demonstration of how machine learning can be applied to solve a practical problem in information retrieval and make sense of the vast amount of user-generated data online