

NLP Review Project

Overview:

With a huge number of products moving through Amazon every day, customer reviews are one of the few ways shoppers can get genuine reviews of the product before they buy. The problem is that these reviews aren't always filtered well, so many of the results or reviews that appear have nothing to do with what the customer actually cares about. That makes it harder to find the information that will lead the shopper to a final buying decision. This project aims to fix that, by focusing on retrieving reviews that are actually relevant to the customer's interests.

The main goal of this project is to implement and test three separate retrieval models the baseline boolean search, method 1 which introduced a polarity search, and method 2 which builds on method 1 and introduces proximity matching and title relevancy. Comparing these 3 methods we are able to test to see which ones produce the best result, i.e. the most relevant reviews to the query at hand.

Background:

For this project we are building off the basic ideas of how information retrieval works and how we can use boolean search and proximity matching techniques are used to ensure relevancy in outputs. Usually, when searching users are given large documents of outputs with retrieved reviews that technically meet the requirements of the boolean search but have little to nothing to do with what the end user cares about. Amazon's reviews are no different, as when a user goes to search something about a product. For example, image quality or audio quality can

return multiple reviews that while they meet the requirements of the search, the review itself has the chance to not be useful as it is about something that is completely different. For example, you could search about a camera having a sharp image quality, and be given reviews that have the use of the opinion “sharp” in a completely different context.

The problem this project focuses on solving is the ability for a search engine to ensure relevant reviews to the user when specific aspects are searched. Instead of just returning things that simply contain the matching keywords, it aims to return reviews that are closer to what the customer is truly intending to find.

Finally, my project does not implement anything new or novel instead it just compares simple techniques that were given by the assignment document: Boolean search(Baseline), polarity filtering(M1), and proximity matching and title relevancy(M2). Showing how each added step improves the relevance of the reviews that are returned.

Methods:

The three methods I implemented into my project are the basic boolean search, method 1 which is the polarity filtering and finally method 2 which builds off the previous two methods and introduces proximity matching and title relevancy. I chose these methods because they were the most straightforward to implement in Python and clearly showed how each additional layer added improved the results returned to the user.

Design:

This project was entirely in Python using simple text-processing tools. I used Pandas to load and work with the dataset of reviews, and relied on built in Python functions like .lower()

and `.split()` to normalize the text and ensure the text was preprocessed before it was used in any fashion. Even with only these built in tools, the program is still able to scan and filter thousands of reviews quickly and reliably. These tools made the cascading pipeline easier to implement since they are all given a clean starting point in terms of the input that is received.

Boolean Search:

In my implementation, since each query is written like `<aspect> : <opinion>`, For example `gps map : useful` or `image quality : sharp`. I first split the query into two pieces, on the colon and treat the first part before the colon as the aspect and the second part is treated as the opinion word.

All reviews that are seen by the Boolean search are lowercased and concatenated into a single searchable string. Then the baseline Boolean search checks and compares each review using pure Boolean rules. Using these rules it is able to determine if the proper aspects and opinions are present in the review, and if the rules are met the program returns it as a match. However if it does not it is marked as false and moved on to the next review in the dataset. In addition, depending on which test it is doing, it will use a different combination of Boolean rules to show how different combinations can pull improved or worse results.

This method does not have any other things like proximity searching or polarity filtering built into it, this method only checks to see if the desired aspects and opinions are present in the review.

Because of this, the baseline retrieves anything and everything that contains the queried aspects and opinion words, even if the actual content of the review is completely useless to the user.

Method One:

My method one implementation takes the same core structure of the baseline Boolean search but adds the idea of polarity filtering. So, now in addition to the split query, each query is assigned a polarity label(either positive or negative) based on the opinion word, For example, for the query audio quality : poor, “audio” and “quality” will be treated as aspects, and “poor” will be the treated as the opinion word, so the whole thing will be then assigned as a negative polarity. A positive example would be along the lines of gps map : useful.

The polarity filter works by first checking the reviews star rating before it does any text matching at all. If the opinion is positive it only checks for reviews that are rated with 4 or 5 stars, and if the review is negative it looks for ones that are rated with 2 stars or less. Any review that is in-between that range is ignored and discarded since it does not directly match the intent behind the opinion word, rather it sits in the gray area in between.

After the polarity check, it takes all the reviews that were retrieved and then behaves like a stricter version of the baseline Boolean search. It lowercases the review text and checks to ensure the reviews contain the desired aspects and opinions. If a review makes it past the polarity check but does not contain the desired query aspects and opinions it is discarded and ignored otherwise it is returned.

By forcing this new filter, Method one is able to remove the reviews that technically contain the desired aspects and opinions, but contradict the opinion’s intent. Making the output more aligned with what the user actually wants to read.

Method 2:

Lastly, I have my implementation of method 2 which cascades off of the polarity rating from method 1 and adds proximity matching and title relevancy in order to make the results appear that much closer to what the user wants.

First it still enforces the polarity filtering by checking the star rating of the review and considering the positive reviews to be the ones rated 4 or 5 stars and the negative ones to be 2 stars or lower, all the other reviews in-between are ignored and discarded.

After the rating check, the method looks at the review title, the title is lowercased, method 2 checks if the title contains at least one of the aspect words and one of the opinion words. If both are found in the title, the review is instantly returned as relevant. Using the idea that if the users concern is in the title, then the review itself will almost always be relevant.

If the title itself does not have a strong match, Method 2 then falls back to proximity matching. The text is tokenized into words, and then searched for positions where an aspect word and an opinion word are within a window of each other(in my case 10 words). If any aspect-opinion pair appears within the set distance it is marked and returned as a match, if not it is discarded and ignored.

By combining these ideas-rating based polarity filtering, title relevance, and aspect opinion proximity. Method 2 is much stricter than the baseline Boolean search and it no longer accepts reviews that happen to contain the right words in the review's text. Instead, it prefers reviews where the opinion is clearly being expressed about the given query and the right sentiment.

Results:

To evaluate the methods, I used five queries that combine two aspects words with a single opinion word. For each of the methods I first sampled the top 60 returned reviews and manually judged them to check to see if they were relevant to the query itself. I then scaled those ratios to estimate the relevancy across the true amount of reviews that were returned as matches per each method.

Below are the results of each of the methods using different queries:

Query	Baseline (Boolean)			Method 1 (M1)			Method 2 (M2)		
	# Ret.	# Rel.	Prec.	# Ret.	# Rel.	Prec.	# Ret.	# Rel.	Prec.
audio quality:poor	1882	320	0.17	979	215	0.22	853	273	0.32
wifi signal:strong	304	91	0.30	218	87	0.4	112	72	0.64
mouse button:click problem	2681	152	0.07	661	46	0.07	66	13	0.2
gps map:useful	340	110	0.34	233	86	0.87	30	20	0.67
image quality:sharp	1103	529	0.48	847	364	0.43	170	65	0.38

After looking at all the results, the Boolean search continues to return the most amount of reviews marked as true and returned, but with that it also has the lowest precision percentage. The amount of returns is so high because the Boolean search just checks to make sure that the review contains the queried words themselves. Which in turn shows why the relevancy is so low, showing why as a baseline it is good but in actuality it has a hard time ensuring that returned reviews are actually relevant or useful.

Method 1 reduced the amount of reviews that were returned by introducing the rating threshold or the polarity filter. That checks the polarity by ensuring that the positive reviews are above a 4 star rating and the negative reviews are below a 2 star rating. This helps by immediately cutting any reviews that do not fit into the rating scale that it is asking for, ensuring

that the right sentiment is achieved. This alone increased the precision of the model compared to the baseline.

Method 2 combined the polarity filter with two additional things: title relevancy and aspect-opinion proximity within the text. The titles that mention both the aspect word and the opinion word are instantly returned. If not, the method then looks into the body of the review and checks to see if the aspect word and opinion word appear within 10 words of each other. This made Method 2 much stricter than the other two, reducing the retrieval amount dramatically. Despite the drop in retrieval count, it still proved to return the most relevant reviews in most of the categories.

Discussion:

These results show a clear trend, as more constraints were added by each of the methods the relevancy of the returned reviews increased. The baseline pulled the most reviews but had the lowest precision because it matched words without care for context or sentiment. Method 1 improved precision by filtering the reviews based on their rating. Lastly, Method 2 was the most strict, using polarity, title relevancy and proximity matching and while less reviews were returned, it had and maintained the highest precision in most, if not all, the categories.

Conclusion:

In conclusion, the project showed how much review quality can change depending on the filters implemented on top of the baseline Boolean search. The baseline alone proved that keyword matching alone is not enough to guarantee useful results. Adding the polarity filter to Method 1 improved the precision of the search a bit, but Method 2 pushed the precision accuracy

the most with the additional proximity matching and title relevancy on top of the polarity filter.

Though none of these models were complex, the progression made it clear how each added layer helped return reviews that were actually relevant to the user's query.