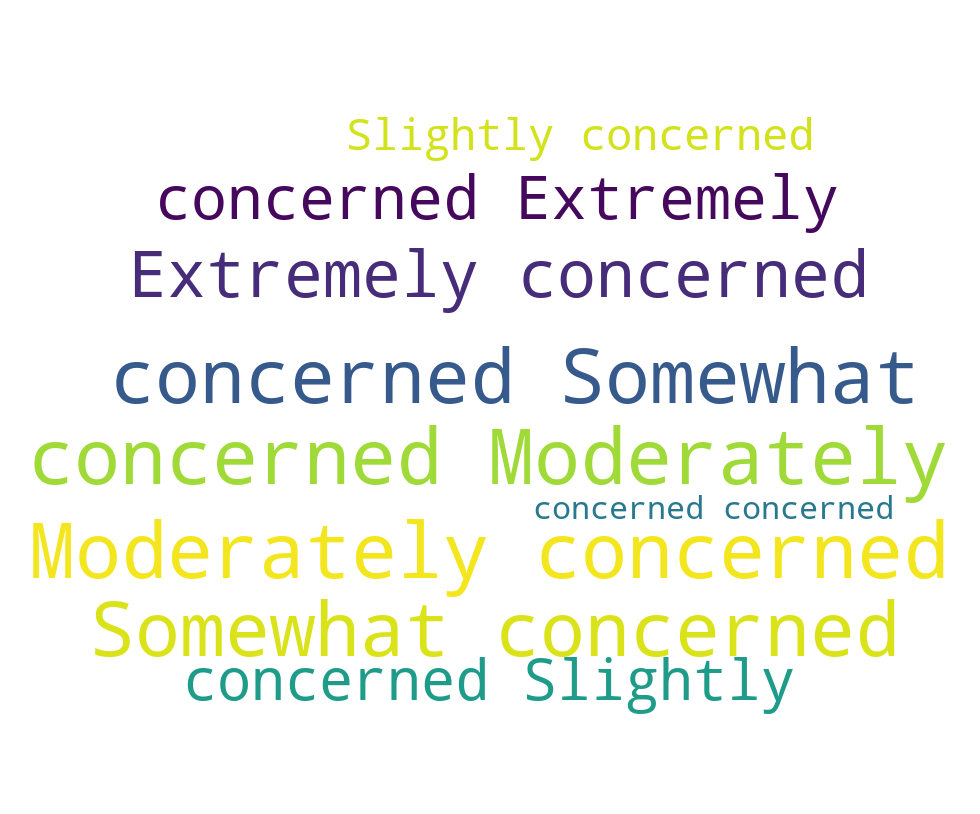
2. Statement of goals

3. Initial Data Exploration

The bar\_soup data is mainly used for testing our keyphrase extraction and summarization result. The data contains 457 surveys answers of five questions: “What is healthy skin?”, “How do you know your skin is healthy?”, “How do you get healthy skin?”, “How do you maintain healthy skin”, “How concerned”. What first comes to mind when doing keyphrase extraction and summarization is that the more frequent a word occurs, the more important it is, therefore, for the initial data exploration on this dataset, I combined the answers to each question and make a word cloud for each answer set. Word cloud is a way to display the words based on word frequency. The bigger the word looks, the more frequent it occurs in the dataset. Since all the question is asking about healthy skin, therefore, “healthy” and “skin” themselves will occur a lot in the answers but doesn’t give us any useful information, therefore, I count them as stop words when generating the word cloud. And also, I only choose the first 200 words in each answer set that occurs most frequently to get a more readable graph. Below are the five word clouds based on the answers to the five question.



*figure 3.1 What is healthy skin?* *figure 3.2 How concerned?*



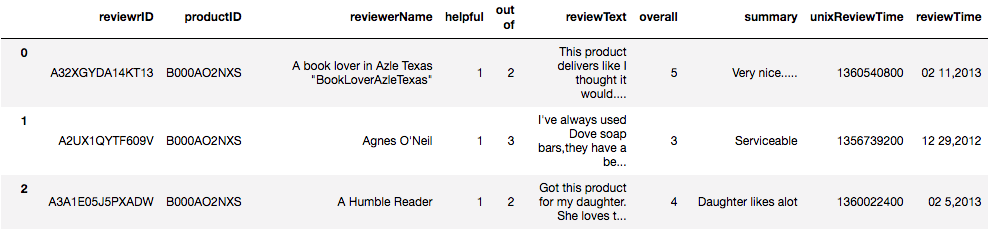
*figure 3.3 How do you get healthy skin? figure 3.4. How do you know your skin is healthy?*



*figure 3.5. How do you maintain healthy skin?*

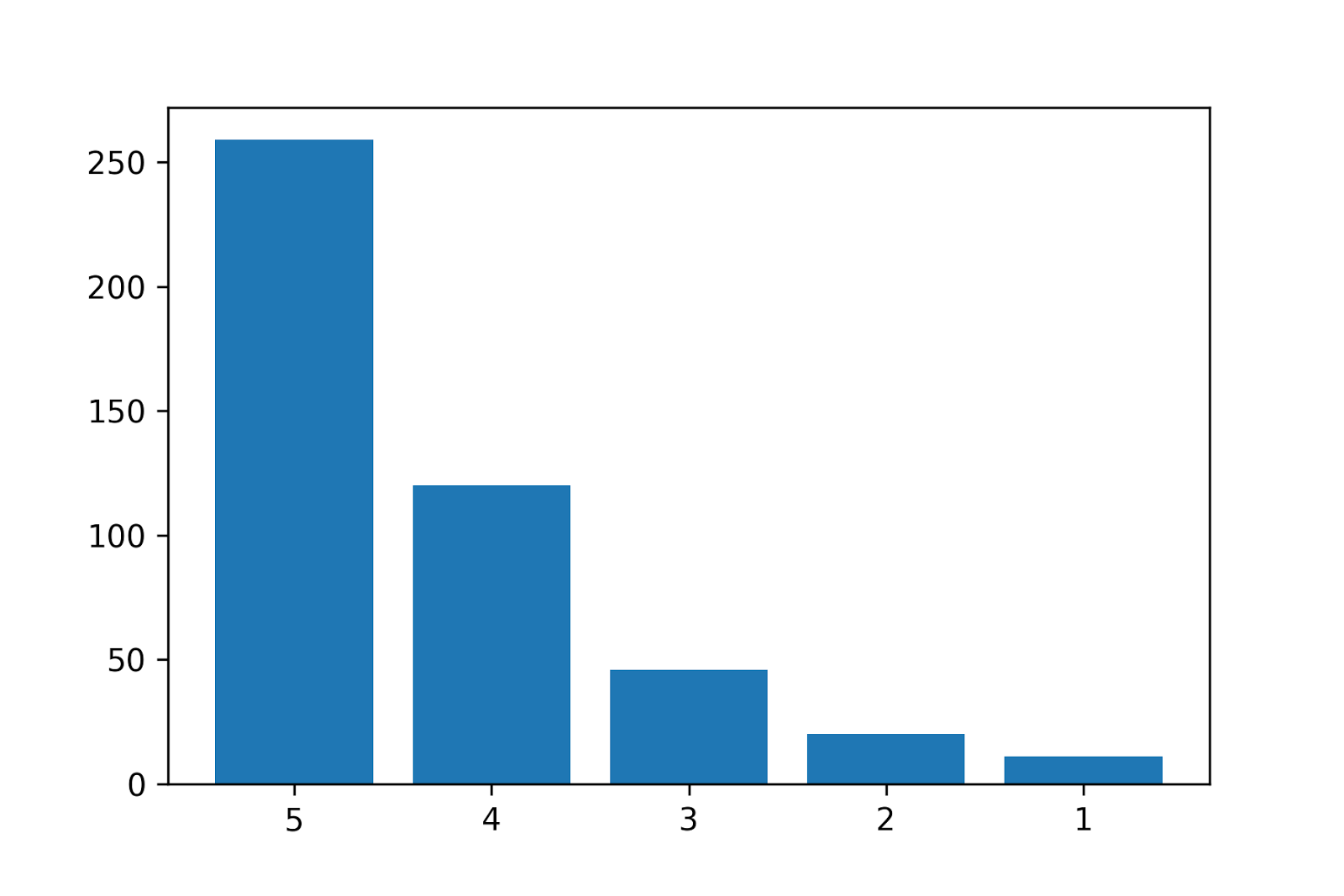
From the word clouds, we can see that using word frequency does give us reasonable result but not always. For example, in the question “what is healthy skin”, “smooth”, “soft” and “clear” stands out to be the most frequent words which make sense. But “dry” also appears as the most frequent appeared word. This is because many answers will mention “not dry” or “no dry”, while “not” and “no” are removed as stopwords, “dry” remains as most frequent appeared word. In the question “how do you maintain healthy skin?”, we can construct some most frequent answers from the word cloud, “use lotion product daily”, ”drink water”, ”eating right” that gives a good summarization to the answers. However, we can also see, there are a lot of repetitions, for example, “moisturizing” and “moisturizer”, “drink water” and “water”. Thus, from the initial data exploration on the bar\_soap data, we can see that purely rely on word frequency when doing keyphrase extraction and summarization is not enough. We also need to take care of negation, repetition and word form to get a more accurate result.

For the sentimental analysis part, the main dataset we are using is the amazon beauty review data. Below is an image presents a snippet for the data.

****

*figure 3.6 snippet of amazon\_review\_data*

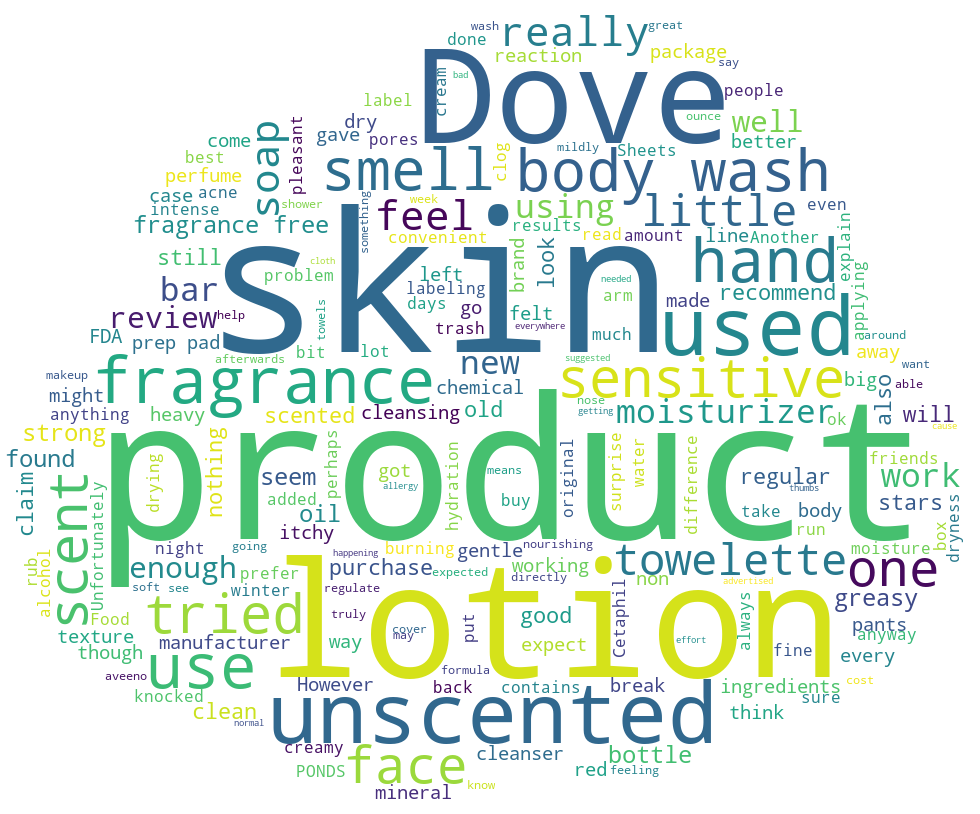
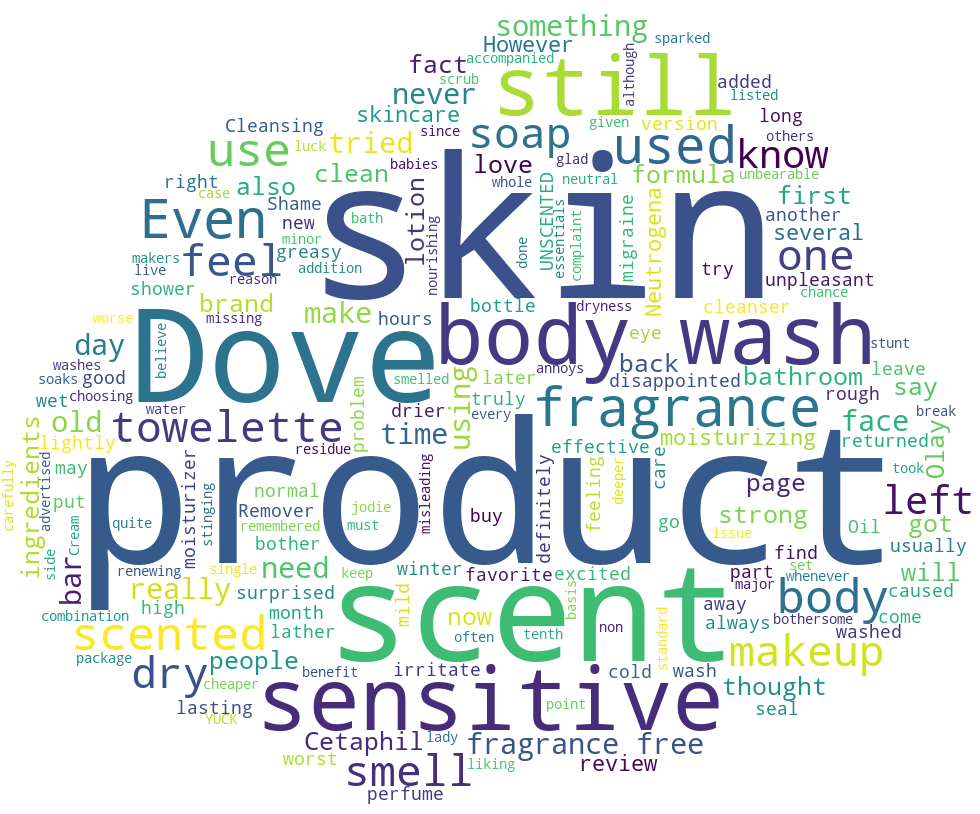
There are 10 columns in this dataset, “reviewrID”, “productID”, “reviewerName”, “helpful”, “out of”, “reviewText”, “overall”, “summary”, “unixReview”, “reviewTime”. The columns that we will use mostly for sentimental analysis are reviewText and overall. The ratings are divided into 5 levels. If the rating is 1 or 2, we can infer that the review text has a negative sentiment, if the rating is 3, then we can consider the review text to be neutral and if the rating is 4 or 5, the review text is likely to have a positive sentiment. Below is a bar chat I made from R that shows how many reviews in each rating level.



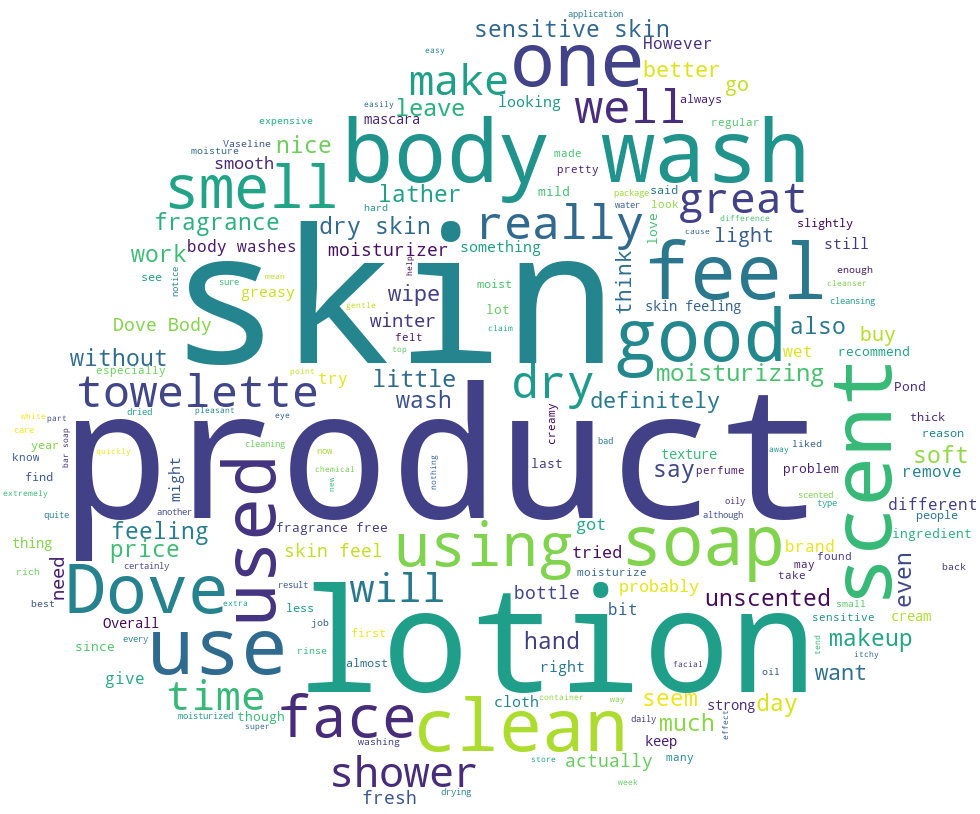
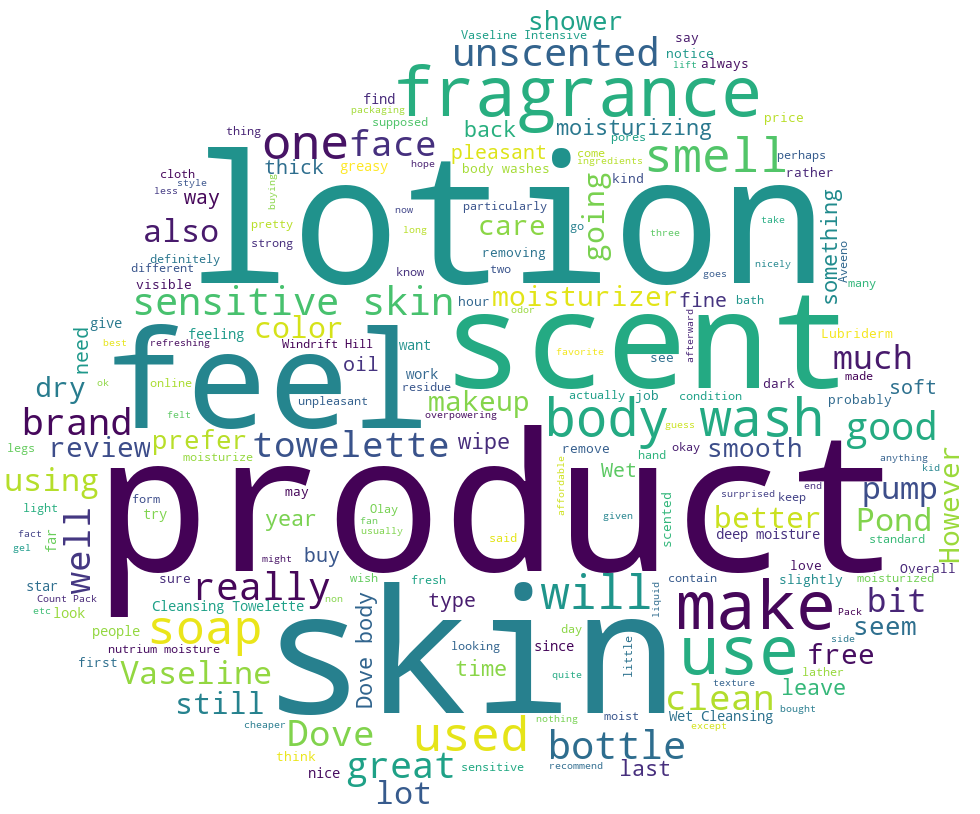
*figure 3.7. Overall rating distribution*

From the graph, we can see that in this dataset, most people gives 4 or 5 star ratings, less people give 1 or 2 star ratings, which indicates that there are more positive sentiment review text than negative sentiment review text.

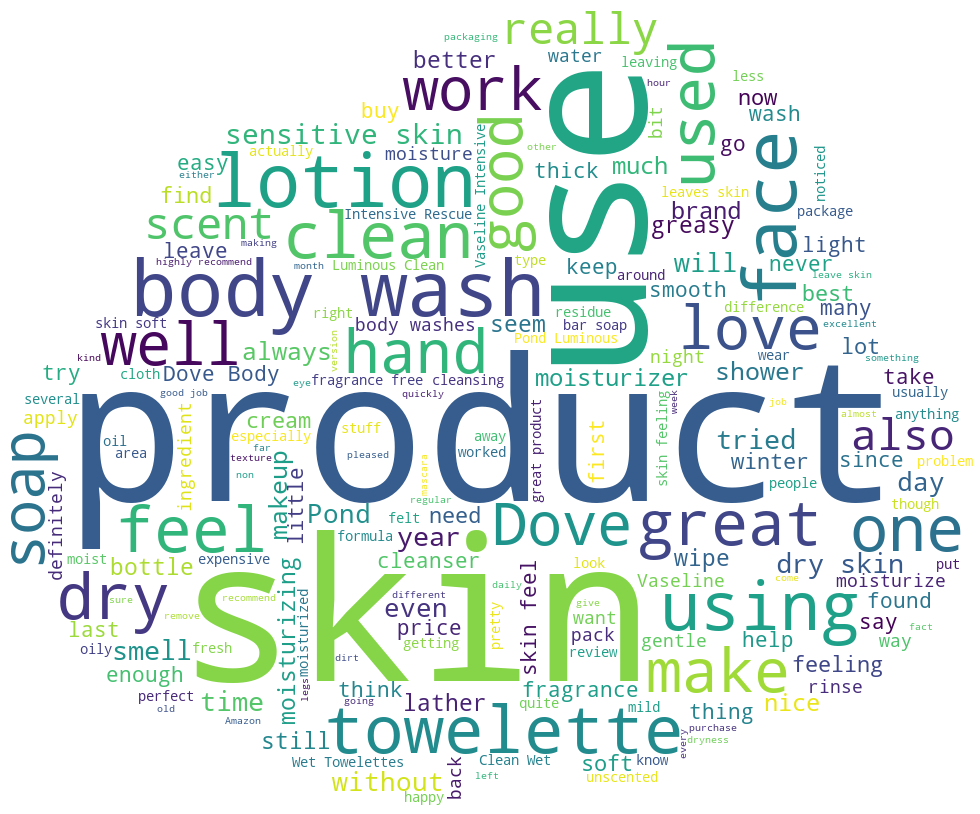
In this part of data exploration, I grouped the review text based on the overall rating and plot a word cloud for each rating level. Hopefully, the word cloud can show us what words frequently occurs in positive reviews and what words frequently occurs in negative reviews. Below are five word clouds that are generated from reviews of five rating groups.



*figure 3.8 Review text with rating 1 figure 3.9 Review text with rating 2*



*figure 3.10 Review text with rating 3 figure 3.11 Review text with rating 4*



*figure 3.12 Review text with rating 5*

From the word cloud, we can see that regardless of what rating it is, the review texts have common most frequent appear words such as product, skin, use, body wash and so on. This indicates that these words are not a good candidate that represents the sentence’s sentiment. Therefore, when analyzing the review text sentiment, we shouldn’t take these words into consideration and focus on the rest of the words to represent the sentence’s sentiment.

From the initial data exploration, we also found that the the data size if quite small for learning a supervised classification model, therefore, we have written a crawler that fetches all the reviews for the specified product from Amazon.com. We have collected 5,000 reviews so far for 10 products and collecting more reviews with time, which we will use later to improve our project.

4. Literature Review

4.1 Keyword Extraction

In the field of keyphrase extraction, the most widely used and well-known methods are using tf-idf, which stands for term frequency-inverse document frequency, TextRank and Rake, which stands for rapid automatic keyword extraction. The first one tf-idf was introduced the earliest in 1983 by Salton, G and McGill, M.J. in their book *Introduction to Modern Information Retrival*. Tf-idf is divided into two terms. TF stands for term frequency which gives a measure of how frequent a term occurs in a documents. Often, we will use a normalization to account for the fact that word will be more likely to occur in a longer document than a shorter document. The formula is given by . IDF stands for inverse document frequency which gives a measure of the importance of a word, for example, we want to weighted down stop words such as a, the, is. The formula is given by . However, the assumption that tf-idf used which is the smaller document frequency a word has, the more important it is in a document is not entirely correct, therefore, it put limitations on the accuracy of tf-idf. Also, the algorithm didn’t take the position of a word into account. For our purpose, tf-idf requires inputting a lot of documents to train in order to get keywords from one document, therefore, it wouldn’t be a good strategy for us.

TextRank and Rake is more suitable for individual document keyphrase extraction. The TextRank algorithm was brought up by Rada Mihalcea and Paul Tarau in 2004 in their paper *TextRank: Bringing Order into Texts.* The idea of this algorithm was inspired by Google’s PageRank and build texts into graph to be able to use graph-based ranking algorithms in texts. The vertices in the graph is text units and the edges are the relations between graphs. To reduce the size of graph, the algorithm will do a preprocessing step, using part of speech tags to construct a syntactic filters and only add lexical units that pass the syntactic filter to the graph. Then we iterate the graph-based ranking algorithm until convergence to get the score of vertices. Finally, we can sort words by their scores to get the most important words in the document. TextRank algorithm not only consider the frequency of a word but also the relationships between words, however, one limitation is converting text to graphs and using graph-based ranking algorithm is very time and space consuming. It also requires additional steps to combine keywords to key phrases. Therefore, we introduce the third algorithm Rake.

The idea of Rake is very simple. According to the authors, Michael W. Berry and Jacob Kogan in their book, Text Mining: Applications and Theory, “it’s based on our observation that keywrds frequently contain multiple words but rarely contain standard punctuation or stop words.” Given a document, the algorithm will first separate the document into sentences and then using a stopword list and word delimiters to separate sentences into candidate keywords. The score of a candidate keyword is the sum of the word score of its member word. Each member word score is calculated by the equation. , where word degree gives more weight to words that occurs often in longer candidates and word. After the scoring process, we select candidate keywords that have high score as keyphrases in the document. According to the evalution describes in the book, Rake was able to extract keywords 6 times faster than TextRank with the approximately the same precision. Rake algorithm completely depends on the sentence structure to extract keywords, therefore, often it neglects the meaning of a word. For example, it won’t take negation or synonyms into account, also the word scoring method is not optimized. Therefore, in the next step, my approach is to optimize the Rake algorithm to handle negation, synonyms and providing more flexible word scoring to achieve a better result in keyword extraction.

4.2 Summarization

Text summarization methods can be classified into extractive and abstractive summarization method. An extractive summarization method involves collecting and selecting important sentences from the original document to generate summaries into shorter form. The sentence is extracted based on statistical and linguistic features. For example, using technique to determine the most important key phrases, or directly choose the most important sentence. The advantage of extractive summarization is that it’s not necessary to have a huge dataset, and the calculation is simple and straightforward. The most famous method to generation extraction summarization is called Latent Semantic Analysis(LSA). LSA is a method based on statistical calculations to extract and represent the contextual meaning of words and the similarity of sentences. It is an unsupervised method of deriving vector space semantic representation from a large corpus of data, which doesn't need any training or external knowledge. LSA uses context of input document and extracts information such as  which words are used together. And which common words are seen in different sentences. We can conclude that if the number of common words between sentences is high, it means that the sentences are more semantically related. LSA method is based on SVD method, which stands for singular value decomposition. LSA mainly include three steps:

i. The creation of input matrix: the text (input document) is represented as a matrix. Each row represents the word and each column represents the sentence. The cell value represents the importance of the word. There are many different approaches to fill the cell values such as the frequency of the words in sentences.

ii. Singular Value Decomposition (SVD): singular value decomposition is a mathematical method applied to the input matrix. SVD is used to identify patterns in the relationships between the terms and sentences. SVD as a mathematical equation can be represented as an m×n matrix (M). M is formed as M = U Σ VT,  where U is an m×n matrix which represents the original rows as vectors of extracted values , Σ is an n×n rectangular diagonal matrix with nonnegative real numbers on the diagonal representing the scaling values, and VT (the conjugate transpose of V) is an n×n real or complex unitary matrix which represents the original columns as vectors of extracted values .

iii. Sentence Selection: after applying the SVD, its result is used to select the sentences to generate the summary. There are many methods and algorithms to select the sentences. [1]

**LSA has many properties that make it widely applicable to many problems as follows:**

1. LSA is a global algorithm that has the ability to collect all trends and patterns from all documents and all words.

2. LSA provides the ability to retrieve documents based on words and vice versa. Where LSA is used to map the documents and words to the same concept space.

3. The concept space contains fewer dimensions where these dimensions contain the most information and least noise. [1]

**LSA has several limitations that must be considered when deciding whether to use LSA.**

1. LSA is difficult to handle the polysemy. Polysemy means that the words with multiple meanings depending on the context. In other words, the same word with different meanings has the same concept and this will cause a big problem.

2. LSA depends on SVD. SVD has some disadvantages which are (1) SVD is time consuming and (2) when new documents are added, their calculations are very hard to be performed. Since SVD is a very complex algorithm, the performance is decreased. [1]

Based on the benchmark work from last semester, the LSA method got a very reasonable result. Through manually checking the result, most review summarizations contain function of the product, also the representative attitude from most customers. However, the only drawback of this method is that we can only choose several whole sentences from original text, and some of the sentences are pretty similar, which is actually redundant and not readable. From the perspective of improvement, my next step is to reduce similar keyphrase and sentence in the summarization.

References:

[1] Rasha Mohammed Badry and Ahmed Sharaf Eldin and Doaa Saad Elzanfally, 2013. “Text Summarization within the Latent Semantic Analysis Framework: Comparative Study”

[2] Josef Steinberger and Karel Ježek, “Using Latent Semantic Analysis in Text Summarization and Summary Evaluation”

[3] Jen-Yuan Yeh and Hao-Ren Ke and Wei-Pang Yang and I-Heng Meng, 2014. “Text summarization using a trainable summarizer and latent semantic analysis”

[4] Mihalcea, Rada, and Paul Tarau, “TextRank: Bringing Order into Texts.” 2004.

[5] Berry, Michael J. A., and Jacob Kogan. “Automatic Keyword Extraction from Individual Documents.” *Text Mining: Applications and Theory*, Wiley, 2010.

[6] Gerard Salton and M. J. McGill. “Introduction to Modern Information Retrieval.” McGraw Hill Book Co., New York, 1983.