Implementation of Boolean Search and Advanced Opinion Mining Methods for Amazon Product Reviews

Kishan Yerneni, 2209717

A Machine Learning Approach to Sentiment Analysis for Enhancing Boolean Search in Identifying Relevant Amazon Reviews Based on Aspects and Opinions

Abstract

This project aims to develop a robust natural language opinion search engine leveraging advanced natural language processing (NLP) techniques. The system processes a real-world dataset of Amazon reviews to enable effective retrieval of opinions related to specific aspects of products. Two main methods were implemented. The first employs a Support Vector Classifier (SVC) to classify reviews based on the sentiment orientation of an opinion regarding given aspects. The second expands query terms using bigram collocations and retrieves sentiment-matched reviews using embeddings generated by a pre-trained BERT model. Both methods emphasize relevance and coherence, with the latter incorporating semantic similarity measures to refine results. Evaluative metrics such as precision, relevance, and retrieval count are utilized to compare the methods against a Boolean baseline. Our findings demonstrate the utility of combining statistical methods and transformer-based embeddings to improve opinion-based query relevance.

Introduction

Throughout this study, we have explored a variety of techniques aimed at efficiently processing and understanding human language. This project leverages some of these methodologies to develop an optimized Boolean search engine tailored for opinion-based retrieval from textual data. As a baseline, the system implements three foundational Boolean query methods: **(1)** `aspect1 OR aspect2 OR opinion`, **(2)** `aspect1 AND aspect2 AND opinion`, and **(3)** `(aspect1 OR aspect2) AND opinion`. Building upon this foundation, we propose and implement two advanced methods that extend the functionality of the baseline by incorporating machine learning techniques. These methods are designed to enhance the precision and relevance of the retrieved results. The selection and design of these extended methods are guided by a machine learning-driven approach, aiming to address the limitations of traditional Boolean logic in capturing nuanced sentiment and semantic relationships within textual reviews. This paper discusses the proposed methods, their implementation, and their comparative performance against the baseline approaches.

Methodology

This section outlines the five Boolean search methods utilized in this study. The first three methods serve as baseline approaches, relying solely on traditional Boolean logic. The remaining two methods represent advanced extensions that incorporate unique concepts and leverage machine learning techniques. These include a custom-built model and a pre-trained model, designed to enhance the search engine's precision and relevance. The baseline methods are discussed briefly, followed by a detailed explanation of the supplementary approaches.

Baseline Approaches

The baseline approaches utilize traditional Boolean logic for opinion-based retrieval. Three foundational query structures form the basis of these methods:

Method 1: Aspect1 OR Aspect2 OR Opinion. This query retrieves reviews containing any of the specified aspects or opinions, ensuring high recall but potentially lower precision.

Method 2: Aspect1 AND Aspect2 AND Opinion. This query retrieves reviews that contain all specified aspects and opinions, aiming for high precision at the cost of recall.

Method 3: Aspect1 OR Aspect2 AND Opinion. This query balances precision and recall by combining disjunctions and conjunctions of aspects and opinions.

Advanced Methods

The advanced methods integrate machine learning techniques to enhance the baseline approaches. These methods aim to improve relevance by leveraging semantic and sentiment-based analysis.

Method 4: Support Vector Classifier (SVC)-Based Approach

Overview. This method employs an SVC model to classify reviews based on sentiment orientation, enhancing the relevance of retrieved opinions. The main idea behind this method is to basically ignore the opinion term and focus on just its sentiment. We perform an AND operation on the aspect terms and filter them out based on the opinion sentiment.

This approach is heavily dependent on a lexicon containing around 6800 positive and negative words located here.

A. Training the model.

Label Creation. The first step in training the model is the creation of labels (positive or negative). To create this label, I used both the customer rating and a scale of the positive and negative words in the review text (lets call the sentiment score) using the above mentioned lexicon. I used Min-Max scaling to scale the number of positive and negative words in the text to be between zero and five. I then took the average of the customer review and the scaled sentiment score and subtracted five. Thus, if the new value (sentiment score) is positive, we assign it a label of 1 else 0.

Text Pre-processing. I wanted the model to receive a pre-processed piece of text and categorize it into positive or negative sentiment. I had two main choices on the method I wanted to use during the preprocessing: **(1)** `Using a Tfidf vectorizer`, **(2)** `Using a Word2Vec Model`.
I created two models where all the steps of creation are the same other than the type of pre-processing method I used and these are the results:

Cross-Validation for Tfidf Vectorizer Model:
Best Parameters:
'max_iter': 3000,
'loss': 'squared_hinge',
'C': 0.08858667904100823
Best Cross-Validation Accuracy: 0.8339876988643546
Accuracy on Test Set: 0.8343652883543283

Label	precision	recall	f1-score	support
0	0.79	0.62	0.69	12773
1	0.85	0.93	0.89	29380
accuracy			0.83	42153
macro avg	0.82	0.77	0.79	42153
weighted avg	0.83	0.83	0.83	42153

Cross-Validation for Word2Vec Model:
Best Parameters:
'max_iter': 2000,
'squared_hinge': 'squared_hinge',
'C': 1.623776739188721
Best Cross-Validation Accuracy: 0.7782074290367536
Accuracy on Test Set: 0.7786159941166703

Label	precision	recall	f1-score	support
0	0.71	0.46	0.56	12773
1	0.80	0.92	0.85	29380
accuracy			0.78	42153
macro avg	0.75	0.69	0.70	42153
weighted avg	0.77	0.78	0.76	42153

Why Choose TF-IDF Over Word2Vec?. TF-IDF outperforms Word2Vec for this sentiment classification task because the dataset consists of straightforward reviews where specific words like "great" or "bad" are crucial for determining sentiment. TF-IDF effectively captures the importance of such words by weighting them higher, making it well-suited for tasks focused on word-level features. On the other hand, Word2Vec, which captures semantic relationships between words, doesn't provide significant benefits in this case, especially when embeddings trained on the same dataset may lack the richness of pre-trained models like GloVe or FastText. Therefore, TF-IDF is the better choice for this task (1).

Model Visualizations. The following visualizations represent the output of the Tfidf model.

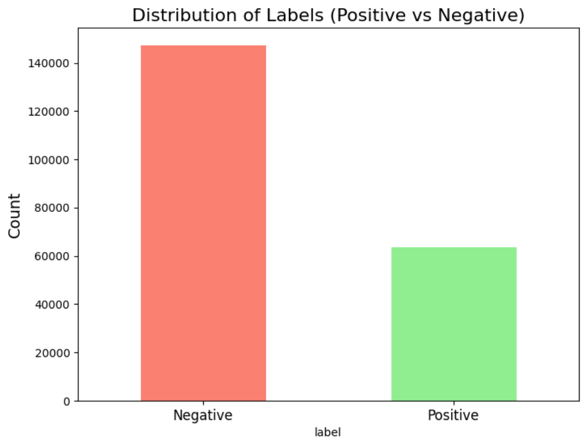

Fig. 1. Class Distribution.

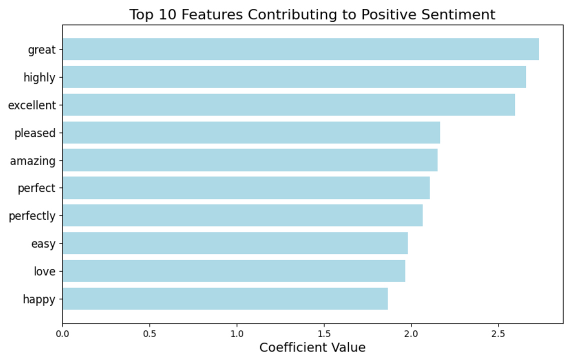

Fig. 2. TF-IDF Feature Importance.

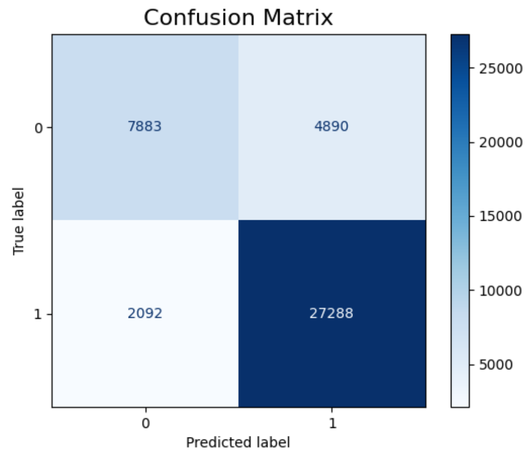

Fig. 3. Confusion Matrix.

The number of true positives is notably higher than other metrics, suggesting that the model is especially proficient at identifying positive sentiment reviews. However, this could also be influenced by the imbalance in the dataset, where positively labeled reviews are significantly fewer than negatively labeled ones (1). The code for training the model using Word2Vec and TF-IDF can be found in the code/SVC folder located in the root directory.

B. Running the method. This method begins with an AND operation, as outlined previously, to identify relevant reviews. The model is then applied to these reviews, further refining the results based on the model's predictions and the sentiment associated with the opinion term(s). While this approach ensures high precision, it may result in a decrease in recall. The following presents the number of reviews retrieved for a selection of example queries:

Query (aspect:opinion)	# of docs returned
audio quality:poor	251
wifi signal:strong	92
gps map:useful	415
image quality:sharp	1137

As observed in the table, the number of retrieved documents is relatively low. This can be attributed to the strict requirement that the reviews must contain both aspect1 and aspect2 exactly as specified. This constraint significantly limits the number of documents that match the query, thereby reducing the overall retrieval count.

C. Limitations. As previously discussed, the performance of this model is heavily reliant on the lexicon, which contains only 6,500 entries. Words not included in the lexicon may not be recognized during the pre-processing stage, potentially leading to inconsistencies in the analysis. Additionally, the presence of misspelled negative terms could similarly cause issues, as such words would not be correctly identified or processed. A significant limitation of this approach arises when the opinion term of a review is absent from the lexicon, as this can severely impact the accuracy of the sentiment analysis.

Method 5: Pre-Trained BERT Model for Semantic Matching

Overview. Method 5 improves review retrieval by expanding the search query using collocations (bigrams) related to predefined aspect and opinion terms, which captures more context and enhances query precision. Once relevant reviews are retrieved, semantic similarity scores are computed using sentence embeddings. Reviews are ranked based on their similarity to the expanded query, with only the top 25% retained. This method ensures high precision and maintains recall by performing an OR operation on the posting lists of all terms in the expanded query. Additionally, the `spaCy` package is employed for POS tagging to filter out relevant terms (nouns, verbs, adjectives, and adverbs) from the bigrams, further refining the query expansion.

D. Collecting Bigrams. Collocations are extracted from the pre-processed text using the `BigramCollocationFinder` and related functions from the `nltk.collocations` library (nltk collocations). The relevant code can be found in either the `codes/boolean_search.py` or the `codes/bigrams_with_BERT/visualization.py` files. To reduce computational time, the `numpy` module is used to calculate the 85th percentile, filtering out the top 15% of bigrams based on their score. Additionally, I ensure that only bigrams with relevant parts of speech (POS) are selected for query expansion, focusing on terms such as nouns, verbs, adjectives, and adverbs. Below is the top 15 bigrams with the following query "audio quality:poor":

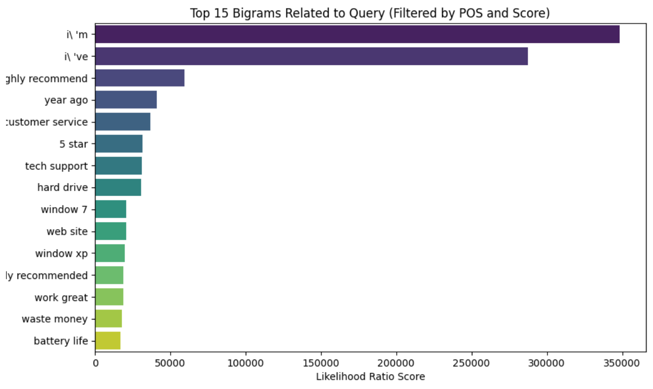

Fig. 4. Top Bigrams with Scores for audio quality:poor.

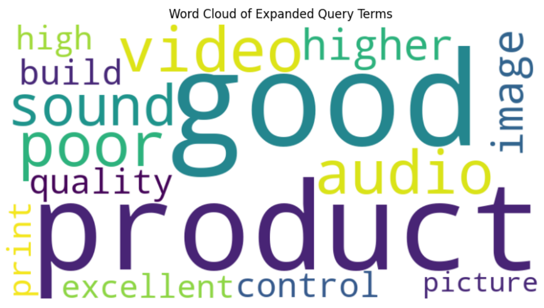

Fig. 5. word cloud for audio quality:poor expanded query.

As you can see, some of the words in the expanded query are not that relatable to the query, which is a potential downside of using n-grams.

E. Embedding the Reviews Using a Pre–trained BERT Model. In this approach, reviews are embedded using the pre-trained BERT model `SentenceTransformer('all-MiniLM-L6-v2')`, which efficiently transforms text into dense, fixed-size embeddings. This model is optimized for sentence-level embeddings, capturing the contextual meaning of the text and performing well with shorter texts such as product reviews. By leveraging pre-trained models, we can quickly generate meaningful representations of review data, enabling tasks such as semantic similarity comparison and enhanced retrieval performance.

To save computation time during repeated queries, the embeddings are computed once and stored in a numpy file. This eliminates the need to re-run the embedding process each time a query is executed, making the system more efficient for large datasets. However, the trade-off is an increased storage requirement, as the embeddings are stored as large numerical arrays. Despite this, the time saved during future queries justifies the trade-off, making it a valuable method for optimizing review retrieval and analysis processes.

F. Sentiment Analysis using a Pre-trained BERT Model. To analyze the sentiment of text data, a pre-trained multilingual BERT model is utilized (different from the one used while embedding). The model processes textual inputs by tokenizing them, preparing the data for sentiment scoring. For texts that exceed the maximum token limit, the content is truncated to ensure compatibility with the model's constraints while preserving as much meaningful information as possible.

The sentiment scoring process outputs labels such as *"1 star"* or *"5 stars"*, which indicate the overall sentiment conveyed in the text. This approach is effective for handling diverse text inputs and ensures accurate sentiment evaluation, even for lengthy content.

This process is implemented in the `get_sentiment_score` function within the `codes/bigrams_with_BERT/visualization.py` file.

G. Executing the Method. To refine the review retrieval process, the sentiment of the specified opinion is first identified by locating its associated reviews and extracting the sentiment score of the corresponding review. The retrieved reviews from the expanded query are then filtered based on their alignment with the identified sentiment of the opinion. For instance, if the sentiment of the opinion is rated higher than "3 stars," only reviews with a sentiment score of "3 stars" or above are retained.

Subsequently, semantic similarity between reviews is computed using the FAISS (Facebook AI Similarity Search) library. FAISS is particularly advantageous for this task as it is optimized for fast nearest-neighbor searches in high-dimensional spaces, such as the embedded vectors generated by BERT-based models (FAISS). Its seamless integration with pre-trained models ensures efficient and accurate similarity calculations. By leveraging FAISS, the nearest neighbors to the query—comprising all expanded query terms—are identified. Finally, the top 25% of the most similar reviews are selected, ensuring that the output is both semantically relevant and sentimentally aligned with the query. Using the top 25% of reviews helps mitigate an overload of retrieved documents caused by the OR operations among the 15 terms in the expanded query. Additionally, the use of a percentile threshold, rather than a fixed similarity score (e.g., 0.5), ensures stability in the number of documents returned.

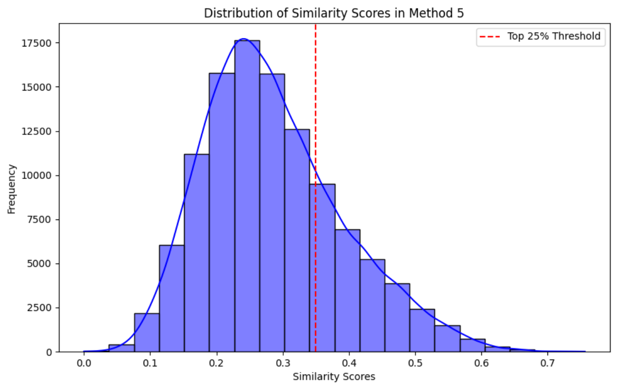

Fig. 6. Distribution of Similarity Scores.

The similarity scores generated during the process exhibit a distribution that closely resembles a normal distribution. This pattern indicates a balanced spread of semantic similarity values across the retrieved reviews.

To compare the results from Method 1 and the current approach, a Venn diagram is presented. The diagram highlights a substantial overlap between the two methods. This overlap is expected, as both methods rely heavily on the use of OR operations in their retrieval strategies, leading to significant intersections in the results. Despite this, the current method refines the results further by incorporating sentiment filtering and semantic similarity ranking, enhancing the precision of the retrieved reviews.

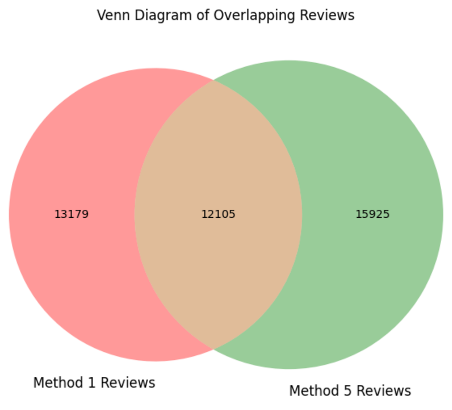

Fig. 7. Venn Diagram of Reviews (Method 1 vs Method 5).

The following presents the number of reviews retrieved for a selection of example queries (as done in the previous section):

Query (aspect:opinion)	# of docs returned
audio quality:poor	7897
wifi signal:strong	2170
gps map:useful	1030
image quality:sharp	6152

The number of documents retrieved in this method appears to be much higher compared to the previous method, suggesting that the recall has been significantly improved. This increase in recall indicates that more relevant documents are being retrieved. However, the resulting word cloud contains a broader range of words that do not necessarily align with the targeted sentiment, leading to a potential decrease in precision 5. As a result, while the method may retrieve more relevant reviews overall, its precision may be lower than that of the previous method, which more tightly focused on sentiment-specific terms.

Results

To compare the results obtained from Method 4 and Method 5, we propose leveraging a sentence transformer or sentiment analysis model to evaluate the retrieved reviews against an approximated ground truth. Establishing the absolute ground truth would require manually annotating all reviews with sentiment labels, which is infeasible given the size of the dataset. Instead, we use the sentence transformer applied in Method 5 to assign sentiment scores to the reviews. These scores serve as a reference for evaluating the results of Method 4. Additionally, an alternative metric is required to evaluate the performance of Method 5 since it already employs a sentence transformer for review evaluation.

Method 4 Results. In Method 4, reviews containing both *aspect1* and *aspect2* are retrieved through a Boolean AND operation. These reviews are then filtered based on the sentiment of the *opinion*, as determined by the sentiment column in the dataset. This column is generated using the sentence transformer employed in Method 5. The implementation details are available in the `outputs/results.py` file.
To assess the sentiment of the opinion terms, we use a pre-trained BERT-based sentiment analyzer. This model helps refine the selection of reviews containing the specified aspects and opinion sentiment.

Query (aspect:opinion)	Precision	Recall	F1-Score
audio quality:poor	0.248	0.152	0.188
wifi signal:strong	0.604	0.348	0.441
gps map:useful	0.610	0.283	0.387
image quality:sharp	0.722	0.418	0.530

As expected, Method 4 demonstrates relatively low recall due to the Boolean AND operation, which restricts the number of retrieved reviews. This trade-off highlights the precision-focused nature of the approach.

Disclaimer. The results presented in the table are based on approximations using pre-trained BERT models and logical assumptions. For a more accurate evaluation, it would be necessary to manually review the entire dataset and label relevant reviews.

Method 5 Results. For this section, we propose comparing the results of Method 5 by defining the relevant query list as the union of reviews containing *aspect1* OR *aspect2* OR *opinion*. This list is then compared against the retrieved bigrams. Using this approach, we generate the following performance metrics:

Query (aspect:opinion)	Precision	Recall	F1-Score
audio quality:poor	0.443	0.442	0.442
wifi signal:strong	0	0	0
gps map:useful	0	0	0
image quality:sharp	0.437	0.153	0.227

The results indicate that for certain queries, the use of bigrams has a substantial negative impact on the retrieved indexes. This issue could stem from limitations in the naive method used to identify and evaluate relevant reviews. Further analysis and refinement of the evaluation methodology will be explored in the conclusion section to address these shortcomings.

Conclusion and Future Work

In this study, we explored two enhanced methods beyond traditional baseline techniques. Method 4, which utilizes Support Vector Classification (SVC), demonstrated higher precision compared to other approaches; however, its recall was significantly impacted. To address this, future work could focus on refining the composition of boolean queries by experimenting with alternative query formulations such as `(aspect1 AND aspect2) OR (aspect1 AND opinion)` or `(aspect1 AND aspect2) AND (opinion OR sentiment)`. Additionally, the inclusion of bigrams in the query expansion process could further improve performance by capturing more nuanced relationships between terms.

The method employing BERT models (Method 5) yielded much higher recall, but its precision was compromised due to the reliance on general bigrams. To improve precision, one potential direction is to use sentiment-specific n-gram extraction methods to better align bigrams with the target sentiment, as opposed to relying on generic collocations. Furthermore, incorporating pre-existing, domain-specific bigrams could improve the relevance of the expanded queries, leading to more accurate retrieval. Another strategy for enhancing precision could involve adjusting the number of bigrams used for query expansion, focusing on a smaller set of high-quality, sentiment-aligned bigrams. Additionally, applying weak supervision techniques in the training of BERT models, as demonstrated in (2), could be explored. Although this approach showed lower recall, it achieved high precision in toxicity text detection, suggesting its potential applicability to sentiment-specific text classification tasks.

Bibliography

1. The analysis of which model performed better was mainly derived from chatgpt with a little of my guidence.
2. Vaibhav Jain and Mina Naghshnejad. Entity at SemEval-2021 Task 5: Weakly Supervised Token Labelling for Toxic Spans Detection. *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 935–940, 2021. doi: 10.18653/v1/2021.semeval-1.127.

While most of the analysis was derived from the lecture notes and documentation, ChatGPT provided valuable assistance in applying packages and techniques that were not specifically covered in the lecture materials. This helped in expanding the scope of the analysis and implementing more advanced tools for tasks such as semantic similarity computation and efficient search operations, ultimately enhancing the overall methodology.