CMPT 412: EMMANUELLA EYO & ZOLI DIGNESS

Sentiment Analysis: Unveiling the Emotional Pulse of Amazon Product Reviews

Abstract

The project “Understanding the Influencing Factors and Sentiment Analysis of Amazon Product Reviews”,thoroughly investigated how online reviews affect consumer decisions and businesses' standing. This document covers key concepts, features, social computing concerns, utilized technologies, implementation structure, test outcomes, and a thoughtful evaluation of challenges faced and avenues for enhancement within the project.

Introduction

- **Main Idea and Functionality**:

  The significant influence of online reviews on both consumer choices and the standing of a business underscores the importance of delving into the sentiments conveyed in Amazon product reviews. In this segment, we examine the approaches used in performing sentiment analysis, uncovering the emotional nuances that contribute to user feedback.

- **Research Questions:**
  - What are the key factors influencing the positive or negative Amazon reviews?
  - How can we use sentiment analysis techniques to categorize reviews into their various sentiment?
  - What anomalies can be detected and how can they be distinguished from genuine reviews?
  - What is the relationship between review sentiment and the product's overall reputation?

Main Idea & Functionality of the Project

The main idea centers around understanding the sentiment and dynamics of Amazon Reviews, transcending conventional analysis by incorporating sophisticated techniques. Rather than merely assigning numerical ratings, the project aims to delve into the nuanced details of user sentiments, identifying key factors that contribute to both positive and negative feedback.

Functionality:

- Data collection
- Utilizing text and sentiment techniques, helped to place these reviews into three primary sentiments: positive, negative and neutral. We look at instances where the sentiment score contradicts the numerical rating assigned by the user. For example, a high rate review with a low/negative sentiment score.

What Social Computing Issues are Addressed in this Project?

This project looks into various social computing issues, these include:

Authenticity:

- **Challenge**: ensuring the authenticity of user review is a critical challenge. Due to the increase in biased reviews, fake accounts or even paid or influenced reviews, it has become key to distinguish genuine review feedback from potentially influenced feedback.

    - **Approach**: the use of NLP tools, such as stemming, stop-word removal, and tokenization was key to textual cleaning and preprocessing. Additionally, the use of NER (named entity recognition) was used to add contextual understanding.
- **Reputation**:
    - **Challenge**: uncovering how different percentages of positive or negative reviews influence a product's reputation or user engagement with the product.
    - **Approach**: The project employed advanced sentiment analysis techniques, including the use of both rule-based (Vader) and machine learning-based (RoBERTa) models. By categorizing reviews into positive, negative, and neutral sentiments, we aimed to quantify the sentiment distribution and explore its correlation with the overall product rating.
- **User Motivation**:
    - **Challenge**: understanding why users leave reviews, what they experienced with a product, is a real issue that needs to be addressed.Motivation ranges from personal experience to external factors (promotions, gamification). Investigating these, give more insight into the dynamic of user engagement.
    - **Approach**: The project delves into the motivation and trends behind users on Amazon through contextual analysis of the reviews. By leveraging NLP and sentiment analysis, we uncover implicit cues in the language used by reviewers, shedding light on the underlying motivations, whether they be altruistic, self-expression, or influenced by external factors.

Technology and Techniques Used

Review data:
The data was collected by UCSD in 2018 and covered multiple review categories: electronics, pet-supplies, video games, and books. The amount of reviews in each category ranged from 500k to 10m. The format was originally in JSON format, but we converted it into CSV using python before pre-processing. Important tags were 'overall' which was the rating of the review ('5.0', '4.5', etc.), and 'reviewText' which was the string of words in the review.

Pre-processing:
NLTK with Python aided all of the pre-processing. The review strings were tokenized into a list of words (e.g. "A review" → ["A", "review"]. The tokenizer used was specifically a regular expression tokenizer which allowed us to remove punctuation as well. We then removed stop words which are filler words that we don't need to analyze, "A" would be removed from the previous list. Finally the list needs to be lemmatized, this means that different inflections of the same word are converted to that word. Having all of 'changed', 'changes', and 'changing' turn into 'change' helps analyze real frequencies of words. After all of this each review will be trimmed into a list of important words.

Text Analysis:
Python was used for all of the coding. The techniques used were the bag of words model, and the vector space model. With the bag of words model you just count the occurrence of every word in every review. We used this model by separating words into different lists for initial

surface level analysis. First we created high-rated reviews (3+ stars), and low-rated reviews (<3 stars). This would give us a general overview of common ideas surrounding good and bad reviews. Next we created another list that was gonna hold words that appeared in reviews with 10+ upvotes. This distinction would provide insight on what reviews the general public finds useful (list of upvoted words) vs. what reviews the general public actually give (list of all review words). Next we attempted to create a comprehensive list of words describing popular review categories. For example “shipping”, “delivery” and “package” would all be designated as the shipping category. Going through all of the reviews and incrementing the categories count would give insight on what customers value the most in an amazon product. Finally we created lists for items that had many reviews (3,000+) and little reviews (<100). This was used in anticipation that frequently reviewed items would have a lot of useless reviews that didn’t say much because the item was so reputable.

The vector space model would instead treat each word as a ‘feature’ and give more complex insights past just term occurrence. Firstly we wanted to separate reviews with rare words, to do this we used IDF (inverse document frequency). We took the 10,000 rarest words and compared how often upvoted and non-upvoted reviews had at least one. Knowledge we wanted to gain from this was if companies should be treating these upvoted reviews as a very important tool to see what the public wants. Next we wanted to see how similar reviews were, for that we needed to use cosine similarity which is a scale from 1.0 → 0.0 with 1 being the exact same sentence. Cosine similarity tested on the set of upvoted and non-upvoted reviews, and on high and low reviews. This would give insight on which reviews are more specific, as it is harder to have the same sentence if you have a unique point of view.

Sentiment Analysis:

- Two distinct approaches used are the VADER and RoBERTa models
  - VADER Sentiment Analysis: a lexicon and rule based sentiment analysis tool that provides compound scores for overall sentiment categorization
    - Positive sentiment:

```
text = "Today is a wonderful and joyful day"

vader_sentiment(text)
```

```
{'neg': 0.0, 'neu': 0.345, 'pos': 0.655, 'compound': 0.8225}
```

    - Negative sentiment:

```
text = "I do not like this product, it broke as soon as i opened the package"

vader_sentiment(text)
```

```
{'neg': 0.309, 'neu': 0.691, 'pos': 0.0, 'compound': -0.6007}
```

- Neutral sentiment:

```
text = "The product met my expectations in terms of performance and quality.\
 It functions as described, and I haven't encountered any issues so far. \
 However, it doesn't stand out significantly from similar products on the \
 market but some aspects could be improved for a better user experience."

vader_sentiment(text)
```

```
{'neg': 0.0, 'neu': 0.84, 'pos': 0.16, 'compound': 0.8402}
```

- RoBERTa Sentiment Analysis: (Robustly Optimized BERT Approach) a transformer-based model fine-tuned for sentiment analysis tasks, offering nuanced sentiment scores. It handles a large
  - Positive sentiment:

```
text = "Today is a wonderful and joyful day"

roberta_sentiment(text)
```

```
{'roberta_neg': 0.0013733781,
 'roberta_neu': 0.0069114724,
 'roberta_pos': 0.99171513}
```

  - Negative sentiment:

```
text = "I do not like this product, it broke as soon as i opened it"

roberta_sentiment(text)
```

```
{'roberta_neg': 0.9820971,
 'roberta_neu': 0.01512042,
 'roberta_pos': 0.0027825548}
```

  - Neutral sentiment:

```
text = "The product met my expectations in terms of performance and quality.\
 It functions as described, and I haven't encountered any issues so far. \
 However, it doesn't stand out significantly from similar products on the \
 markebut some aspects could be improved for a better user experience."

roberta_sentiment(text)
```

```
{'roberta_neg': 0.1495027, 'roberta_neu': 0.38939306, 'roberta_pos': 0.4611042}
```

- Merging Functionality: The results from both sentiment analysis, and text analysis were merged and used to perform its analysis and create a comprehensive data frame and patterns and correlations. For example the matplotlib, seaborn scipy libraries in Python.

Methods used for data-analysis

Text analysis:

Because we couldn't run the program forever and there were way too many reviews for some methods (cosine similarity) we would sometimes take the first 10,000 reviews only. This meant we had to first shuffle the reviews so that we didn't get a run of all the same items and had a good variety.

```
random.shuffle(reviews)  # shuffle the reviews
some_reviews = reviews[:10000]
```

NLTK offers a frequency distribution method on Python lists. This allowed us to visualize many of the bag of words approach lists we created very easily.

i.e. the low rated and high rated reviews have all of the words seen appended into low_words and high_words:

```
low_words = nltk.FreqDist(low_words)
high_words = nltk.FreqDist(high_words)
```

```
[('graphic', 356), ('level', 315), ('bad', 257),
, 160), ('gameplay', 160), ('playing', 156), ('p
147), ('version', 146)]
```

In the vector space model we needed an algorithm to turn the words into vectors and compare each review. In the following code data[0] and data[1] are 2 different word tokenized reviews. First it creates the vectors in the first for loop, then it calculates how similar each vector is in the next for loop and returns the average.

```
def cosine_sim(data):
    X_list = data[0]
    Y_list = data[1]
    l1 = []
    l2 = []
    # form a set containing keywords of both strings
    rvector = X_list + Y_list
    for w in rvector:
        if w in X_list:
            l1.append(1)  # create a vector
        else:
            l1.append(0)
        if w in Y_list:
            l2.append(1)
        else:
            l2.append(0)

    c = 0
    # cosine formula
    for i in range(len(rvector)):
        c += l1[i] * l2[i]
    cosine = c / float((sum(l1) * sum(l2)) ** 0.5)
    return cosine
```

Sentiment analysis:
The sentiment process begins with importing the VADER lexicon using NLTK and initializing the SentimentIntensityAnalyzer. The function ‘vader_sentiment” is used to conduct this sentiment analysis of a text. It proceeds to calculate the polarity scorers (‘neg’ for negative, ‘pos’ for positive, and ‘neu’ for neutral) of the given text.

```
# Sentiment Analysis using Vader
nltk.download('vader_lexicon')
sia = SentimentIntensityAnalyzer()


def vader_sentiment(text):
    if isinstance(text, str):
        return sia.polarity_scores(text)
    else:
        return {'neg': 0.0, 'neu': 0.0, 'pos': 0.0, 'compound': 0.0}
```

To get a second opinion, we used the pretrained RoBERT tokenizer and model (**cardiffnlp/twitter-roberta-base-sentiment**). The function, to prevent error, limits the text length, processes the text by truncating to fit the limit. Then it tokenizes the text into RoBERTa compatible tokens and converts into PyTorch tensor for model input. The scores are fed into softmax to be normalized and returned for use.

```
# Sentiment Analysis using RoBERTa
MODEL = f"cardiffnlp/twitter-roberta-base-sentiment"
tokenizer = AutoTokenizer.from_pretrained(MODEL)
model = AutoModelForSequenceClassification.from_pretrained(MODEL)


def roberta_sentiment(review_text):
    # Limit the text length to avoid potential errors
    max_length = 512  # Define the maximum length for BERT input

    # Truncate or split the text if it's too long
    text = review_text[:max_length] if len(review_text) > max_length else review_text

    # Get BERT embeddings for the processed text
    encoded_text = tokenizer(text, return_tensors='pt', truncation=True, max_length=max_length)

    # Use the model to get the sentiment scores
    output = model(**encoded_text)
    scores = output[0][0].detach().numpy()
    scores = softmax(scores)
    return {
        'roberta_neg': scores[0],
        'roberta_neu': scores[1],
        'roberta_pos': scores[2]
    }
```

To merge and accumulate the result from both text and sentiment analysis

```
merged_results_df = pd.DataFrame(results).T
merged_results_df.reset_index(inplace=True)
merged_results_df = merged_results_df.merge(df, left_index=True, right_index=True, suffixes=('_merged', '_original'))
```

Results and discussion:

For the bag of words model there was a clear distinction between the leisure items (video games and books) and non-leisure items (electronics and pet supplies). High leisure item reviews saw a huge increase in proper-noun usage, for example author names. In contrast, low reviews had ‘author’ as a top 5 used word. This shows that when it comes to items like books and video games a large portion of the users praising the product are dedicated fans of the author or publisher. Productivity and non-leisure items like electronics and pet-supplies had reviews that spoke mainly about price and reliability of product. Low reviews had ‘work’, and ‘broken’ in the top 5 words. High reviews had ‘quality’, and ‘price’ in the top 5 words. Because a

majority of reviews are positive, this shows people only leaving negative reviews if a product really doesn't function. It is imperative that you put out a functioning product as advertised or you will start off with many 'broken' reviews that take you off buyers' radars.

The results for sorting reviews by categories are shown below. Interesting takeaways for this is that price was the most common topic on average. This highlights how vast Amazon's catalog is, when so many products exist users just want to know if this one is worth the price or if they should just move on to the next item. Sellers need to make sure they are competitively priced or they will be weeded out quickly as it's the most common topic.

Often reviewed items had an overwhelming amount of 'price' as a word. It was by far the most common word and highlights this exact category imbalance. For scarcely rated items it was the opposite, a lot of the words were very rare and price didn't have nearly as much of a share. Items have 2 life cycles: being actually studied and diagnosed by the buyers for its function, and being a reputable item that is only studied through price.

Average for 10,000 reviews per item category:

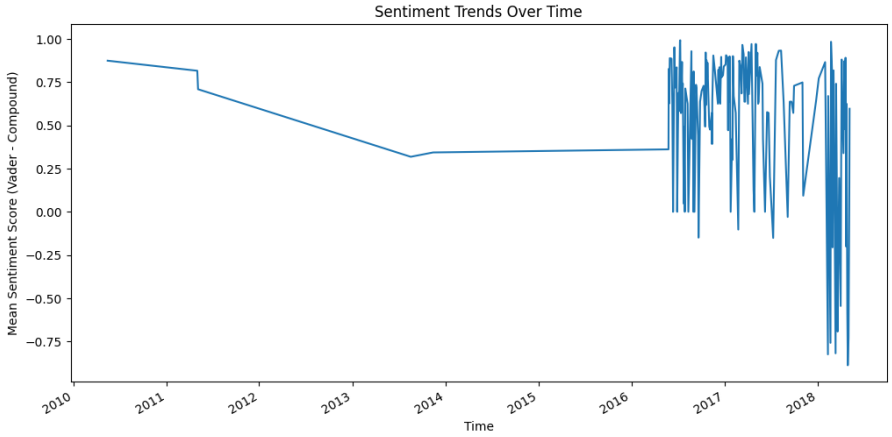
With vector space and IDF, interesting results were that almost every single upvoted review contained one of the 10,000 rarest words. For comparison, in the set of all reviews less than half contained a rare word. These words were more eye-catching and made the reviewer seem credible and experienced (i.e., 'stitching', 'heftier', 'outweighs').

```
221 voted reviews had rare words out of: 242
4128 reviews had rare words out of: 9999
```

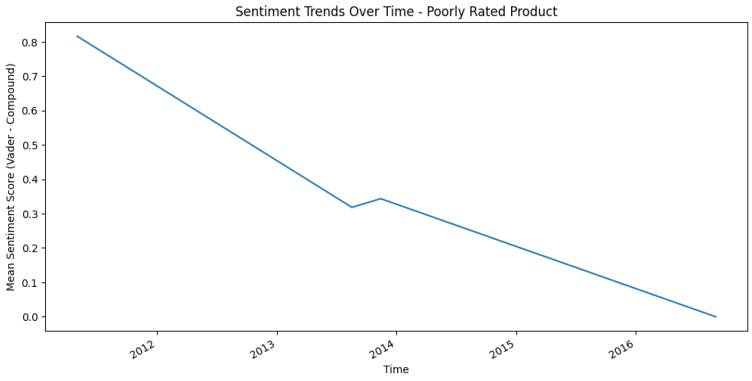
It should be the goal of every company to break into the thousands of reviews range where the item is flooded with comments about how the product is worth it. To do this it is very important for companies to competitively price and advertise products as they actually are. Positive reviews are the most common because if something works, and is priced reasonably, buyers

find it worthy to review. Negative reviews contain the most unique substance as shown from the cosine similarity being much higher for positive reviews. Buyers should then focus on negative/upvoted reviews because these will shed light on unique aspects of an item beyond price and reputation.

```
average cosine similarity for low reviews: 0.30110779763607887
average cosine similarity for high reviews: 0.3987486547495028
```



This highlights the sentiment trend of highly rated products for a 100 beauty products reviews. For a few years there was not much happening, but eventually, there was more engagement with these products.



But for the lowly rated beauty products, all we see is a continuous decrease between the years. This indicates with sentiment scores, we can see how a product does over time with user engagement. As expected, low rated items do not do very well because of their poor ratings, which makes this plot reliable for analysis.

```
reviews.query("overall == 4.0").sort_values('pos', ascending=False)
r
```

1 entry (filtered from 12 t

level_0	index	neg	neu	pos	compound	roberta_neg	roberta_neu	roberta_pos	overall	reviewerID	reviewText	Sentiment_Tag	R_sentimentTag
151	151	0.318	0.427	0.255	-0.1494	0.03288811817765236	0.12580615282058716	0.8413057327270508	4.0	A2WNVJ6S7OVZP4	It worked very well i disliked nothing	Negative	Positive

As we can see from the above image, the code snippet selects reviews where the overall rating is exactly 4.0 and then arranges these selected reviews based on their positive sentiment score from the vader model ('pos') in descending order, showing the most positively rated reviews with an overall score of 4.0 at the top. We used this to query for discrepancies, although most text were categorized accordingly.

The index 151 among others highlighted a discrepancy, the vader model considers this negative, while RoBERTa considers it positive. From reading the text, it comes off across as a positive statement. WHY?

VADER Analysis: VADER assigns a compound score of -0.1494. This negative compound score implies a slightly negative sentiment overall. VADER interprets this based on various linguistic rules and a lexicon of words with known sentiment scores. It also breaks down the text into negative, neutral, and positive components. Here, the negative score (0.318) is higher than the positive score (0.255), contributing to the overall negative compound score.

RoBERTa Analysis: The RoBERTa model assigns sentiment probabilities to different categories (negative, neutral, positive). In this case, the 'roberta_pos' score is notably high at 0.841, indicating a strong likelihood of positive sentiment.

Phrase Interpretation: VADER might focus on the phrase "i disliked nothing" as negative due to the word "disliked," while RoBERTa might consider the overall context, giving more weight to the phrase "It worked very well," resulting in a positive sentiment.

Reflection on Obstacles and Areas for Improvement:

Reflection:

- Understanding the context of words posed a significant challenge. We struggled to confirm context without manually crafting scripts to check the surrounding words' conformity to our expectations. Even creating keywords to categorize reviews (like shipping or quality) was a laborious task, encountering similar issues where numerous false positives or negatives were overlooked during sorting.
- This project shed light on several analysis hurdles, such as the need for robust data cleaning and encountering runtime errors during analysis. Some reviews were missing altogether, limiting and complicating the analysis process. Upon reflection, addressing these challenges involves incorporating additional sentiment analysis models for enhanced accuracy, refining error handling, and incorporating a broader range of review data for a more comprehensive analysis.

Areas for improvement:

- We can use more available reviews. There was an issue of some products did not exist on Amazon, so this made analysis slightly difficult
- Another area of improvement would be looking for more advanced techniques and models for analysis.
- And lastly, to allocate more time to the given task and project.