Mining Amazon User Reviews for Product Features and Sentiment

Final Report

Nicholaus Hooper

1. Executive Summary

This project presents a comprehensive analysis of Amazon user reviews for a specific product category using sentiment analysis methods as well as clustering. The overall goal was to gain insight into the relationship between user reviews and quantified sentiment for the purpose of extracting insights to better inform business decisions.

The dataset used consisted of a large collection of user reviews and product metadata of home office desk chairs listed on amazon.com. The two main methodologies used were sentiment quantification and clustering. By using a lexicon-based sentiment analysis approach, sentiment could be quantified for the purpose of observing whether the textual content of user reviews correlates with the rating assigned by the user. The quantified sentiment scores were analyzed further after utilizing a clustering algorithm to group the reviews into contextually similar groups. The clusters were then analyzed based on their average overall sentiment scores to identify specific product features which received high sentiment and those which received low sentiment from users.

The sentiment quantification allowed for the observation of a positive correlation between the content of a user's review in relation to the assigned rating. Furthermore, the sentiment-based cluster analysis provided insights into customer sentiment toward products on a granular level. Overall, the techniques used in this analysis produced valuable insights that would prove useful to drive business decisions in areas such as product development and increasing customer satisfaction.

2. Data Extraction

The source of the data for the reviews for this analysis was obtained from a dataset created by researchers at UCSD[1]. The original purpose of this dataset was for an analysis of a recommendation system, but it is a substantial source of Amazon review data which will prove useful for many purposes. The repository of data, in its entirety, contains 233.1 million reviews

as well as product metadata in separate datasets for the products being reviewed, broken down into several separate datasets by general product category, i.e., “Home & Kitchen”, “Technology”, etc. Amazon organizes its products into broad categories such as these, and then further into smaller and smaller sub-categories. The target for this analysis is to analyze one type of product to identify positive and negative sentiment toward specific product features. The product chosen for this task is computer desk chairs. To begin narrowing down the available data, manual inspection of desk chair listings on amazon.com was performed to identify the highest and lowest level sub-category which contains computer chairs, which are categorized first under “Home & Kitchen”, and at the lowest level, “Home Office Desk Chairs”. The “review” and “product” datasets were downloaded for the “Home & Kitchen” category which contained 21,923,568 and 1,301,225 items respectively. Appendix A outlines the specific steps to extract the initial working review and product datasets of desk chair data. Table 1 and table 2 show the variables and size of the initial working datasets.

products.csv			
Variable name	Description	Type	Non-null
asin	product ID	string	2088
title	product name	string	2088
description	product description	string	2088
brand	brand name	string	1857
price	price in USD at time of retrieval	string	938

Table 1 – products dataset and variable descriptions

reviews.csv			
Variable Name	Description	Type	Non-null
asin	product ID	string	74681
reviewerID	ID of the reviewer	string	74681
overall	rating from 1 to 5	float	74681
summary	summary of the review	string	74673
reviewText	text of the review	string	74653

Table 2 – reviews dataset and variable descriptions

3. Data Profile

The most important aspect of data profiling for this analysis is the relationship between the reviews and product datasets. Figure 1 shows that 75% of the products have less than 15 reviews and the average is 35.19 reviews per product.

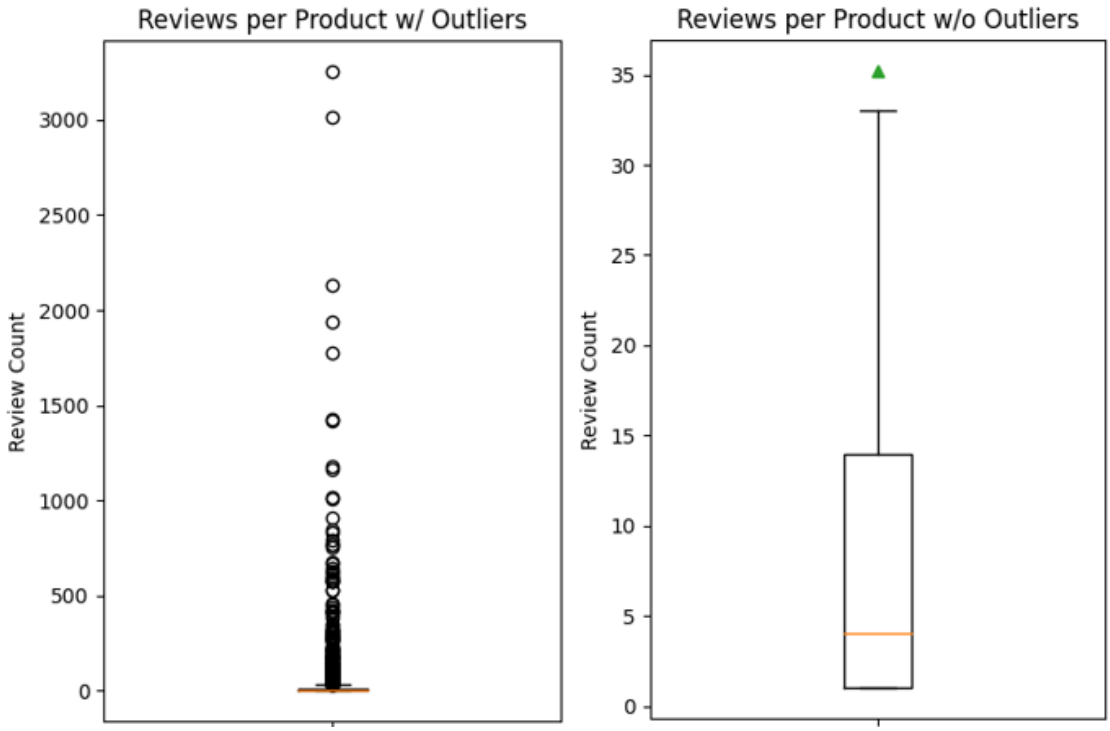

Figure 1 - Distributions of reviews per product. The green triangle on the right plot indicates the average

The next relationship between the reviews and products to investigate, since this analysis focuses on sentiment, is the distribution of the user ratings of the reviews. Figure 2 illustrates that the dataset has a skew toward the positive, with a large portion of reviews having 5 stars.

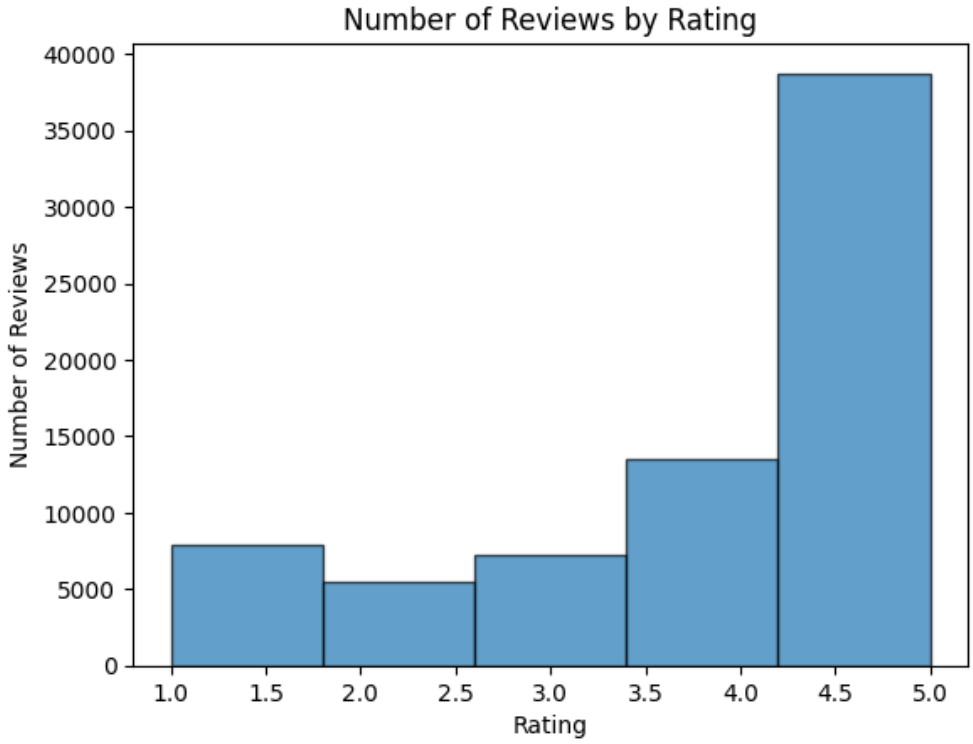

Figure 2 - Number of reviews by rating

The most important variable to profile is the 'reviewText' field of the reviews dataset, since the analysis will revolve around the clustering of this field. To analyze this field, a simple text preprocessing was performed to case-normalize and remove punctuation/numbers/special characters. The average word count of the reviews is 52.44 words and there is a standard deviation of 78.74 words. The longest review has 2,675 words and, after preprocessing, seven reviews were left with 0 words which were removed from the dataset.

The next element to analyze is the document frequencies of the terms in the reviews, i.e., the number of documents which contain a term. The reviewText field has 34,814 unique terms. The twenty with the highest document frequency are as follows:

```
chair : 0.6285972949987655
  and : 0.624687937230803
  the : 0.615003703602096
  it : 0.5504650078187155
  to : 0.5475569943211435
   a : 0.5151161833694549
   i : 0.5131546459630737
 for : 0.4710844695618776
  is : 0.4627993745027571
```

```
    this : 0.4414693698389619
     my : 0.34328276316150447
comfortable : 0.3300732490192313
      of : 0.32948341609283693
      in : 0.329373679734438
    very : 0.3128857918849963
     but : 0.2955748813475625
     not : 0.28497160571726426
     was : 0.2729829085621794
    that : 0.24888206084881073
      on : 0.2407890044168884
```

Interestingly, the term with the highest document frequency only appears in 62.86% of the total documents. Figure 3 below shows that the majority of terms occur in a low number of documents. Of these, manual analysis revealed many of these terms to be various typos and misspellings.

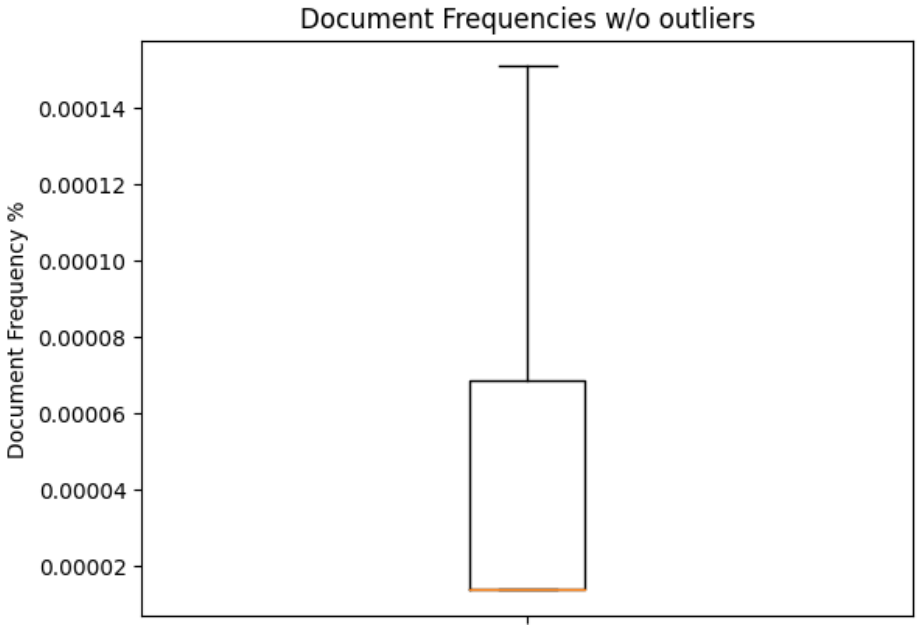

Figure 3 - Document frequencies of terms in the reviews

3. Methodology and Results

With the data prepared, the first goal is to observe whether the quantified sentiment in the content of users' reviews aligns with the ratings that they assign. The first step in the analysis is to quantify the sentiment in the user reviews. The tool that will be used for this is Valence Aware Dictionary for sEntiment Reasoning (VADER)[2]. In the field of sentiment analysis, VADER is

defined as a lexicon-based approach, meaning that sentiment is assigned by comparing a piece of content to a pre-made dictionary consisting of words or expression paired with sentiment scores. Sentiment is assigned by aggregating the sentiment of the individual words or expressions in a text. Lexicon-based approaches have many advantages including not requiring training data, or large amounts of data in general. VADER's lexicon is also specifically tailored to social media "blog-style" content, making it ideal for content such as Amazon user reviews. Appendix B further explains VADER's particular tailoring to social media.

The contents of the 'reviewText' field were processed using the VADER python library created by Hutto et al. and the resulting scores were appended to the dataset. No preprocessing was performed because VADER considers elements such as capitalization and punctuation when calculating sentiment. Next, the VADER compound scores, in the form of a continuous scale from -1 to 1, were segmented into 5 equal ranges and assigned scores from 1 to 5 to be evenly compared to the user ratings. Overall, VADER had an accuracy of 0.5249. Five star reviews were the most easily identified with an accuracy of .7828, the next most accurate was one star with .325 accuracy. The least accurate scores were the two star ratings. The bias toward 5 star could be due to the skew of the amount of 5 star reviews in the dataset, however, figure 4 illustrates a general positive correlation between the VADER assigned scores and the user ratings, therefore, a positive correlation between the content of a user's review and the corresponding rating which they assign.

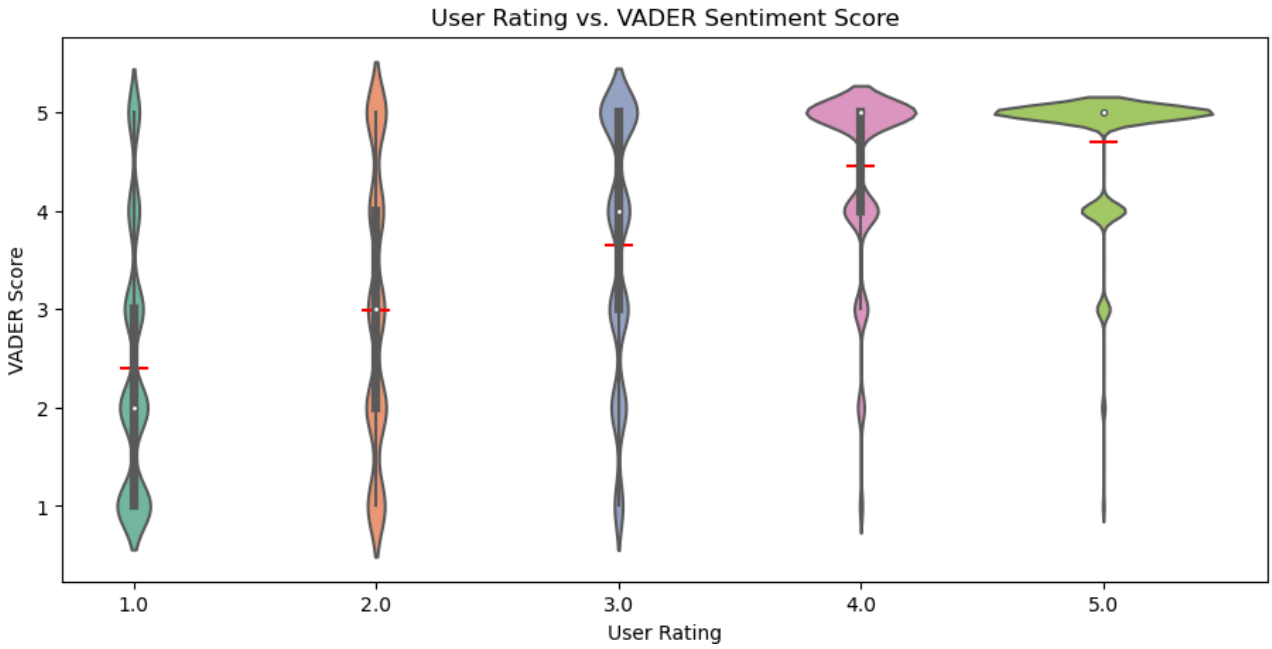

Figure 4 – Violin plot of densities of VADER scores for each user rating. The red lines represent the mean. The inner plots are standard box plots.

The next task which must be completed in order to perform clustering is creating word representations from the collection of user reviews. The chosen tool to accomplish this task is the fastText algorithm, developed by Facebook AI Research Lab in 2016 [3]. fastText has many unique features that make it a formidable option for vectorizing social media style content revolving around its sub-word embedding approach, which exceeds in handling out-of-vocabulary words and capturing syntax from similarly structured words. For this analysis, fastText will be utilized through the Gensim python library, which has many resources for text analytics[4]. Gensim's fastText implementation was used with most default parameters untouched, with a few exceptions. Appendix C further describes the technical details and the parameters used for generating the word embeddings.

With the word embeddings created, the next step to generate clusterings is to compile document vectors from the created word embeddings. This is done by simply taking the average of the word vectors of a given document. This allows reviews that are syntactically and semantically similar to be closer to each other in vector space, which will be observed by the clustering algorithm. The algorithm chosen to perform the clustering is Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN)[6]. HDBSCAN uses a density-based approach to identify clusters in data. Clusters are created based on high-density

areas in the vector space by generating a graph structure and setting a threshold of density to determine if points belong to a cluster or if they are noise. To utilize the algorithm, the hdbscan Python library created by McInnes et al. will be implemented[7]. Three different sets of hyperparameters were used to compare validation results of the clustering. On the topic of validation, since HDBSCAN is a density-based clustering algorithm, distance based measurements of cohesiveness such as silhouette score or the elbow won't be effective. Density-based clustering validation (DBCV) solves these issues by measuring the clusters with respect to density rather than distance and takes noise into account[8]. The hdbscan library includes a function of relative validity to allow the comparison of parameters. Due to computational restrictions, only three models were evaluated with two parameters tested. Table 3 illustrates the parameters chosen for testing and the resulting validity scores. The model with the highest scoring validity was that of the default parameters and will be used going forward for analysis. Appendix D provides in-depth detail into the steps to generate doc embeddings and perform the clustering.

min_cluster_size	min_samples	Validity score
5	0	0.3135
10	5	0.3091
20	10	0.3040

Table 3 – Comparison of HDBSCAN parameters by relative validity

The model produced 242 clusters plus one additional cluster for points not assigned to a cluster, i.e, noise. The resulting clustering classified 47,750 reviews as noise and the majority of the remaining reviews were placed in a singular cluster containing 20,942 reviews. Of the remaining clusters, the counts range from 5 reviews (min_cluster_size parameter) to 189 reviews. The VADER compound scores of the clusters range from -0.478 as the lowest scored cluster to .894 as the highest scored cluster. To analyze the textual information of the clusters, a keyword approach was used by calculating the TF-IDF of the reviews in each cluster to produce a list of the top 10 keywords using scikit-learn's TF-IDF vectorizer[9]. Table 4 contains the clusters with the lowest average compound VADER scores and table 5 contains the clusters with

the highest average scores. Technical details for performing the analysis can be seen in appendix E.

Rank	Keywords	Avg VADER Compound Score
1	Uncomfortable	-0.478
2	week, broke, chair, month, uu, terrible, product	-0.4575
3	Comfortable	-0.4439
4	comfortable, good, comfortable	-0.4153
5	Comfortable	-0.4023
6	uncomfortable, uncomfortable	-0.35
7	pay, guess, alright	-0.0707
8	muy, buena, buenas, es, silllaaaaaaaaa, el, la, calidad, excelente, lo	-0.0493
9	Comfortable	0.0
10	expected	0.0

Table 4 – Clusters with lowest VADER compound scores

Rank	Keywords	Avg VADER Compound Score
1	great, easy, comfortable, chair, assemble	0.894289
2	love, easy, comfortable, assemble	0.892920
3	super, easy, comfortable, assemble	0.886437
4	great, easy, comfortable, chair, assemble	0.885750
5	great, comfortable, looks, easy, assemble, good, comfy, chair	0.879287
6	support, good, comfortable, great, price, chair	0.873583
7	great, value, chair, good, nice	0.872530
8	great, comfortable, price, value, chair, nice, excellent	0.865500
9	love, great, chair	0.858233
10	great, price, chair	0.855188

Table 5 – Clusters with highest VADER compound scores

As seen from tables 4 and 5, the dataset's skew toward positive rated reviews presented some overlap for the lower rated clusters, however, some insights can still be observed. For example, the second cluster in the low VADER score table appears to contain reviews regarding overall quality and chairs breaking. Table 6 shows the original reviews that this cluster represents. Regarding the positive clusters, most appear to contain general notes about the

comfort and ease of assembly. Looking closer at cluster 6, the reviews also mention back support as being a positive feature, which can be seen in table 7.

Broke within a week.
Chair broke within a week.
broke within a week u.u
Broke within a week.
It broke within a week
chair broke within a week
Broke within a month
Terrible product. Broke within a week

Table 6 – Original reviews from a negative cluster

Very comfortable, with good back support.
Very comfortable. Good back support
Great chair and very comfortable with good back support.
Very comfortable great price and has good back support.
Comfortable and good back support
Comfortable and good back support

Table 7 – Original reviews from a positive cluster

3. Limitations and Future Research

The first limitation and consideration for future work is the dataset which was used for analysis. The original design for data acquisition included designing a web scraper to scrape amazon reviews to compile a larger quantity of reviews and have the control of the distribution of user ratings, however amazon.com prevents web scraping by only allowing users access to a limited number of reviews before only allowing searching reviews with a search bar. Amazon does offer a developer API to work with site data, however, it requires registration. Future work could use this API or another source to have more control over the data profile to overcome the issues of the skewed ratings distribution.

Hardware/computational capacity was another source of limitation. The phase most affected by this was running the HDBSCAN algorithm, which cost time. Ideally, a proper grid-search implementation could be used to compare the relative validity of different clusterings

produced by tuning more than two parameters. This could have allowed for a higher quality of clusters produced. Further work in this area could involve using a cloud-based system in order to handle the higher technical demand.

Lastly, different clustering or word vectorization methods could be explored to attempt to increase the quality of the results. The algorithms that were chosen for this analysis were chosen so because of their perceived advantages for the task at hand, however, in conjunction with the two previously mentioned limitations, exploring alternative algorithms and techniques could yield an overall higher quality model.

4. Conclusion

This project aimed to gain insight into customer sentiment by analyzing user reviews. By quantifying sentiment, one can perceive a deeper understanding of the underlying nature of customer satisfaction. Using VADER to compare review sentiment to user ratings revealed that there is a positive correlation between the textual content of user reviews and the corresponding ratings that are assigned. Knowing this, one can predict what kind of ratings a product might receive based on analyzing other textual sources such as social media posts and comments.

Using quantified sentiment for further analysis through clustering allowed for granular levels of customer satisfaction to be observed through similar reviews. Distinct clusters gave insight into sentiment toward specific qualities and aspects of product features. By identifying these trends, companies can better prioritize their strategies to action these areas.

Limitations are present in this current work. Model tuning restrictions and available datasets may have affected the overall outcome, the analysis performed here could be replicated and advanced to produce a higher quality model to receive more valuable insights and even be implemented as a system or application to allow for generalized analysis for any collection of reviews for any category of product.

Appendix A

Both the reviews and products datasets in json format from the “Home & Kitchen” category were unzipped and loaded into a local MongoDB database with a Python script to be queried easily. The code snippets below show the substring query method and the query and projection passed as parameters used to query the product database to obtain a list of unique ASINs (Amazon Standard Identification Number).

```
def mongodb_substring_query(client, database, collection, substring, query,
projection):
    client = MongoClient(client)
    db = client[database]
    collection = db[collection]
    result_list = []

    result = collection.find(query,projection)

    for item in result:
        result_list.append(item)

    client.close()
    return result_list
```

```
collection = "metadata"
substring = "desk chairs"

query = {"category":{"$elemMatch":{"$regex":substring, "$options":"i"}}}
projection = {"_id":0, "asin":1}

asin_list = mongodb_substring_query(client, database, collection, substring,
query, projection)
```

The reviews collection was then indexed on the ASIN element and then queried to retrieve all reviews that matched the ASINs list. The result was then saved to a csv file. The products from the ASINs list were also saved to csv . The code snippet below shows the function used to query the collections for items matching an element in the ASIN list.

```
def mongodb_asin_query(client, database, collection, asin_list):
    client = MongoClient(client)
    db = client[database]
    collection = db[collection]
```

```
    output = []
    count = 0

    result = collection.find({"asin":{"$in": asin_list}})

    for item in result:
        output.append(item)
        count +=1

    client.close()
    return output
```

Appendix B

VADER has several unique qualities that make it a formidable choice in sentiment analysis of social media content. Firstly, the dictionary of a standard lexicon was expanded by including “blog-style” elements such as “:)”, “:(“, or “LOL” and slang such as “grr” or “meh”. The lexicon used human raters to assign sentiment scores from -4 (extremely negative) to +4 (extremely positive) to features which were then averaged. These scores are used by the lexicon to quantify the intensity of the sentiment of a body of text. VADER was also made with five additional heuristics to evaluate sentiment intensity. They are; punctuation usage, word capitalization, degree modifiers (“good” vs “great”), polarity-shifting conjunctions (the sentiment which follows the conjunction is dominant, e.g., “the acting was bad, but the story was great”), and lastly, taking trigrams into account which may detect changes in polarity, e.g., “the movie wasn’t that bad” = positive despite “bad” being negative.

Appendix C

Most of the default parameters were left untouched when generating the word embedding model with two exceptions. The first updated parameter was the vector size, which controls the dimensionality of the model. Vector size was increased to 200 to allow for more depth of syntax, without increasing the computational cost too much, given the size of the corpus. The other updated parameter was setting the training algorithm to skipgram. This was changed due to the skipgram model better capturing semantic relationships [5]. Skipgram is more computationally expensive, however, this liberty can be taken since the corpus size is not too large. The remaining notable parameters were appropriate for the task and remained at their default values, including 5 epochs, sub-word lengths of 3-6, learning rate of 0.025, and the bucket size of 2,000,000 which specifies the number of buckets used by the hashing function of the model to save memory.

Appendix D

Helper methods were created to iterate through the corpus of reviews and aggregate individual words' vectors and then ultimately average them together to create the document vector. The code snippets below show the process.

```
def get_word_embed(word):
    try:
        return model.wv[word]
    except KeyError:
        return np.zeros(model.vector_size)

def doc_embed(text):
    tokens = text.split()
    embeddings = [get_word_embed(token) for token in tokens]
    embeddings = [emb for emb in embeddings if np.any(emb)]
    if embeddings:
        return np.mean(embeddings, axis=0)
    return np.zeros(model.vector_size)

documents = [doc for doc in reviews['reviewText_clean']]
doc_embeddings = [doc_embed(doc) for doc in documents]
embedding_array = np.array(doc_embeddings)
```

Once the cluster labels were generated, a new column was appended to the reviews dataframe which contained the reviews corresponding cluster label.

Appendix E

To analyze the cluster results and compare them to their average VADER compound scores, first the compound scores were added as a new column in the reviews dataframe. The reviews then underwent another phase of preprocessing to remove stopwords prior to calculating the TF-IDF since these features aren't substantial for the purposes of keyword analysis. Each group of reviews was first checked for reviews that weren't empty after removing stopwords and then fit to a TF-IDF vectorization and added to a new dataframe containing the cluster number, keywords, and the VADER compound scores which could then be sorted by compound score to analyze the lowest and highest scored clusters. The code snippet below shows this process.

```
stop_words = list(text.ENGLISH_STOP_WORDS)

tfidf = TfidfVectorizer(stop_words=stop_words)

results = []

grouped = reviews.groupby('cluster')

for cluster, data in grouped:
    non_stopwords_docs = [doc for doc in data['reviewText_clean'] if any(word not
in stop_words for word in doc.split())]

    if non_stopwords_docs:
        tfidf_matrix = tfidf.fit_transform(non_stopwords_docs)

        feature_names = tfidf.get_feature_names_out()

        scores = tfidf_matrix.sum(axis=0).A1
        top_keyword_indices = scores.argsort()[-10:][::-1]

        top_keywords = [feature_names[i] for i in top_keyword_indices]

        avg_numerical_value = data['vader_compound'].mean()

        results.append({
            'cluster': cluster,
            'keywords': ', '.join(top_keywords),
            'vader_compound': avg_numerical_value
        })

results_df = pd.DataFrame(results)
```

References

[1] Ni, J., Li, J., & McAuley, J. (2019). Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. *Empirical Methods in Natural Language Processing (EMNLP)*.

[2] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014

[3] Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146. https://doi.org/10.1162/tacl_a_00051

[4] Rehurek, R., & Sojka, P. (2011). Gensim–python framework for vector space modelling. NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic, 3(2).

[5] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv [Cs.CL]. Retrieved from http://arxiv.org/abs/1301.3781

[6] Campello, R. J., Moulavi, D., & Sander, J. (2013). Density-based clustering based on hierarchical density estimates. *Advances in Knowledge Discovery and Data Mining*, 160–172. https://doi.org/10.1007/978-3-642-37456-2_14

[7] L. McInnes, J. Healy, S. Astels, hdbscan: Hierarchical density based clustering In: Journal of Open Source Software, The Open Journal, volume 2, number 11. 2017

[8] Moulavi, D., Jaskowiak, P. A., Campello, R. J., Zimek, A., & Sander, J. (2014). Density-based clustering validation. *Proceedings of the 2014 SIAM International Conference on Data Mining*. https://doi.org/10.1137/1.9781611973440.96

[9] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.