UNIL | Université de Lausanne

HEC Lausanne

# Text Mining Project

*Analysis of sunscreen reviews using text mining methods*

**Text Mining**

**Fall 2019**

**Group D**

**HEC Lausanne**

**Submitted on 18.12.2019**

## DECLARATION OF AUTHORSHIP

This project was written by us and in our own words, except for quotations from published and unpublished sources, which are clearly indicated and acknowledged as such. We are conscious that the incorporation of material from other works or a paraphrase of such material without acknowledgement will be treated as plagiarism, subject to the custom and usage of the subject, according to the University Regulations. The source of any picture, map or other illustration is also indicated, as is the source, published or unpublished, of any material not resulting from our own research.

Rita Sefraoui Tahiri (10812097)

[illegible]

Charlotte Fleury (14608749)

[illegible]

Astrid Wiedmer (13413141)

[illegible]

Alexandre Schroeter (15400716)

[illegible]

## ABSTRACT

This project aims at studying sunscreen reviews using text mining methods. With that purpose, we analyze separately two corpuses of texts, extracted from two different online platforms, Reddit and MakeupAlley (MUA).

After data extraction, the first step is to perform pre-processing computations, namely lemmatization, stemming, stop words exclusion and tokenization. These apply on both datasets whereas the following analyses will be distinct depending on the dataset. We also create a specific sentiment dictionary relevant in the skincare field for sentiment analysis.

For the Reddit dataset, we first perform a frequency analysis using the TF-IDF. The main insight is that there is no common word to all the reviews. With the sentiment analysis, we find out that trust and anticipation are the two most present sentiments. Through a similarity analysis, we first compute a Latent Semantic Analysis (LSA). We can clearly identify two extreme outliers corresponding to posts generating the most traffic. Otherwise, we see that most of the texts are gathered around the same area: slightly negatively correlated to dimension 1 and equally spread around 0 for dimension 2. We sort out as well 2 extreme outliers (sun and cancer) that are both negatively correlated to both dimensions. Using a Latent Dirichlet Allocation (LDA), we can analyze that some words are strongly associated to some topics while it seems that each topic is equally associated to certain number of documents. We perform a few complementary analyses whose insights are the following: the majority of the posts generate little interaction with a median of 2 comments and people enquiry themselves the most about sunscreen right before summertime or at its early edges (May and July). We also perform a brand analysis using some of the brands from the MUA dataset, and the main element is that the brand La Roche-Posay is the one appearing in the highest number of reviews on Reddit.

When computing analyses on the MUA dataset, we can group our analysis per brand, per product or per review. From the sentiment analysis, we extract that the polarity score is more differently spread between the brands but evenly distributed across products. Per brand, a sense of anticipation is evoked. When performing a similarity analysis, we cannot interpret the cosine similarity. For the LDA, the main insight is that the word "face" is associated to almost every subject except 2 out of 10. Otherwise, we do not see any strong semantic signification about these topics. We finally perform supervised learning methods to see if we manage to predict to which brand pertains a certain review. We use the features present in the original dataset as well as the ones we extracted from the LDA, from a GloVe analysis and from the polarity of the sentiment analysis. We also create a dummy that identifies if the brand is cited in the review. We test two classification methods: a Random Forest and a Neural Network (lower accuracy). We do a small grid search to evaluate the optimal number of trees of the Random Forest. We reach better results with balanced accuracy but need to put our results into context because the skin care field is very specific.

TABLE OF CONTENTS

## 1. INTRODUCTION

As part of a course of Text Mining, a project on sunscreens is addressed under the supervision of Dr. Marc-Olivier Boldi. This project aims at performing a study on the sunscreen market, given the analysis of an original corpus of texts made of several reviews.

Therefore, this paper aims at better understanding the consumers' perception of different brands by using text mining methods. We will analyze separately two corpuses of text, extracted from two different online platforms, Reddit and MakeupAlley.

First, a description of the data obtained to create our corpuses is provided in section 2. The description focuses on a set of reviews extracted from both sources. We detail the data cleaning and pre-processing work we perform to obtain our corpuses as well as the use of specific packages. Second, the methods applied to analyze our data are explained in section 3. The different tools used to construct relevant dictionaries are explained thoroughly (section 3.1.1). The methods used to perform unsupervised learning analysis (section 3.2) and supervised analysis (section 3.3) are detailed as well. Finally, the results are described in section 4. We first explore the results of the Reddit corpus (section 4.1) before the ones from the MakeupAlley corpus (section 4.2).

## 2. DATA DESCRIPTION

### REDDIT dataset

The reddit dataset is composed of reviews extracted from the website Reddit.com when performing a tailored research on sunscreen among the Skincare section. At this stage, the extraction produces a data frame with a flat structure, there is no order or hierarchy of individual comments. We extract nine different threads whose composition regroup the same 18 features in data frames named respectively d1, d2, d3, d4, d5, d6, d7, d8 and d9. d1 is composed of 85 observations (comments), d2 of 459, d3 of 316, d4 of 367, d5 of 8, d6 of 115, d7 of 36, d8 of 153 and d9 of 10 observations. The features are displayed below:

**Id:** is the identification number of the different comments of a thread

**Structure**: labels the structure of the comment section if one post is responding to another specific one

**Post_date**: gives the date when the initial post is posted

**Comm_date**: gives the date when a comment is posted on that specific post

**Num_comments**: indicates the number of comments of the specific post.

**Subreddit**: links to the reddit section affiliated; here to SkincareAddiction

**Upvote_prop**: gives a uniform proportionality per dataframe

**Post_score**: labels a unique score per thread

**Author**: gives the name or online pseudo of the author of the 1st post/thread

**User:** gives the name or online pseudo of the person commenting the post

**Comment_score:** provides the number of interactions following a post

**Controversiality:** is always ranked 0 for every dataframe

**Comment:** gives the actual content of the post/comment

**Title:** provides the name of the initial post

**Post_text**: gives a uniformed text for every row corresponding to the initial post, except for d1, d3 and d4 which are empty

**Link:** provides the http link to access solely to the image affiliated to a post to illustrate it

**Domain:** corresponds to website's domain

**URL:** gives the URL link to access to the post on Reddit

### MAKEUPALLEY dataset

Our dataset is made of 1062 observations with 24 variables. This dataset contains 12 different products of seven different brands. We describe below the content of the 24 columns:

**totalResults:** number of reviews for a given product identifier, *numeric*

**brandId:** identifier of the brand, *factor*

**brandName:** name of the brand, *factor*

**categoryId:** category identifier (sunscreen has categoryId equal to 22), *factor*

**categoryName:** name of the category; e.g sunscreen, *factor*

**helpful:** number of people who found a given review helpful, *numeric*

**productid:** unique product identifier, *factor*

**productName:** name of the product, *factor*

**rating:** rating of the product, from 1 to 5 (unit increment), *numeric*

**review:** review of the product, *character*

**reviewDate:** date when the review was made, *date*

**reviewId:** unique review identifier, *factor*

**thumbnail:** link of the thumbnail of the product

**username:** name of the user, *factor*

**votes:** number of votes of a given review, *numeric*

**skinType:** skin type of the reviewer, *factor*

**skinTone:** skin tone of the review, *factor*

**skinUndertone:** skin undertone of the reviewer, *factor*

**hairColor:** hair color of the reviewer, *factor*

**hairType:** hair type of the reviewer, *factor*

**hairTexture:** hair texture of the reviewer, *factor*

**eyeColor:** eye color of the reviewer, *factor*

**ageRange:** range age of the reviewer with levels (“Under 18”, “19-24”, “25-29”,”30- 35”, “36-43”, “44-55”, “56 and over”) , *factor*

**reviews:** number of reviews made by a given reviewer, *numeric*.

## 3. METHODS

In this section, we explain how the study is conducted, which tools are used and why. To view the reproducible code, you can go to the appendix, but to have a well commented well formatted code, please refer to the public Github repository: “rsefraou/textminingProject.git ”

### Data extraction

We decide to use two different datasets to run our project: Reddit and MakeUpAlley (MUA). For our first dataset, we use reviews extracted from the Reddit website. Reddit is an online bulletin board and a social networking website where registered users can submit and discuss content. We use a specific R package to extract the reviews: RedditExtractoR which creates a graph file from a single Reddit thread. This package uses Reddit API to retrieve comments together with all corresponding attributes from Reddit threads.

We then extract sunscreen reviews from MakeupAlley.com website and select the category of product we are interested in, namely sunscreen. We filter for the products with 50 or more reviews and choose 12 products with different grades, from seven different brands. The brands are chosen to be both relevant in the US and the EU.

We select the item number on the website. For the webscrapping, the most convenient format for the extraction is the JSON format. We create a function, get_last_page(), to receive the number of pages of reviews for a given item/product. In the meantime, we define a second function, get_data_from_json(). This function takes a given JSON object and transforms it into a data frame object. Finally, we create a third function scrape_write_table() which calls the two aforementioned functions. The later function concatenates an URL common to all items on the website with the specific identifier of a given item. It then extracts the total number of pages of reviews for this item. Following this, it generates the target URLs for all the pages of a given product. The number of URLs is equal to the number of pages. Subsequently, the function calls the function get_data_from_json() which takes a given URL, extract the associated JSON object and turns it into a data frame object. After this, scrape_write_table() binds all the data frames of a given object together, modifies the headers and save the resulting object into a .csv format. We apply scrape_write_table() on a vector made of all our item identifiers. Finally, we load all our .csv files into one object.

## Construction of a relevant dictionary

We build a custom dictionary for sentiment analysis. We want it to include the notions of comfort and discomfort, as well as some connotations which can be relevant in the skincare field. For example, “oily” is neutral in the “nrc_sentiments” dictionary, but we want it to reflect disgust as it is always negatively connoted in the reviews. A positive word for oily would by balmy, creamy, moisturizing or rich...etc. To create our dictionary, we look at lexical fields on google on the topics of skin, skincare and sunscreen. We put a grade of 0 or 1 on ten sentiments: fear, disgust, anticipation, sadness, joy, trust, comfort, discomfort, anger and surprise to the words we extracted from the lexical fields search. We then modify the existing “nrc_dictionnary” to include our two new sentiments. Finally, we join the nrc dictionary with our two new columns of sentiments. When there is twice the same word, we keep ours. We transform this table with a loop to eliminate the 0 and 1 encodings. Indeed, we want the word to be in a column *word* and the sentiment associated to it in a column *sentiment*, both in the same row. We modify the function get_sentiments() from textdata to get_sunsentiments().

## Data preprocessing

**MakeupAlley:** We use the package “textstem” to lemmatize our comments. This package uses the dictionary “hash_lemmas” to recognize the words of our text and brings them to their lemma. We then proceed to transform our comments (*review*), our brands (*brandName*) and our products (*productName*) into all lowercase words. We then create a vector of stop words containing the pre-existing common terms of the “quanteda” dictionary for the English language, as well as the words “sunscreen”. We finally tokenize our reviews with the “tidytext” function “unnest_tokens()” at the word level. We remove the stop words because they are not likely to highlight any brand/product/review specificity.

**Reddit:** We use the package “tm” to stem and lemmatize the comments. For the tokenization of the comments, we use the same method as for the MUA database.

## TF-IDF and Document-Term-Matrix

**MakeupAlley:** We compute the TF_IDF of words per review and per product to see if some words seem to characterize the reviews or the given devices. To compute these calculation we use the “tidytext” package to apply the bind_tf_idf() function on the TF database we build with the “dplyr” library. To visualize the TF-IDF we print a Kable table and order it by TF-IDF. We want to look at the highest ones as they are linked to the words that represent the best our texts (reviews or products).

**Reddit :** We use the package “tm” with the function “DocumentTermMatrix()” to produce the document-term-matrix of our Reddit database. This matrix is used to manually compute the TF-IDF of the words.

## Sentiment analysis

**MakeupAlley**

**i) Valence shifters**

We use the package “sentimentr” and its dictionary “hash_valence_shifters” to compute the sentiment per review given the context of the its polarized words. We plot a boxplot of the distributions of sentiments per brands to see if some brands have more consistent sentiments or more recognizable patterns. We do the same by product.

**ii) Without valence shifters**

We look at the occurrence of words which we associate with given sentiments using the function “get_sunsentiments()”. This sentiment analysis is made by brand and by product to see if we see patterns unique to certain brands or products. We make the sentiments appear as percentages to account for different number of words per product/brand.

**Reddit:** We look at the occurrence of words which we associate with given sentiments using the function “get_sunsentiments()”.

## Similarity analysis

**MakeupAlley**

**i) Cosine similarity**

We decide to use the cosine similarity as a measure of similarity instead of the Jaccard coefficient to account for differences in length of documents. To compute the cosine similarity, we tranform our corpus into a “dfm” format from “quanteda”, from which we remove our stop words. We then compute manually the cosine similarity between texts and plot the texts as a dendrogram.

**ii) Latent Dirichlet Allocation (LDA)**

We use the library “topicmodels” to perform a Latent Dirlichet Allocation on our MUA corpus. We use the function LDA(), pass our document-term-matrix as argument and set the desired number of topics we want to see appearing. We consider ten topics. Using the ggplot package we represent the beta parameters (topics-to-term probabilities) and the gamma parameters (the topics-per-document probabilities). We extract the gamma and beta parameters and represent the most frequent words per topic as well as the documents the most associated with each topic.

**Reddit**

i) **Word frequency:** we plot the most frequent words in each review.

ii) **Latent Semantic Analysis (LSA)**

We use the “quanteda” package to compute the LSA of our corpus. After removing the stop words and putting the corpus in a “dfm” format, we use the function “textmodel_lsa()” where we choose 5 subjects. We represent the comments on dimensions 1 and 2 using the “ggplot” package and we do the same with the terms. We then look at the most frequent terms per topic and the documents associated with the latter.

iii) **LDA:** we use the same method as for the MUA database.

### Word Embedding: GloVe:

**MakeupAlley:** We create a co-occurrence matrix of our corpus with a window of size five to apply the GloVe model on our data with the help of the “text2vec” library. We implement this method to visualize the principal component analysis and distance between words and documents. This method is further discussed in the following section.

### Supervised learning

**MakeupAlley:** We use machine learning techniques to try to predict the brand. We have features from the webscrapping process (e.g. number of votes for a given review, skin type of the user, skin tone of the user,…) but we also create extra features resulting from the text mining methods we describe above. We use word-embedding (GloVe model) with 50 dimensions to represent each of the reviews. Each dimension starts with the letter V (*V1, V2,…*). We extract the beta and gamma matrices resulting from the Latent Dirichlet Allocation, multiply them and get a matrix with 1062 rows and 10 columns which we call *mlda*. Each dimension starts with the letter G. Finally, we use the result of the sentiment analysis we performed before, which is the difference between the number of positive words and the number of negative words, *sentimenttext*. This gives us a numeric feature. Furthermore, we add nine binary features, which simply indicates whether a sunscreen brand was mentioned in the review or not. This will help our classification task. Finally, we compute the number of words a given review has. We join our original sunscreen dataframe with the GloVe components, the matrix resulting from the beta/gamma multiplication, the dataframe of the sentiment analysis and the 9-dummy matrix by using the *reviewId* and text number, *text*. This dataframe is called *sunscreen.sl.brand.*

We use two advanced methods; random forests with the packages randomForest and caret and a neural network with the package Keras to perform our multi-class classification task. We split the *sunscreen.sl.brand* into a training and testing set, using a proportion of 75 percent. We preprocess our 72 features by normalizing them, as to weigh all features equally. We then perform a small grid search for the number of trees using: 100, 500, 1000, 1500 and 2000 trees and create the confusion matrix for each of the random forests. We cross validate our results by using 10-fold cross-validation, repeated five times and use the accuracy as metric.

For the keras neural network, we use a feedforward neural network with one input layer with 72 nodes, three hidden layers of 40, 20 and 15 notes respectively and an output layer with seven units, one for each class. Our first four layers use a ReLU activation function, while the output layer uses a softmax activation function. We use the categorical crossentropy as loss, the optimizer “RMSprop” and the accuracy as metric. To prevent overfitting, we adopt a validation set approach by allocating 15 percent of our training set to a validation set during the training of our model. Finally, we construct the confusion matrix.

### Complementary Analysis

We use the ggplot package to visualize some variables of interest of the reddit database. We create a boxplot of the comment score and we filter out negative values and outliers by setting a threshold at the value of 10. We also perform an analysis of the comment date by creating a barplot of the month of the posting date to see if there is a particular pattern. To do so we convert the variable *comm_date* into a date format and we create a new column indicating the month. We then perform a brand analysis. We search for brand names in our dataset; the same ones that are analysed in the MakeupAlley database and we create a new data frame containing the observations of those brands. We then plot the number of times a brand appears in and per reviews and we perform a sentiment analysis using the created function get_sunsentiments().

## 4. RESULTS

### 4.1 Reddit results

#### 4.1.1 Frequencies

We start by looking at the terms’ frequency of our document tokens. We sort out the most used terms in the reviews. As seen in Appendix 12 (Figure 12 - FREQUENCY ANALYSIS), the two most frequent terms are use and tri.

In the Appendix 13 (Figure 13 - FREQUENCY ANALYSIS PER DOCUMENT), we split our frequency analysis per document. “Good” is the token that first comes up twice in separate reviews. There is no common word to all the reviews. The few tokens that appear more than once in separate reviews are: good, thank, sun, use, aqua. For review five, we can see only four tokens highlighted, that is because the rest of the tokens share the same frequency of one. Another remark is that in the review six the token Purito is the most frequent and it is a sunscreen brand. The purpose of this analysis is not to compare the review between themselves but to have a general overview, we have to beware the differences of scale.

#### 4.1.2 Sentiment analysis

We compute the sentiment associated to the tokens. From the dictionary built, the ten sentiments associated are shown in Appendix 14 (Figure 14 - SENTIMENT ANALYSIS). Trust and anticipation are the two sentiments the most present. The opposites joy and sadness are equally split sentiment across the corpus. Then, it is anger and fear that are the first negative sentiments coming out.

### 4.1.3 Similarity analysis

We perform a LSA. We choose to compute nodes of size five, enabling to create less disperse clusters, regrouping more features (Figure 15 - LSA). We can clearly identify two extreme outliers corresponding to texts 905 and 216. Those correspond to posts generating the most traffic, due to a large number of characters and an important comment length. Otherwise, we see that most of the texts are gathered around the same area: slightly negatively correlated to dimension 1 and equally spread around 0 for dimension 2. We can see in Appendix 1 and 2 (Figure 1- LSA TEXT ASSOCIATED TO TOPIC 1, Figure 2 - LSA TERMS ASSOCITED TO TOPIC 1)the detail of the terms and documents associated to topic 1 and topic 2.

We then look at the terms associated to dimensions 1 and 2 (Figure 16 - LSA TERMS TO DIMENSIONS). We input as well the sentiment linked to each word. From a first look, we can identify two extreme outliers: sun and cancer are both negatively correlated to both dimensions. The other words sharing the trust sentiment are usually gathered around the majority of the terms. Most of the other outliers are not associated to any sentiment. In general, the relationship of the documents and the terms towards dimensions 1 and 2 is pretty similar.

Our next computation consists in performing a LDA. We first sort out the term-per-topic graph (

Figure 18 - TERM-PER-TOPIC LDA (REDDIT). We arbitrarily decide to split the corpus within ten topics. We can analyze that some words are strongly associated to some topics. For example, “white” is predominant in topic 3 and “cancer” in topic 6.  We also compute the document-per-topic. Unfortunately, the graph (Figure 3 - DOC PER TOPIC ANALYSIS) is not readable. We thus decide to select only the top ten doc-per-topic. It corresponds to the Appendix 19 (Figure 19 - TOPIC PER DOCUMENT (REDDIT)). It seems that each topic is equally associated to certain number of documents.

### 4.1.4 Complementary analysis

We decide to analyze how much traffic an initial text/comment can generate. We compute a boxplot taking into account the comment score. As we see in Appendix 4 (Figure 4 - BOXPLOT OF THE TRAFFIC GENERATED BY A POST), we have an extended number of outliers and the boxplot itself is very little. We even have some negative values corresponding to downvotes. We analyze in depth the boxplot by zooming in and obtain the boxplot (Figure 20 - BOXPLOT OF TRAFFIC FOLLOWING A POST) without outliers (comment score < 10) and with only positive values. We can conclude that the majority of the post generate little interaction with a median of 2 comments.

We then look at the quantity of comments generated per month. We want to know if it is linked to what we assume to be the peak season for the sunscreen business (summer months). Therefore, by looking at Appendix 21 (Figure 21 - QUANTITY OF COMMENTS GENERATED PER MONTH), we see that people enquiry themselves about sunscreen right before summertime or at its early edges. Indeed, between May and July, we face the most important traffic about the subject. November and December show a very little flow, and it seems to make sense since the use of sunscreen is less pronounced during those months. We remind that we only show the flow from

May to December as it corresponds to the data that we have. Likewise, the data gathered concern a European seasonal pattern.

In order to connect as much as possible our two databases, we perform a brand analysis, scoping to the main brands that will be explore latter in the MUA dataset. We consequently isolate the brands Biore, Cerave, Clinique, Neutrogena and (Roche) Posay as shown in Appendix 22 (Figure 22 - APPEARANCE OF BRANDS IN REVIEWS). The most frequent brand to appear all reviews combined is Cerave, with an iteration of 45 times.

Following up our analysis of brand appearance, we see in Appendix 23 (Figure 23 - APPEARANCE OF BRANDS PER REVIEWS) that Cerave which appears the most is mainly cited in review 4 (or document 4). Clinique which appears at a very low frequency is present only in the thread of review 2. The brand Posay is the one appearing in the highest number of reviews.

We then perform a sentiment analysis on our selected brand, to identify which sentiment is mostly linked to each brand. The sentiments affiliated are evenly spread out between the different brands as seen in Appendix 24 (Figure 24 - BRAND ANALYSIS). We have a predominance of the anticipation, trust and joy sentiments. We can conclude that overall it emerges a positive feeling towards all the brands.

## 4.2 MUA results

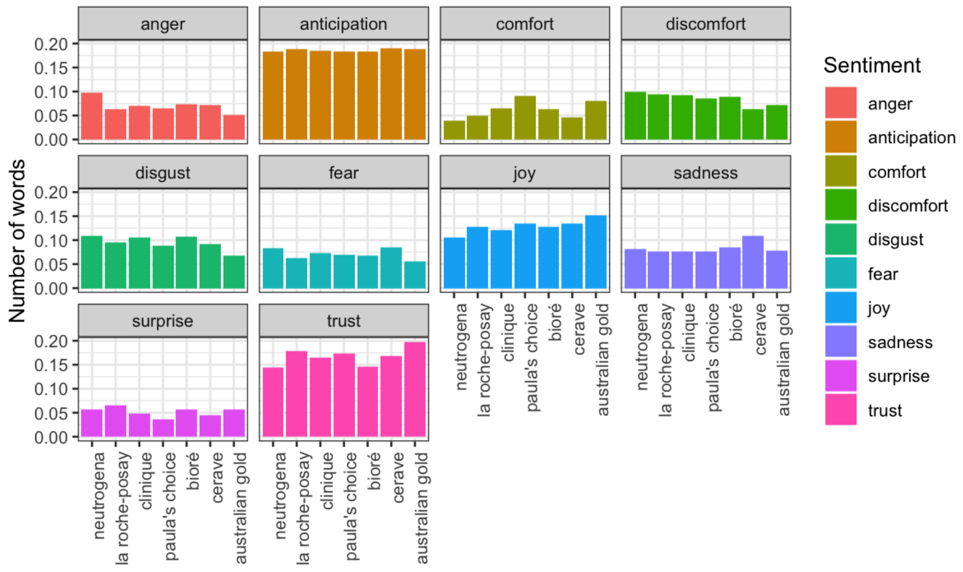
### 4.2.1 TF-IDF

Looking at the words which characterize the most the reviews (Figure 5 - TF-IDF PER REVIEW (MUA)), we see that "variation" is the word which characterizes the most the review 2255442. It has the highest TF-IDF of the all the words per review. If we do the same but per product instead of per review (Figure 6 - TF-IDF PER PRODUCT (MUA)), the tf-idfs are all lower. Here, the word "mexoryl" is the most characteristic of the Anthelios XL, spf 60+ product. Making a small research, mexoryl is a chemical spf filter that is found in this sunscreen and is patented by LaRochePosay[1] so it validates our characterization of this product.

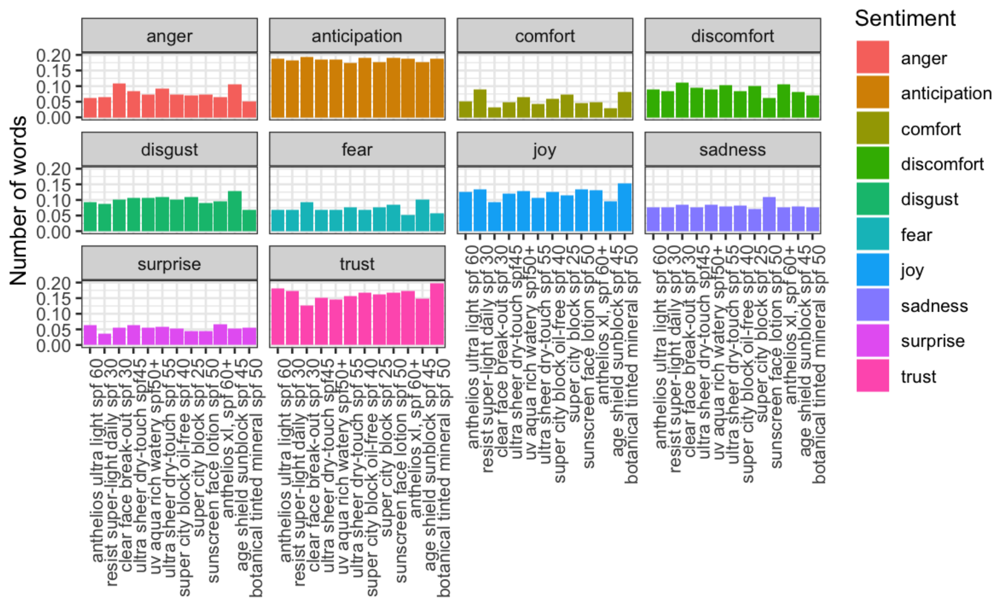
### 4.2.2 Sentiment Analysis

We first look at the sentiment score distribution per brand and per product. Using the dictionary "hash_valence_shifters", the sentiment polarity per review was computed taking into account the valence shifters of the sentences (like the presence of negators or amplificators). We represent this polarity as boxplots (Figure 7 - SENTIMENTS WITH VALENCE SHIFTERS (MUA)) per brand and per product. We see, for example, that Clinique has a strong positive outlier when Cerave has more consistent polarity across reviews. However, when we look at the products, it seems that the polarity is distributed more similarly across them and none look particularly different from one another. When looking at the sentiment occurrence per brand (Figure 9 - SENTIMENT OCCURRENCE

[1] https://www.laroche-posay.fr/article/anthelios-haute-protection-solaire-au-service-des-peaux-sensibles/a31013.aspx



PER BRAND (MUA) ), we see that all of them seem to provoke a great sense of anticipation. Paula's Choice and Australian Gold brands seem to stand out as more comfortable to wear and Neutrogena seems to be less associated with occurrence of joyful words. Australian Gold also stands out for having the most trust associated terms. Per product (Figure 8 - SENTIMENT OCCURRENCE PER PRODUCT (MUA)



) we see the resist super light daily spf 30 associated with more comfort and the sunscreen face lotion spf50 seems to have more sadness related words in its review. However, because this dictionary was made by us solely, and with no consideration of valence shifter, it is clear that while this representation is interesting, we cannot deduce from it alone a representative and true sentiment about the product. A simpler way to know what the user thought about the product would be to look at the grades attributed.

### 4.2.3 Similarities

Cosine similarity: We see (Figure 10 - COSINE DISTANCE BETWEEN TEXTS ( MUA)) that there seem to be a lot of clusters with texts that are very close together meaning that they are difficult to be clearly distinguished with this method. This makes the interpretation of the cosine similarity impossible for us.

LDA: We can observe in the Appendix 17 (Figure 17 - TOP TEN DOC-PER-TOPIC LDA (MUA)) the words most associated with the 10 topics we extracted from our MUA database. The word "quot" seems strongly associated with topic 3. The word "face" is associated with topics 1, 2, 3, 4, 5, 6, 7, and 10.

In the Appendix 11 (Figure 11 - TOP GAMMAS PER TOPIC IN LDA ANALYSIS (MUA)), we can observe, for each topic, its most strongly associated reviews (text). Here, we print only the most linked text per topic but, if we look at all the texts, we see that most texts are related to many subjects. We do not see any strong semantic signification about these topics, but we use them as inputs in our supervised learning program.

### 4.2.4 Word Embeddings: GloVe

We apply the GloVe method and look at the clustering tree (Figure 27 - GLOVE TREE CLUSTERING (MUA)). It is interesting to see the closeness of the words « niacinamide » and « missha ». They are in the same branch, close together as missha has a sunscreen having this ingredient. Otherwise, the results are difficult to interpret. The GloVe is built mainly for supervised learning features.

### 4.2.5 Supervised learning

When we tune our random forests and try to select the best model according to accuracy, we notice that the random forest with 500 trees is the one which performs the best, reaching an accuracy of 56 percent (Figure 25 - ACCURACY OF RANDOM FOREST). While it is not the one with the highest upper limit of accuracy (this is the RF with 1000 trees) nor the one with the most robust results (RF with 2000 trees), the RF with 500 trees (Figure 26 - SUMMARY OF RANDOM FOREST WITH 500 TREES) gives us the best accuracy for this run. The 95 percent confidence interval on the accuracy ranges from 0.493 to 0.620. As for the neural network, its accuracy is much lower, reaching only 38 percent. Therefore, we discard this model.

As though we acknowledge that a 56-percent accuracy may seem deceiving, we must put this into perspective. First, we have seven classes, this means that on average we could expect to randomly predict 14.28 percent correctly, but we are four times better than a random prediction. Secondly, the vast majority of the features we use are not directly linked to our outcome but rather artificial features resulting from dimension reduction techniques. While we try to improve our accuracy by looking for brand tags, our results remain relatively low. When we consider the balanced accuracy (BA) of each class, we reach a BA between 60 percent and 74 percent with a mean BA of 68 percent which is higher if we take this criterion into account. We also try to correct class imbalances by using up sampling methods as not to reduce our training set too much but results were inconclusive and we discarded this option.

Finally, if we had the time and computational power to do this, we would extract more sunscreen brand reviews with 50 reviews or more as to have a bigger dataset. Also, advanced machine learning methods such as XGBoost or AdaBoost may improve our prediction because these do not work simultaneously (contrary to random forests) but sequentially, by correcting for each previous learner’s mistakes. These two suggestions may improve our predictive capability.

## 5. CONCLUSION

Studying these two databases gives us a superficial insight on the opinions about sunscreens we could find online.

On Reddit, we see that each document seems distinguishable from the other with no common words whereas it seems that on MakeupAlley, the comments are more similar to one another. Given the structure of the two databases, this first result makes sense as Reddit is a forum with questions and answers and more flexibility in the users‘ inputs when MUA is a review platform with no interactions between the users. With a more flexible Reddit structure, we can expect to see more subjects arising in the conversations as well as more comment formats. Because the Reddit posts generate traffic, the ones generating the most traffic (therefore more controversial) can be spotted as outliers in the LSA. The Reddit corpus also sees the question of cancer discussed when MUA seems more focused on cosmetically elegant ingredients (like with the occurrence of Mexoryl). The brands discussed on Reddit seem different than on MUA. On MUA we selected the brands with most comments and Clinique was one of them, when on Reddit the brand is almost not mentioned.

The sentiment analyses of both platforms are similar but our results are very limited in this field. Indeed, we create a dictionary sentiment from scratch manually, but using a Machine Learning program to generate it would probably have been more unbiased and rigorous. The performance of this analysis is also undermined by the fact that we do not look at the effect of valence shifters using this dictionary. We looked at the valence shifters effect for the MUA database but using a standard existing dictionary that is not specialized in our field. Building a program that would take into account both the cosmetic/health field as well as valence shifters would be optimal for a future analysis.

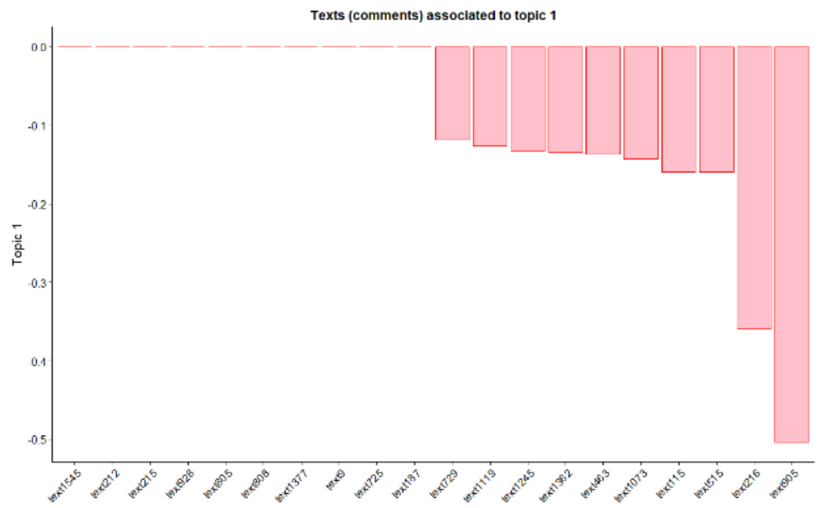
The LDA subjects for both MUA and Reddit seem difficult to interpret, but we see the occurrence of the word face appearing in MUA which we do not see on Reddit.

If we were to make a corpus with all the comments only, without the metadata (because they are not compatible nor structured in the same way), we could create a program which tries to find the source of the text using this feature. This would be an interesting study to conduct to deepen this one.

As far as our Machine Learning program, we focus on predicting brands with a Random Forest algorithm. Further tuning of this algorithm as well as a more extensive grid search would be a more efficient approach. It could also be good to test other advances methods, such as boosting. The choice of predicting brand itself is disputable, as predicting grades, or age range could also be interesting for the brands themselves if they wanted to study this market. More features could be included and created, but ultimately, the choice of these features as well as the choice of the dependent variable to predict would depend on the aim of the person who studies these products.

## 6. APPENDIX

*Figure 1- LSA TEXT ASSOCIATED TO TOPIC 1*



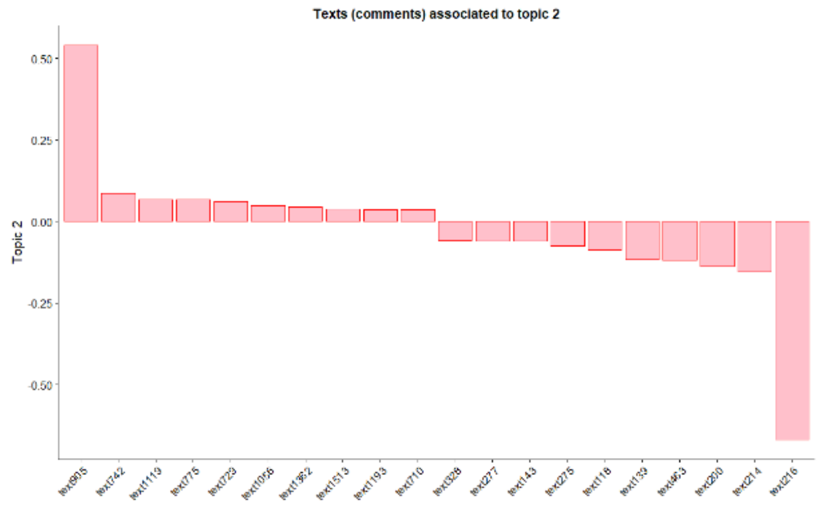
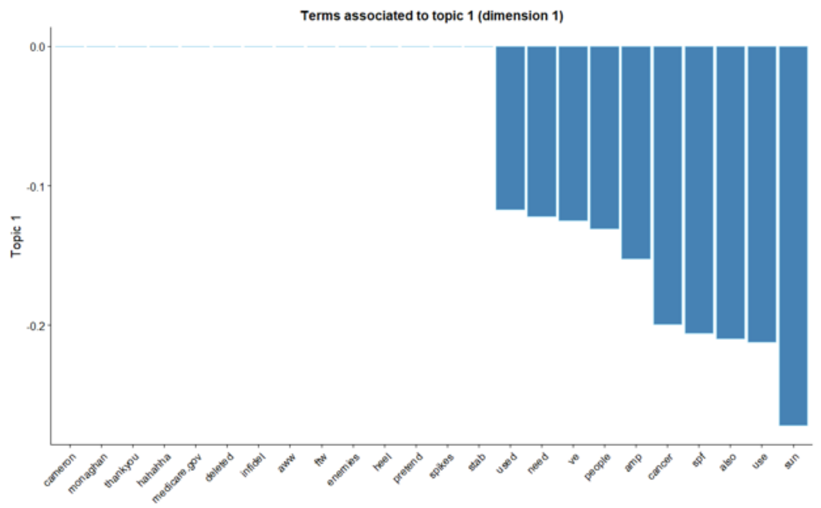


*Figure 2 - LSA TERMS ASSOCITED TO TOPIC 1*



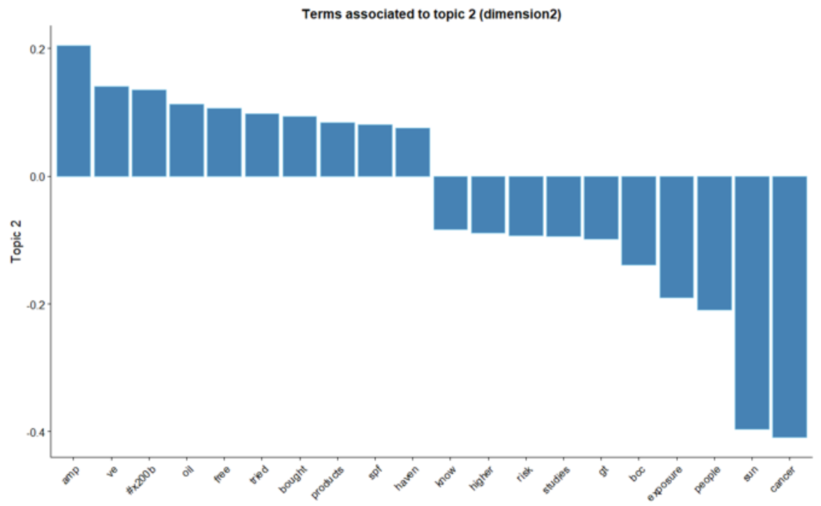


*Figure 3 - DOC PER TOPIC ANALYSIS*

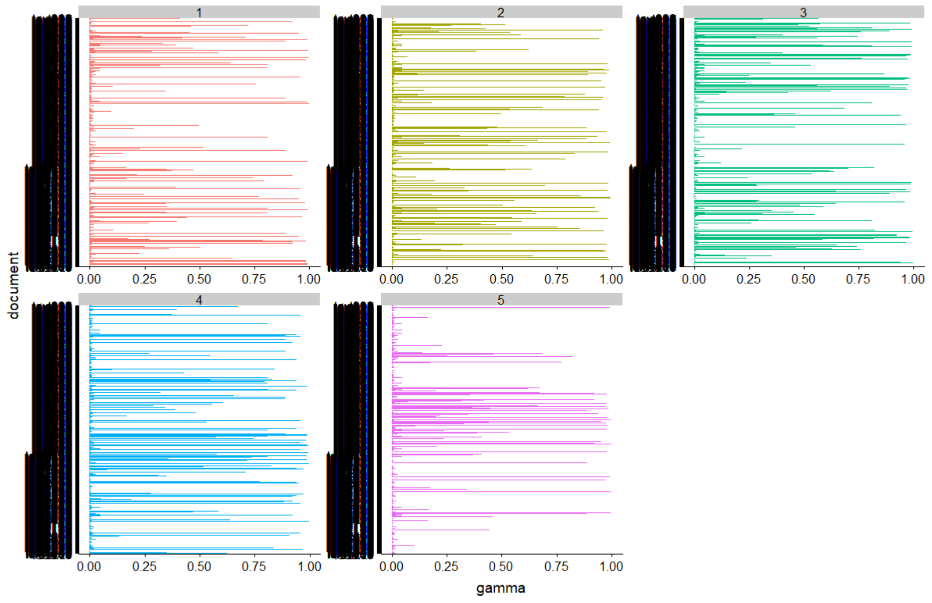


*Figure 4 - BOXPLOT OF THE TRAFFIC GENERATED BY A POST*

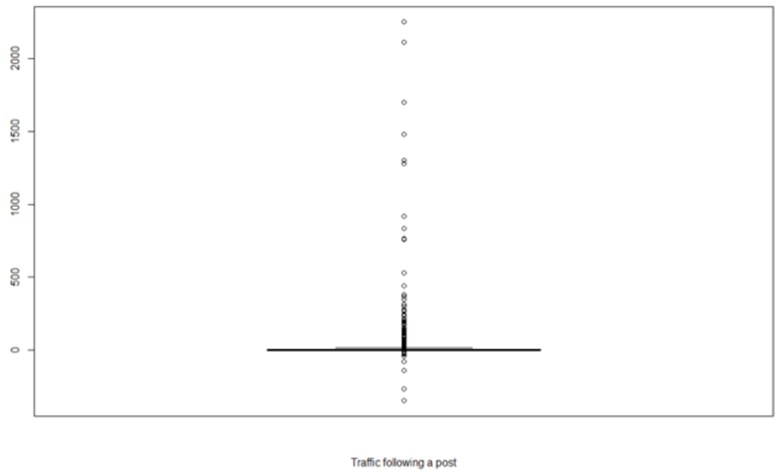
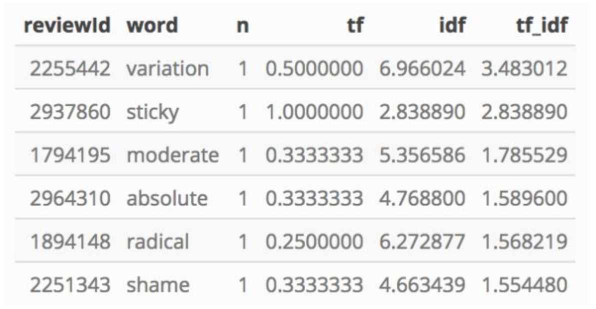
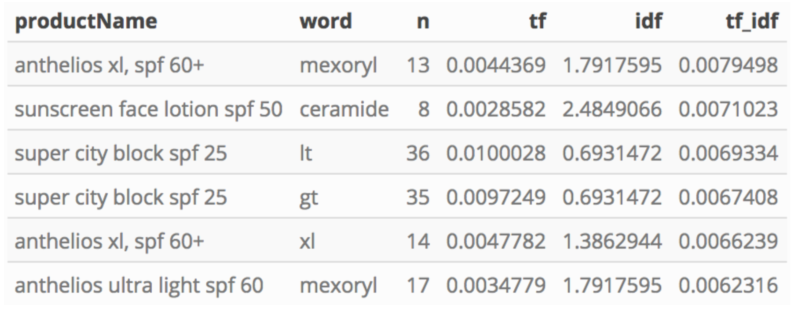


*Figure 5 - TF-IDF PER REVIEW (MUA)*



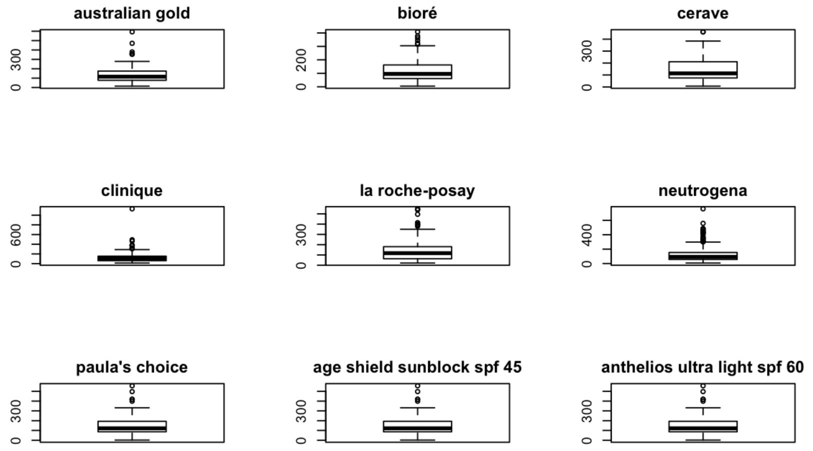
| reviewId | word | n | tf | idf | tf_idf |
|---|---|---|---|---|---|
| 2255442 | variation | 1 | 0.5000000 | 6.966024 | 3.483012 |
| 2937860 | sticky | 1 | 1.0000000 | 2.838890 | 2.838890 |
| 1794195 | moderate | 1 | 0.3333333 | 5.356586 | 1.785529 |
| 2964310 | absolute | 1 | 0.3333333 | 4.768800 | 1.589600 |
| 1894148 | radical | 1 | 0.2500000 | 6.272877 | 1.568219 |
| 2251343 | shame | 1 | 0.3333333 | 4.663439 | 1.554480 |

*Figure 6 - TF-IDF PER PRODUCT (MUA)*



| productName | word | n | tf | idf | tf_idf |
|---|---|---|---|---|---|
| anthelios xl, spf 60+ | mexoryl | 13 | 0.0044369 | 1.7917595 | 0.0079498 |
| sunscreen face lotion spf 50 | ceramide | 8 | 0.0028582 | 2.4849066 | 0.0071023 |
| super city block spf 25 | lt | 36 | 0.0100028 | 0.6931472 | 0.0069334 |
| super city block spf 25 | gt | 35 | 0.0097249 | 0.6931472 | 0.0067408 |
| anthelios xl, spf 60+ | xl | 14 | 0.0047782 | 1.3862944 | 0.0066239 |
| anthelios ultra light spf 60 | mexoryl | 17 | 0.0034779 | 1.7917595 | 0.0062316 |

*Figure 7 - SENTIMENTS WITH VALENCE SHIFTERS (MUA)*



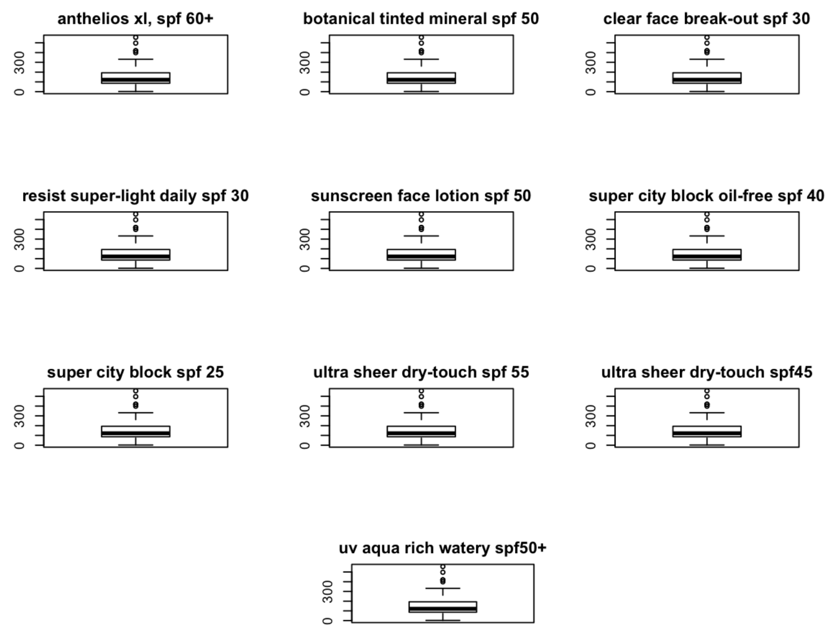


*Figure 8 - SENTIMENT OCCURRENCE PER PRODUCT (MUA)*

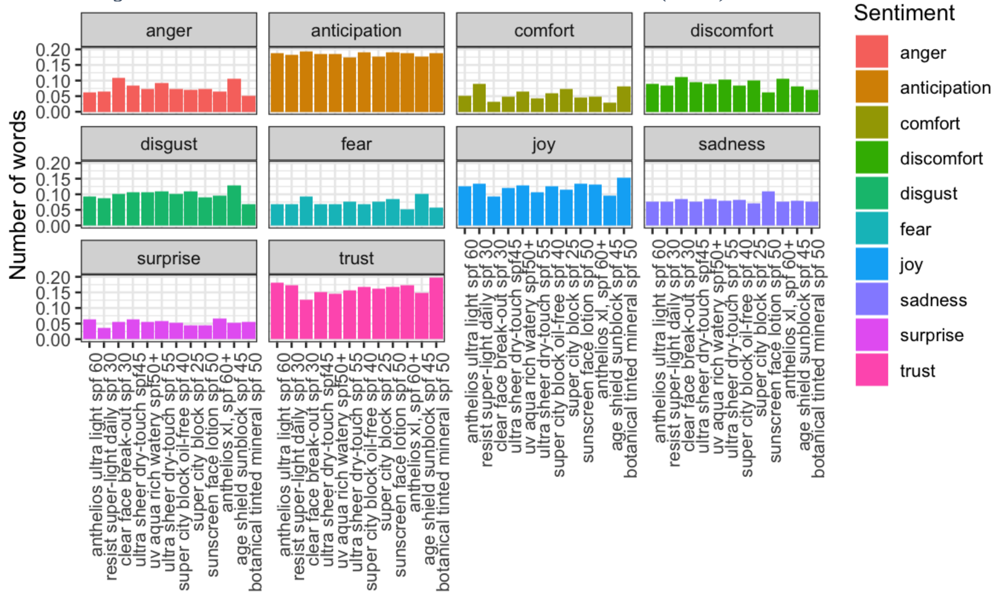


*Figure 9 - SENTIMENT OCCURRENCE PER BRAND (MUA)*

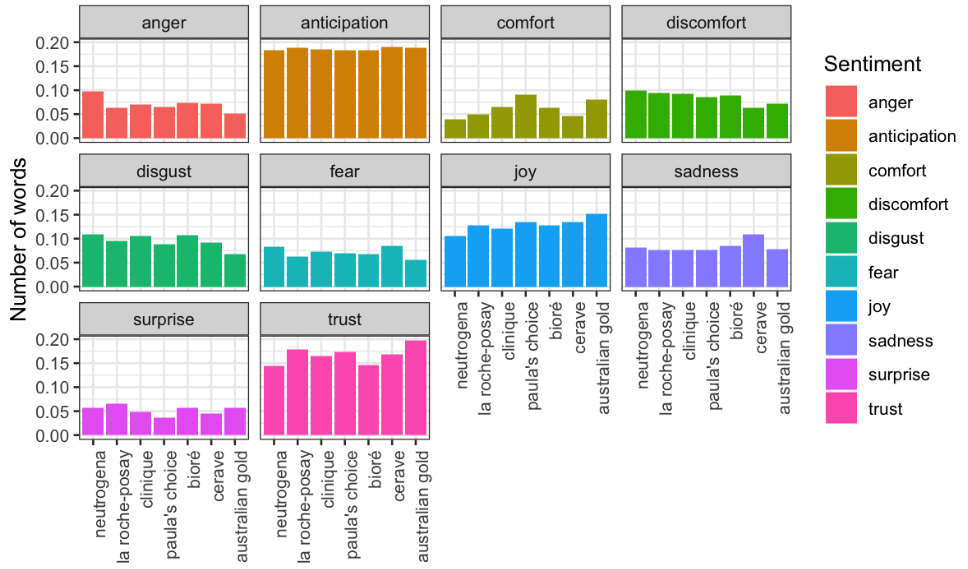


*Figure 10 - COSINE DISTANCE BETWEEN TEXTS ( MUA)*

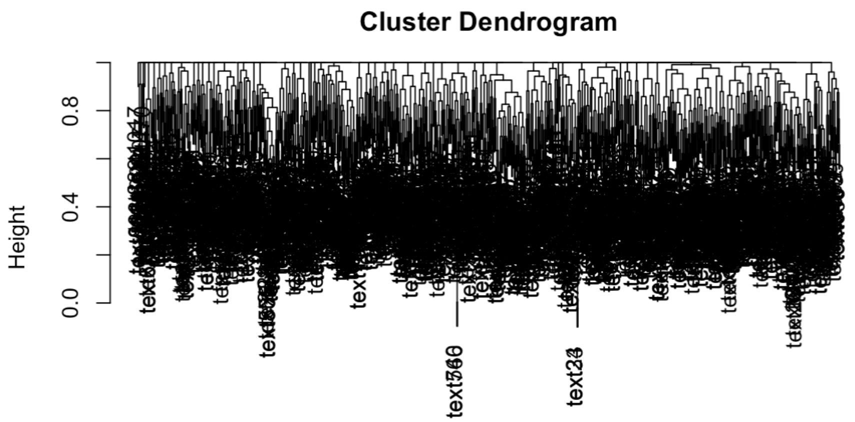
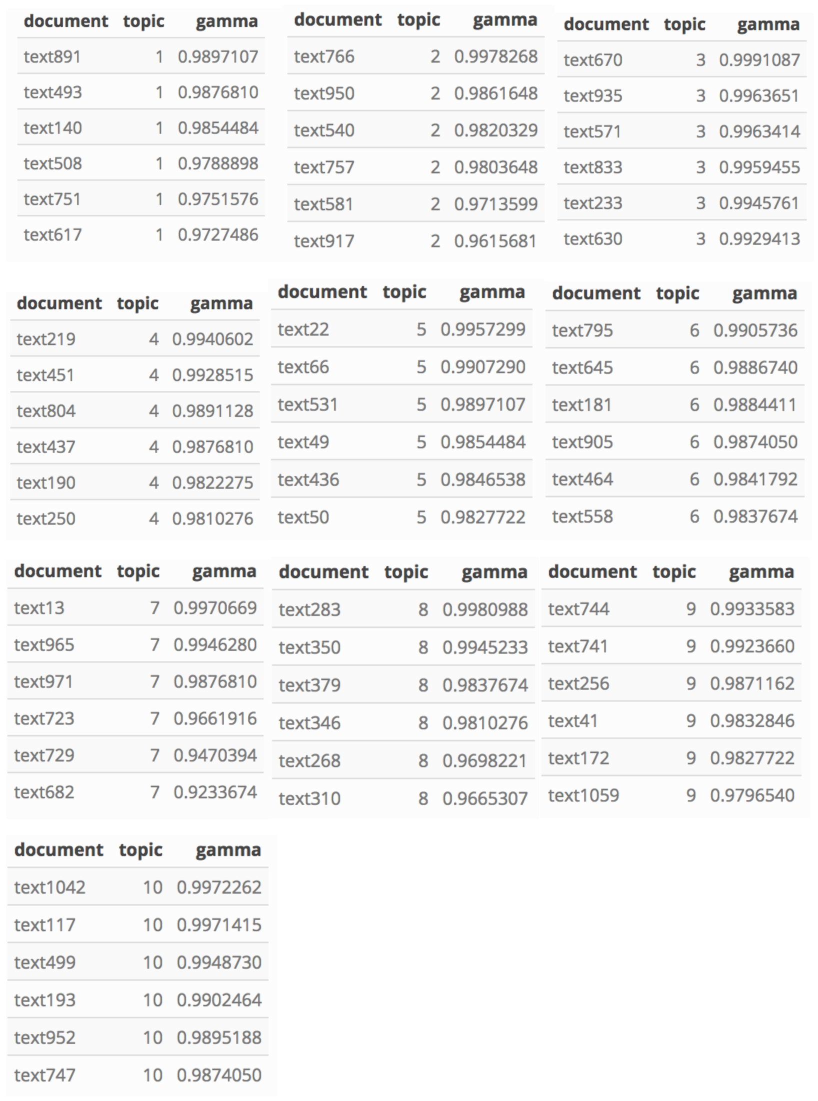


*Figure 11 - TOP GAMMAS PER TOPIC IN LDA ANALYSIS (MUA)*



| document | topic | gamma |
|---|---|---|
| text891 | 1 | 0.9897107 |
| text493 | 1 | 0.9876810 |
| text140 | 1 | 0.9854484 |
| text508 | 1 | 0.9788898 |
| text751 | 1 | 0.9751576 |
| text617 | 1 | 0.9727486 |

| document | topic | gamma |
|---|---|---|
| text766 | 2 | 0.9978268 |
| text950 | 2 | 0.9861648 |
| text540 | 2 | 0.9820329 |
| text757 | 2 | 0.9803648 |
| text581 | 2 | 0.9713599 |
| text917 | 2 | 0.9615681 |

| document | topic | gamma |
|---|---|---|
| text670 | 3 | 0.9991087 |
| text935 | 3 | 0.9963651 |
| text571 | 3 | 0.9963414 |
| text833 | 3 | 0.9959455 |
| text233 | 3 | 0.9945761 |
| text630 | 3 | 0.9929413 |

| document | topic | gamma |
|---|---|---|
| text219 | 4 | 0.9940602 |
| text451 | 4 | 0.9928515 |
| text804 | 4 | 0.9891128 |
| text437 | 4 | 0.9876810 |
| text190 | 4 | 0.9822275 |
| text250 | 4 | 0.9810276 |

| document | topic | gamma |
|---|---|---|
| text22 | 5 | 0.9957299 |
| text66 | 5 | 0.9907290 |
| text531 | 5 | 0.9897107 |
| text49 | 5 | 0.9854484 |
| text436 | 5 | 0.9846538 |
| text50 | 5 | 0.9827722 |

| document | topic | gamma |
|---|---|---|
| text795 | 6 | 0.9905736 |
| text645 | 6 | 0.9886740 |
| text181 | 6 | 0.9884411 |
| text905 | 6 | 0.9874050 |
| text464 | 6 | 0.9841792 |
| text558 | 6 | 0.9837674 |

| document | topic | gamma |
|---|---|---|
| text13 | 7 | 0.9970669 |
| text965 | 7 | 0.9946280 |
| text971 | 7 | 0.9876810 |
| text723 | 7 | 0.9661916 |
| text729 | 7 | 0.9470394 |
| text682 | 7 | 0.9233674 |

| document | topic | gamma |
|---|---|---|
| text283 | 8 | 0.9980988 |
| text350 | 8 | 0.9945233 |
| text379 | 8 | 0.9837674 |
| text346 | 8 | 0.9810276 |
| text268 | 8 | 0.9698221 |
| text310 | 8 | 0.9665307 |

| document | topic | gamma |
|---|---|---|
| text744 | 9 | 0.9933583 |
| text741 | 9 | 0.9923660 |
| text256 | 9 | 0.9871162 |
| text41 | 9 | 0.9832846 |
| text172 | 9 | 0.9827722 |
| text1059 | 9 | 0.9796540 |

| document | topic | gamma |
|---|---|---|
| text1042 | 10 | 0.9972262 |
| text117 | 10 | 0.9971415 |
| text499 | 10 | 0.9948730 |
| text193 | 10 | 0.9902464 |
| text952 | 10 | 0.9895188 |
| text747 | 10 | 0.9874050 |

*Figure 12 - FREQUENCY ANALYSIS*

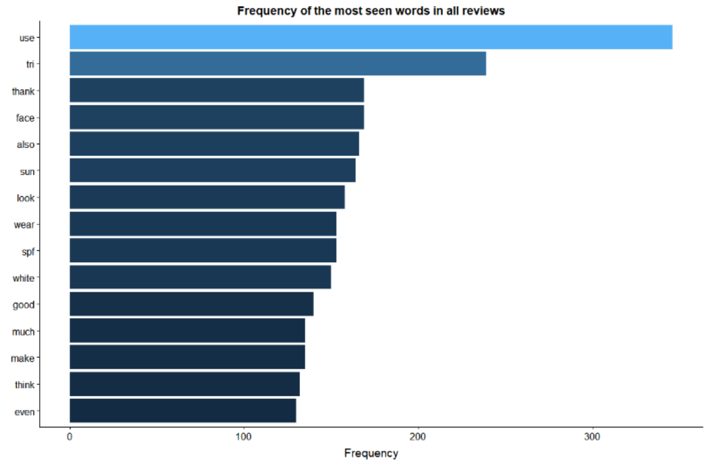


*Figure 13 - FREQUENCY ANALYSIS PER DOCUMENT*

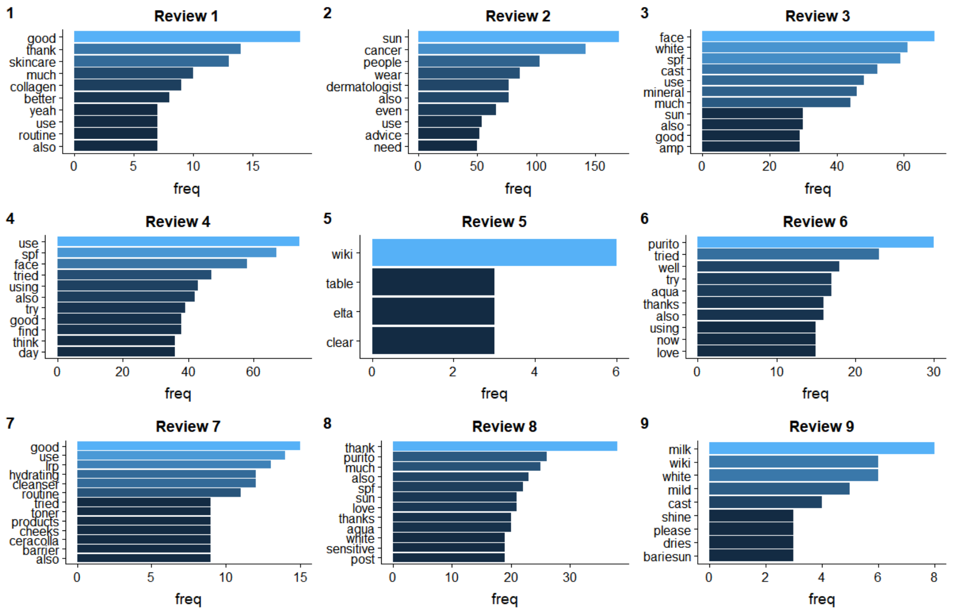


*Figure 14 - SENTIMENT ANALYSIS*

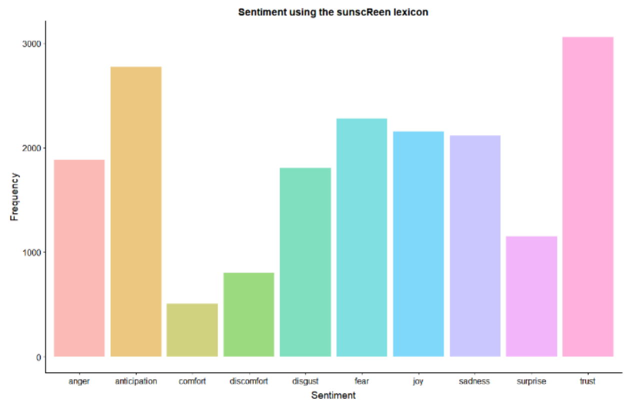


*Figure 15 - LSA*

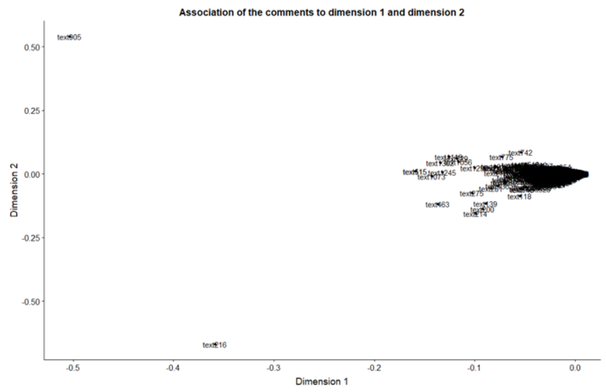


*Figure 16 - LSA TERMS TO DIMENSIONS*

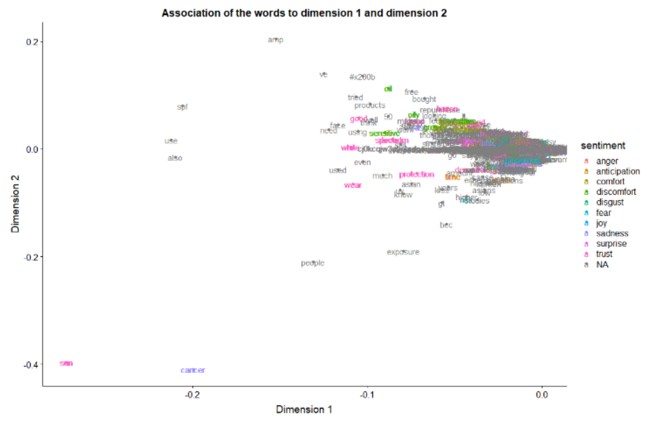


*Figure 17 - TOP TEN DOC-PER-TOPIC LDA (MUA)*

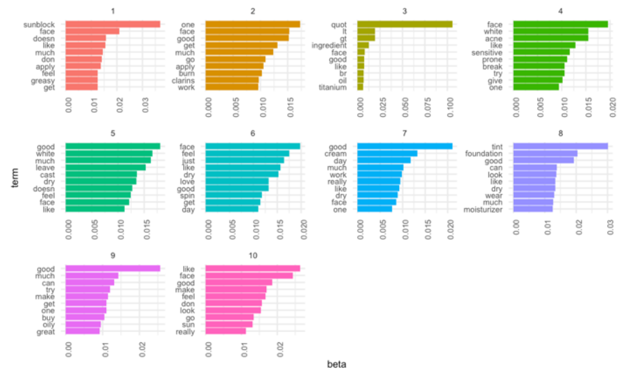


*Figure 18 - TERM-PER-TOPIC LDA (REDDIT)*

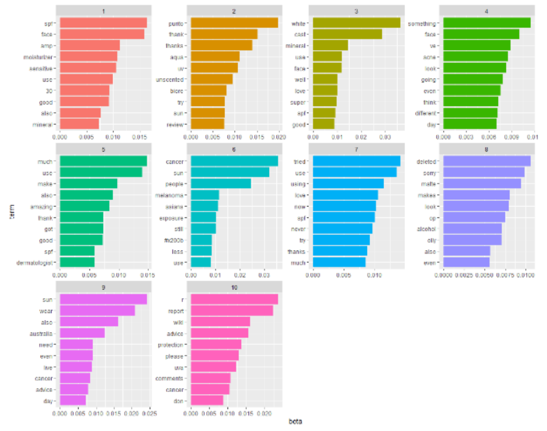


*Figure 19 - TOPIC PER DOCUMENT (REDDIT)*

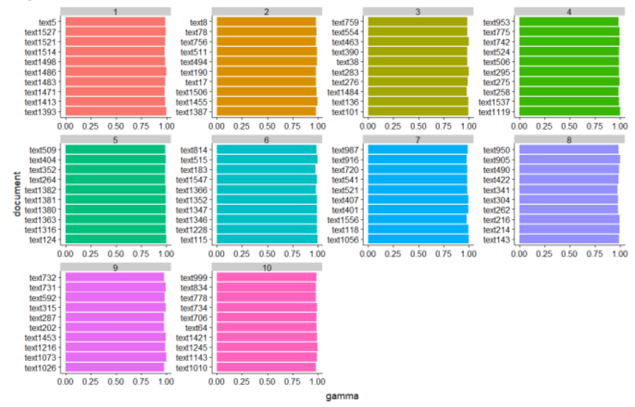


*Figure 20 - BOXPLOT OF TRAFFIC FOLLOWING A POST*

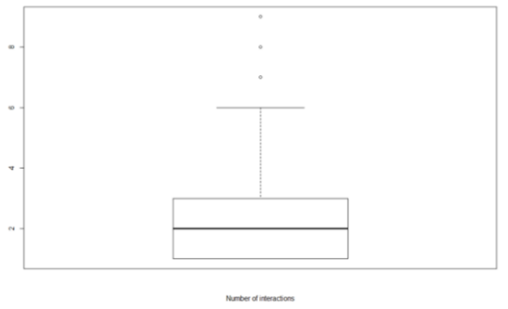


*Figure 21 - QUANTITY OF COMMENTS GENERATED PER MONTH*

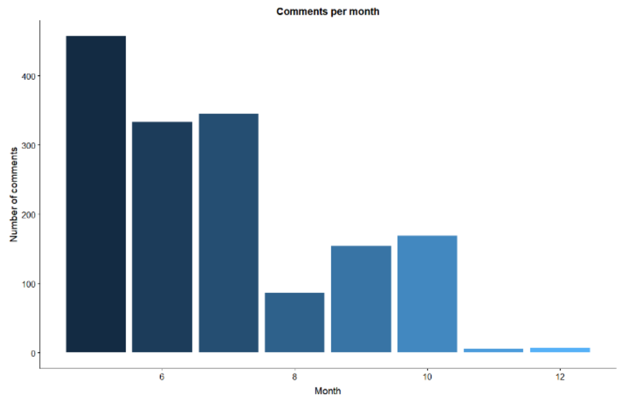


*Figure 22 - APPEARANCE OF BRANDS IN REVIEWS*

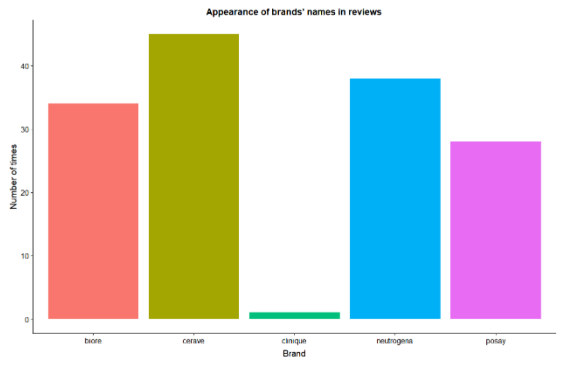


*Figure 23 - APPEARANCE OF BRANDS PER REVIEWS*

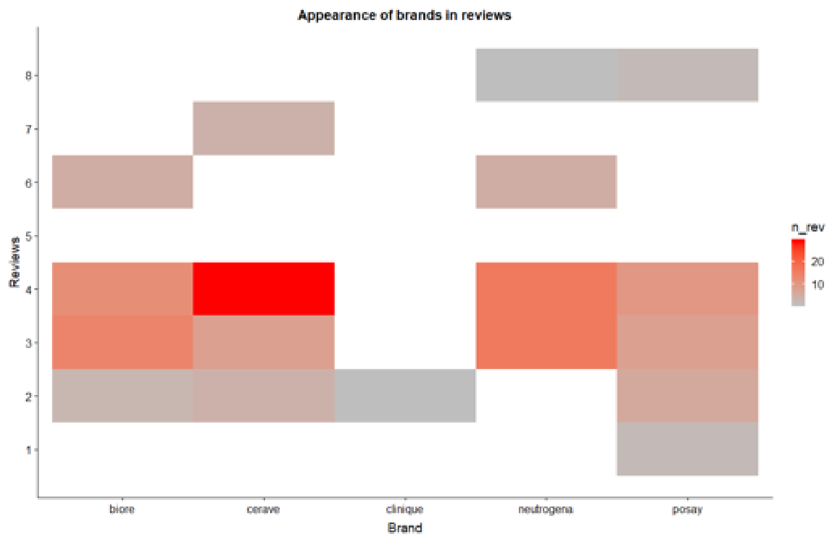


*Figure 24 - BRAND ANALYSIS*

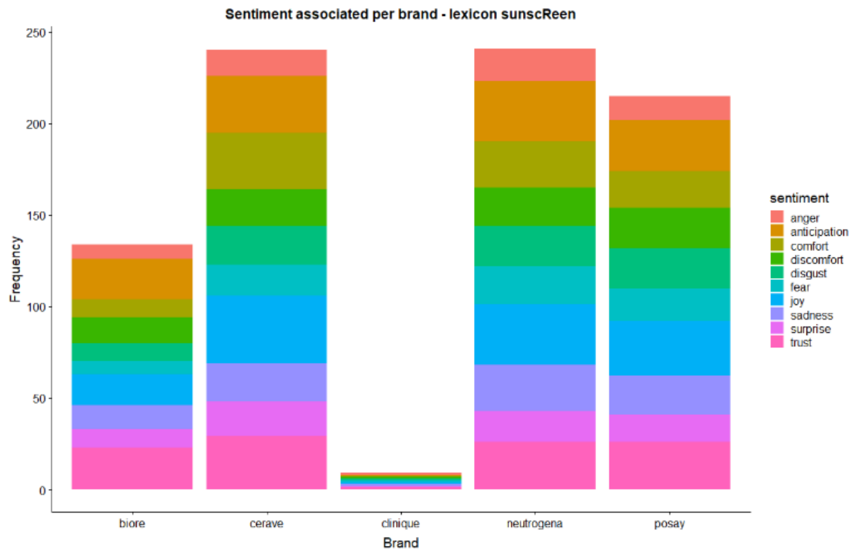


*Figure 25 - ACCURACY OF RANDOM FOREST*

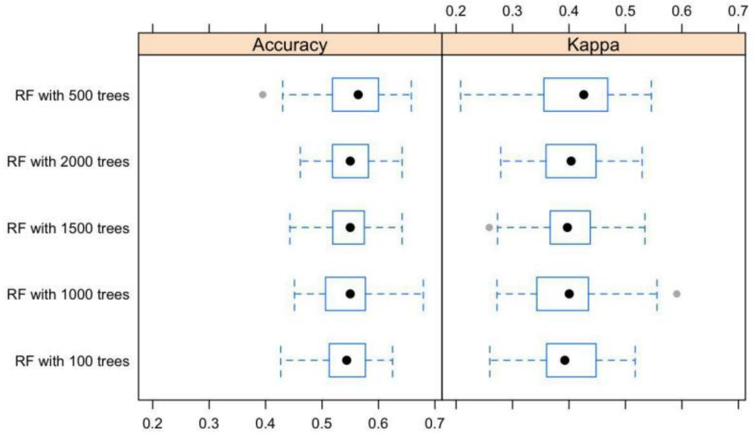
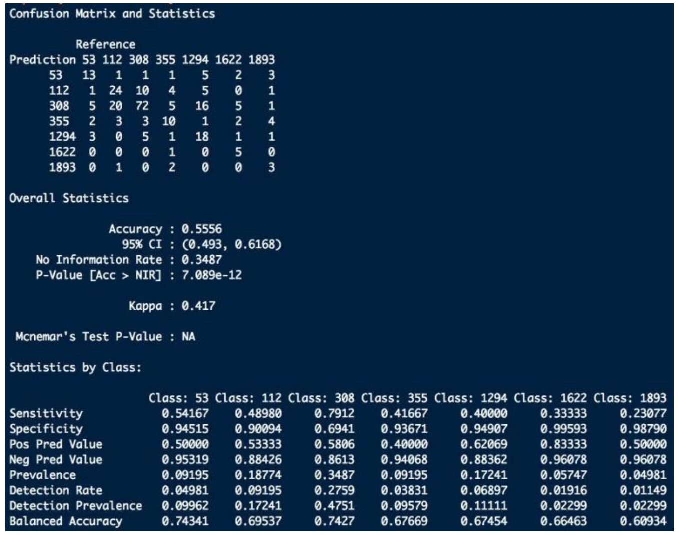
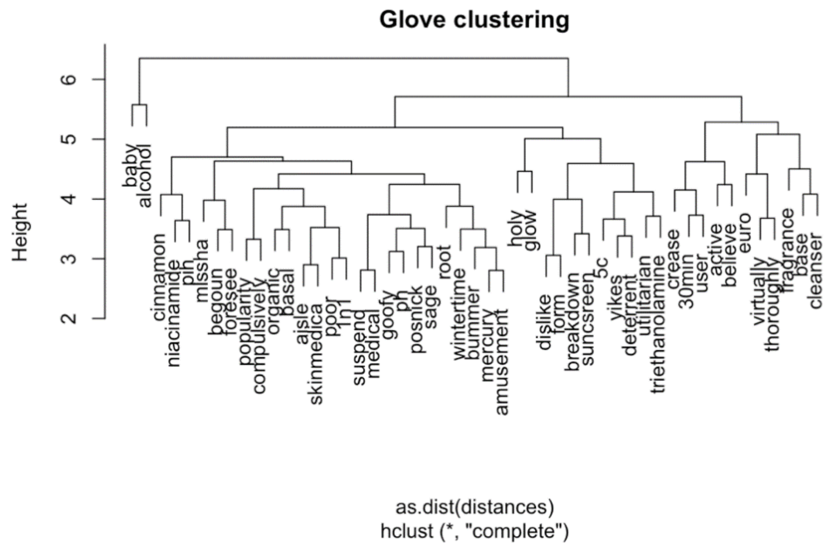


*Figure 26 - SUMMARY OF RANDOM FOREST WITH 500 TREES*



```
Confusion Matrix and Statistics

          Reference
Prediction 53 112 308 355 1294 1622 1893
      53   13   1   1   1    5    2    3
      112   1  24  10   4    5    0    1
      308   5  20  72   5   16    5    1
      355   2   3   3  10    1    2    4
      1294  3   0   5   1   18    1    1
      1622  0   0   0   1    0    5    0
      1893  0   1   0   2    0    0    3

Overall Statistics

               Accuracy : 0.5556
                 95% CI : (0.493, 0.6168)
    No Information Rate : 0.3487
    P-Value [Acc > NIR] : 7.089e-12

                  Kappa : 0.417

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 53 Class: 112 Class: 308 Class: 355 Class: 1294 Class: 1622 Class: 1893
Sensitivity            0.54167    0.48980     0.7912    0.41667     0.40000     0.33333     0.23077
Specificity            0.94515    0.90094     0.6941    0.93671     0.94907     0.99593     0.98790
Pos Pred Value         0.50000    0.53333     0.5806    0.40000     0.62069     0.83333     0.50000
Neg Pred Value         0.95319    0.88426     0.8613    0.94068     0.88362     0.96078     0.96078
Prevalence             0.09195    0.18774     0.3487    0.09195     0.17241     0.05747     0.04981
Detection Rate         0.04981    0.09195     0.2759    0.03831     0.06897     0.01916     0.01149
Detection Prevalence   0.09962    0.17241     0.4751    0.09579     0.11111     0.02299     0.02299
Balanced Accuracy      0.74341    0.69537     0.7427    0.67669     0.67454     0.66463     0.60934
```

*Figure 27 - GLOVE TREE CLUSTERING (MUA)*



CODE REDDIT:

library(c(“RedditExtractoR”,”dplyr”,”tidytext”,”readxl”,”tibble”,”ggplot2”,”tm”,”lexicon”,”textstem”,”stringr”,”textdata”,”sentimenr”,”quanteda”,”topicmodels”, “magrittr”,”readr”,”lubridate”,”cowplot”))

```
d1 <-reddit_content(URL="https://www.reddit.com/r/SkincareAddiction/comments/craf60/selfie_after_a_long_time_of_search_i_have_finally/")
d2 <- reddit_content(URL="https://www.reddit.com/r/SkincareAddiction/comments/btx79r/sun_care_dermatologist_told_me_to_ditch_sunscreen/")
d3 <- reddit_content(URL="https://www.reddit.com/r/SkincareAddiction/comments/c7x8ke/product_question_30_minutes_after_applying/")
d4 <-reddit_content(URL="https://www.reddit.com/r/SkincareAddiction/comments/c096h9/review_me_6_months_ago_sunscreen_is_so_greasy_and/")
d5 <- reddit_content((URL="https://www.reddit.com/r/SkincareAddiction/comments/dmif3o/review_2_skinceutical_sunscreens_and_2_elta_md/"))
d6 <- reddit_content(URL="https://www.reddit.com/r/SkincareAddiction/comments/dcern5/review_the_10_sunscreens_ive_tried_in_my_hg/")
d7 <- reddit_content(URL="https://www.reddit.com/r/SkincareAddiction/comments/dkppo4/sun_care_european_high_uva_sunscreens_for/")
d8 <- reddit_content(URL="https://www.reddit.com/r/SkincareAddiction/comments/d5x4g0/11_sunscreens_for_sensitive_skin_at_low_price/")
d9 <- reddit_content(URL="https://www.reddit.com/r/SkincareAddiction/comments/df7l2i/review_barisun_50_uvauvb_and_anessa_50_pa/")
dtotal <- rbind(d1,d2,d3,d4,d5,d6,d7,d8,d9)
dtotal.tib <- tibble(text=dtotal$comment, doc=c(1:nrow(dtotal)))
dtot.tok <- dtotal.tib%>%   unnest_tokens(word, text, to_lower=TRUE) %>%  count(doc, word, sort=TRUE) %>%  ungroup()
dtot.cp <- VCorpus(VectorSource(dtot.tok$word))
dtot.cp <- tm_map(dtot.cp, removeWords, stopwords("english"))
dtot.cp <- tm_map(dtot.cp,  removeWords, c("m","s","t","skin", "sunscreen", "sunscreens", "like",  "get", "one", "just","can", "really", "skincareaddiction",
"www.reddit.com","https"))
dtot.cp <- tm_map(dtot.cp, stemDocument)
lemmatize_words(dtot.cp, dictionary=hash_lemmas)
dtot.dtm <- DocumentTermMatrix(dtot.cp)
dtot.tfidf <- bind_tf_idf(dtot.tok, word, doc, n)
tfidf <- dtot.tfidf %>%  select(word,tf_idf) %>% group_by(word) %>%mutate(tfidf = mean(tf_idf)) %>% select(word,tfidf) %>%group_by(word,tfidf) %>%
na.omit() %>% summarise()
dtot.fr <- colSums(as.matrix(dtot.dtm))
dtot.df <- data.frame(word=names(dtot.fr), freq=dtot.fr)
ggplot(top_n(dtot.df, n=15), aes(reorder(word,freq),freq, fill=freq))+  geom_col()+  xlab(NULL)+  coord_flip()+  ggtitle("Frequency of the most seen words in
all reviews")+  labs(x="", y="Frequency")+ theme(legend.position = "None")
get_sunsentiments <- function(lexicon = c("sunscReen")) {  lexicon <- match.arg(lexicon)  sunscReen = lexicon_sunscReen()}
lexicon_sunscReen <- function() {  readRDS("data/sunscReen.rds")}
dtot.sentiment.sunscReen <- dtot.tok %>%  right_join(get_sunsentiments("sunscReen")) %>%  count(sentiment)
ggplot(dtot.sentiment.sunscReen, aes(sentiment,nn,fill=sentiment)) +  geom_bar(alpha=0.5, stat="identity", show.legend=F) +  ggtitle("Sentiment using the
sunscReen lexicon") +labs(x="Sentiment", y="Frequency")
dtotal.pol <- sentiment_by(dtotal$comment)
freq.data <- function(data) {dd.tok <- data %>%   unnest_tokens(word, comment, to_lower=TRUE)
dd.cp <- VCorpus(VectorSource(dd.tok$word))
dd.cp <- tm_map(dd.cp, removeWords, stopwords("english"))
dd.cp <- tm_map(dd.cp, removeWords, c("m","s","t","skin", "sunscreen", "sunscreens", "like", "get", "one", "just","can", "really", "skincareaddiction",
"www.reddit.com","https"))
dd.dtm <- DocumentTermMatrix(dd.cp)
dd.fr <- colSums(as.matrix(dd.dtm))
dd.df <- data.frame(word=names(dd.fr), freq=dd.fr)
return(dd.df)}
d1.df <- freq.data(d1)
d2.df <- freq.data(d2)
d3.df <- freq.data(d3)
d4.df <- freq.data(d4)
d5.df <- freq.data(d5)
d6.df <- freq.data(d6)
d7.df <- freq.data(d7)
d8.df <- freq.data(d8) d9.df <- freq.data(d9)
freq1 <- ggplot(top_n(d1.df, n=10), aes(reorder(word,freq),freq , fill=freq))+geom_col()+xlab(NULL)+coord_flip()+ggtitle("Review 1") +theme(legend.position
= "None")
freq2 <- ggplot(top_n(d2.df, n=10), aes(reorder(word,freq),freq , fill=freq))+geom_col()+xlab(NULL)+coord_flip()+ggtitle("Review 2") +theme(legend.position
= "None")
freq3 <- ggplot(top_n(d3.df, n=10), aes(reorder(word,freq),freq , fill=freq))+geom_col()+xlab(NULL)+coord_flip()+ggtitle("Review 3") +theme(legend.position
= "None")
```

```
freq4 <- ggplot(top_n(d4.df, n=10), aes(reorder(word,freq),freq , fill=freq))+geom_col()+xlab(NULL)+coord_flip()+ggtitle("Review 4") +theme(legend.position
= "None")
freq5 <- ggplot(top_n(d5.df, n=4), aes(reorder(word,freq),freq , fill=freq))+geom_col()+xlab(NULL)+coord_flip()+ggtitle("Review 5") +theme(legend.position =
"None")
freq6 <- ggplot(top_n(d6.df, n=10), aes(reorder(word,freq),freq , fill=freq))+geom_col()+xlab(NULL)+coord_flip()+ggtitle("Review 6") +theme(legend.position
= "None")
freq7 <- ggplot(top_n(d7.df, n=10), aes(reorder(word,freq),freq , fill=freq))+geom_col()+xlab(NULL)+coord_flip()+ggtitle("Review 7") +theme(legend.position
= "None")
freq8 <- ggplot(top_n(d8.df, n=10), aes(reorder(word,freq),freq , fill=freq))+geom_col()+xlab(NULL)+coord_flip()+ggtitle("Review 8") +theme(legend.position
= "None")
freq9 <- ggplot(top_n(d9.df, n=9), aes(reorder(word,freq),freq , fill=freq))+geom_col()+xlab(NULL)+coord_flip()+ggtitle("Review 9") +theme(legend.position =
"None")
plot_grid(freq1,freq2,freq3,freq4,freq5,freq6,freq7,freq8,freq9, labels=c("1","2","3","4","5","6","7","8","9"), ncol = 3, nrow = 3)
dtot.cp2 <- corpus(dtotal$comment)
dfmat <- dfm(dtot.cp2, tolower = TRUE, remove = c(stopwords("english"), c("m","s","t","skin", "sunscreen", "sunscreens", "like", "get", "one", "just","can",
"really", "skincareaddiction", "www.reddit.com","https")), stem = FALSE, remove_punct = TRUE)
tmod <- textmodel_lsa(dfmat, nd=5) #we chose 5 nodes
head(tmod$docs) head(tmod$features)
df.doc <- data.frame(dim1=tmod$docs[,1], dim2=tmod$docs[,2])
rownames(df.doc) <- rownames(tmod$docs)
ggplot(df.doc, aes(x=dim1, y=dim2)) +  geom_point() +   geom_text(label=rownames(df.doc)) +  ggtitle("Association of the comments to dimension 1 and
dimension 2")+labs(x="Dimension 1", y="Dimension 2")
df.feat <- data.frame(dim1=tmod$features[,1], dim2=tmod$features[,2], rownames(tmod$features))
rownames(df.feat) <- rownames(tmod$features)
dtot.tok.sent <- dtot.tok %>% right_join(get_sunsentiments("sunscReen"))
df.feat <- left_join(df.feat, dtot.tok.sent, by=c("rownames.tmod.features."="word"))
ggplot(df.feat, aes(x=dim1, y=dim2, col=sentiment)) +  geom_point() +   geom_text(label=df.feat$rownames.tmod.features.) +  ggtitle("Association of the words
to dimension 1 and dimension 2")+labs(x="Dimension 1", y="Dimension 2")
dfmat.test <- tmod$docs %*% diag(tmod$sk) %*% t(tmod$features)
range(dfmat.test - tmod$matrix_low_rank )
lsa.terms.tbl <- tibble(Term=rownames(tmod$features), topic1=tmod$features[,1],   topic2=tmod$features[,2])
topic1.tbl <- rbind(top_n(lsa.terms.tbl, wt=topic1, n=10), top_n(lsa.terms.tbl, wt=topic1, n=-10))
ggplot(topic1.tbl, aes(x = reorder(Term, -topic1), y = topic1))+  geom_bar(stat="identity", color='skyblue',fill='steelblue') +
theme(axis.text.x=element_text(angle=45, hjust=1)) +  ggtitle("Terms associated to topic 1 (dimension 1)") +  labs(x="", y="Topic 1")
topic2.tbl <- rbind(top_n(lsa.terms.tbl, wt=topic2, n=10), top_n(lsa.terms.tbl, wt=topic2, n=-10))
ggplot(topic2.tbl, aes(x = reorder(Term, -topic2), y = topic2))+  geom_bar(stat="identity", color='skyblue',fill='steelblue') +
theme(axis.text.x=element_text(angle=45, hjust=1)) +  ggtitle("Terms associated to topic 2 (dimension2)")+  labs(x="", y="Topic 2")
lsa.doc.tbl <- tibble(Doc=rownames(tmod$docs), topic1=tmod$docs[,1],     topic2=tmod$docs[,2])
topic1.tbl <- rbind(top_n(lsa.doc.tbl, wt=topic1, n=10), top_n(lsa.doc.tbl, wt=topic1, n=-10))
ggplot(topic1.tbl, aes(x = reorder(Doc, -topic1), y = topic1))+  geom_bar(stat="identity", color='red',fill='pink') +  theme(axis.text.x=element_text(angle=45,
hjust=1)) +  ggtitle("Texts (comments) associated to topic 1") +  labs(x=" ", y="Topic 1")
topic2.tbl <- rbind(top_n(lsa.doc.tbl, wt=topic2, n=10), top_n(lsa.doc.tbl, wt=topic2, n=-10))
ggplot(topic2.tbl, aes(x = reorder(Doc, -topic2), y = topic2))+  geom_bar(stat="identity", color='red',fill='pink') +  theme(axis.text.x=element_text(angle=45,
hjust=1)) +  ggtitle("Texts (comments) associated to topic 2") +  labs(x=" ", y="Topic 2")
dtm <- convert(dfmat, to = "topicmodels")
lda <- LDA(dtm, k = 10) #10 topics
terms(lda, 5)  topics(lda, 5) lda@beta[,1:10] lda@gamma
beta.td <- tidy(lda, matrix = "beta")
beta.top.terms <- beta.td %>%group_by(topic) %>%  top_n(10, beta) %>%  ungroup() %>%arrange(topic, -beta)
beta.top.terms %>%  mutate(term = reorder_within(term, beta, topic)) %>%  ggplot(aes(term, beta, fill = factor(topic))) +  geom_col(show.legend = FALSE) +
facet_wrap(~ topic, scales = "free") +  coord_flip() +  scale_x_reordered()
gamma.td <- tidy(lda, matrix = "gamma")
gamma.top10 <- gamma.td %>%  group_by(topic) %>%  top_n(10, gamma) %>%  ungroup() %>%  arrange(topic, -gamma)
gamma.top10 %>%  ggplot(aes(document, gamma, fill = factor(topic))) +  geom_col(show.legend = FALSE) +  facet_wrap(~ topic, scales = "free") +
coord_flip() +  scale_x_reordered()
augment(lda)
boxplot(dtotal$comment_score, xlab="Traffic following a post")
```

```
dtotal_positive <- dtotal %>%   filter(dtotal$comment_score < 10 & dtotal$comment_score > 0)
boxplot(dtotal_positive$comment_score, xlab="Number of interactions")
dtotal$comm_date <-  as_date(dtotal$comm_date)
dtotal <- dtotal %>%  mutate(mois = month(dtotal$comm_date))
postingmonths <- dtotal %>% select(mois) %>%  group_by(mois) %>% count()
ggplot(postingmonths, aes(x=mois, y=n, fill=mois)) +geom_bar(stat = "identity") + labs(x="Month", y="Number of comments") +ggtitle("Comments per
month") +theme(legend.position = "none")
neutrogena <- dtot.tok %>%   filter(word=="neutrogena")
biore <- dtot.tok %>%  filter(word=="biore") cerave <- dtot.tok %>%   filter(word=="cerave") paula.choice <- dtot.tok %>%   filter(word=="paula s choice")
australian.god <- dtot.tok %>%   filter(word=="australian god") roche.posay <- dtot.tok %>%   filter(word=="posay") clinique <- dtot.tok %>%
filter(word=="clinique") brands <- rbind(neutrogena,biore,cerave,clinique, roche.posay, australian.god)
reviews <- data.frame(num_comments =
c(d1[1,]$num_comments,d2[1,]$num_comments,d3[1,]$num_comments,d4[1,]$num_comments,d5[1,]$num_comments,d6[1,]$num_comments,d7[1,]$num_co
mments,d8[1,]$num_comments,d9[1,]$num_comments),review = c(1,2,3,4,5,6,7,8,9))
dtotal <- dtotal %>% mutate(doc=1:nrow(dtotal))
brands <- left_join(brands, dtotal) brands <- left_join(brands,reviews) brands$doc <- as.factor(brands$doc)
ggplot(data=brands, aes(x=word, y=n, fill=word)) +geom_bar(stat="identity") + labs(x="Brand", y="Number of times") + ggtitle("Appearance of brands' names
in reviews") +theme(legend.position = "none") ggplot(data=brands, aes(x=word,y=doc)) + geom_tile(aes(fill=n))+  ggtitle("Appearance of brands in comments")
dd$doc <- as.factor(dd$doc)
brands <- brands %>%  group_by(review, word) %>%  mutate(n_rev = sum(n)) %>%   ungroup()
ggplot(data=brands, aes(x=word,y=review)) + geom_tile(aes(fill=n_rev))+  ggtitle("Appearance of brands in
reviews")+scale_y_continuous(breaks=c(1,2,3,4,5,6,7,8,9)) +  scale_fill_gradient(low="gray", high="red") + labs(x="Brand", y="Reviews")
brands.tib <- tibble(text=brands$comment, doc=c(1:nrow(brands)))
brands.tok <- brands.tib%>% unnest_tokens(word, text, to_lower=TRUE) %>%count(doc, word, sort=TRUE) %>% ungroup()
brands.sentiment.sunscReen <- brands.tok %>% right_join(get_sunsentiments("sunscReen"))
brands.sentiment.doc <- brands.sentiment.sunscReen %>% group_by(doc) %>%   count(sentiment) %>%   na.omit()
brands.sentiment.doc <- left_join(brands.sentiment.doc,brands.tib, by="doc")
brands.sentiment.doc <- left_join(brands.sentiment.doc, brands, by=c("text"="comment"))
brands.sentiment.brand <- brands.sentiment.doc %>%   group_by(word) %>% count(sentiment)
ggplot(brands.sentiment.brand, aes(x = word, y = nnn, fill=sentiment)

geom_bar(stat="identity") + ggtitle("Sentiment associated per brand - lexicon sunscReen") + labs(x="Brand", y="Frequency")
```

**MAKEUPALLEY:**

```
library(c(“caret”,”data.table”,”ggforce”,”ggwordcloud”,”kableExtra”,”keras”,”purrr”, “RColorBrewer” ,”readr” ,”Rsentiment” ,”rword2vec”
,”sentimentr” ,”text2vec”,”tidyverse”, “randomForest”))

sunscreen <-  list.files(pattern = "*.csv") %>%  map_df( ~ read_csv(.))

sunscreen$review <- sunscreen$review %>% tolower()

sunscreen[c(2:5, 7:8, 14, 16:23)] <-lapply(sunscreen[c(2:5, 7:8, 14, 16:23)], factor)

levels(sunscreen$productName) <- c(  "Age Shield Sunblock SPF 45",  "Anthelios Ultra Light SPF 60" ,  "Anthelios XL, SPF 60+",  "Botanical
Tinted Mineral SPF 50",  "Clear Face Break-Out SPF 30",  "RESIST Super-Light Daily SPF 30",  "Sunscreen Face Lotion SPF 50",  "Super City
Block Oil-Free SPF 40",  "Super City Block SPF 25",  "Ultra Sheer Dry-Touch SPF 55",  "Ultra Sheer Dry-Touch SPF45",  "UV Aqua Rich Watery
SPF50+")

brands <- levels(sunscreen$brandName) %>% tolower() %>% c()

productnames <-evels(sunscreen$productName %>% as.factor()) %>% tolower() %>% c()

sunscreen$review <-lemmatize_strings(sunscreen$review, dictionary = hash_lemmas)

sunscreen$review <- tolower(sunscreen$review)

sunscreen$brandName <- tolower(sunscreen$brandName)

sunscreen$productName <- tolower(sunscreen$productName)

my_stop_words <- c( word = c("#","s","ve","re","skin","sunscreen","product",  "spf", brands,  productnames,"shield",  "sunscreen","sunscreens", "t",
"it", "It", "use" ), c(stopwords::stopwords("en")) ) %>% as_tibble()

my_stop_words$word <- my_stop_words$value

sunscreen_cleaned <- sunscreen %>%mutate(review2 = review) %>%  as.tibble() %>%  unnest_tokens(word, review) %>%anti_join(stop_words, by
= "word") %>%  anti_join(my_stop_words, by = "word") %>% filter(is.na(as.numeric(word)))

tf_byreview <- sunscreen_cleaned %>%  group_by(reviewId) %>%  count(word) %>% ungroup()
```

```
tfidf_byreview <- tf_byreview %>% tidytext::bind_tf_idf(word, reviewId, n)

tfidf_byreview %>% dplyr::arrange(desc(tf_idf)) %>% head() %>% kable() %>%kable_styling(  bootstrap_options = c("striped", "hover",
"condensed"),  full_width = FALSE)

tf_byproduct <- sunscreen_cleaned %>%group_by(productName) %>%  count(word) %>%  ungroup()

tfidf_product <- tf_byproduct %>%  bind_tf_idf(word, productName, n)

tfidf_product %>%  dplyr::arrange(desc(tf_idf)) %>%head() %>%  kable() %>%  kable_styling(  bootstrap_options = c("striped", "hover",
“condensed"), full_width = FALSE)

for (i in c(1:length(sunscreen$review))) {sunscreen$sentiments[i] <- sentiment_by(sunscreen$review[i], hash_valence_shifters)}

sunscreen$sentiments <- sunscreen$sentiments %>%as.numeric()

sentiment_valence_shifter_bybrand <- sunscreen %>%group_by(brandName) %>%mutate(sentimentbybrand = mean(sentiment))

sentiment_valence_shifter_byproduct <-sentiment_valence_shifter_bybrand %>%group_by(productName) %>%mutate(sentimentbyproduct =
mean(sentiment))

for (in brands) {x <- sentiment_valence_shifter_bybrand %>%filter(brandName == i)

boxplot(x$sentiments, main = paste(i))+coord_cartesian(ylim=c(0,400))}

for (i in productnames) {y <- sentiment_valence_shifter_byproduct %>%filter(productName == i)

boxplot(x$sentiments, main = paste(i))}

get_sunsentiments <- function(lexicon = c("sunscReen")) {lexicon <- match.arg(lexicon)

sunscReen = lexicon_sunscReen()}

lexicon_sunscReen <- function() {readRDS("../data/sunscReen.rds")}

spf <- str_extract_all(sunscreen$productName, "\\d+") %>%as.data.frame() %>%t()

sunscreen <- cbind(sunscreen, spf) my_stop_words <- as_tibble(my_stop_words) my_stop_words$word <- my_stop_words$value

sunscreen_cleaned_for_wordcloud <- sunscreen %>%dplyr::mutate(review2 = review) %>%dplyr::group_by(brandName) %>%as.tibble()
%>%tidytext::unnest_tokens(word, review) %>%anti_join(stop_words, by = "word") %>%anti_join(my_stop_words, by = "word")
%>%dplyr::filter(is.na(as.numeric(word)))

sentiment_by_brand_sun <- sunscreen_cleaned_for_wordcloud %>%inner_join(get_sunsentiments("sunscReen"), by = "word")
%>%group_by(brandName, sentiment) %>%count()

sentiment_normalized_perbrand_sun <- sentiment_by_brand_sun %>%group_by(brandName) %>%mutate(norm = n / sum(n))

plot_sentiment_pagenormalized_sun <- function(p) {ggplot(sentiment_normalized_perbrand_sun,aes(x = reorder(brandName,-n),y = norm,fill =
sentiment)) +geom_bar(stat = "identity") +ggforce::facet_wrap_paginate(facets = ~ sentiment,nrow = 3,ncol = 4,page = p) +theme_bw()
+theme(axis.text.x = element_text(angle = 90, hjust = 1)) +labs(x = "", y = "Frequency", fill = "Sentiment")}

plot_sentiment_pagenormalized_sun(1) %>% plot()

sunscreen_cleaned_for_wordcloud_product <- sunscreen %>%dplyr::mutate(review2 = review) %>%dplyr::group_by(productName) %>%

as.tibble() %>%tidytext::unnest_tokens(word, review) %>%anti_join(stop_words, by = "word") %>%anti_join(my_stop_words, by = "word")
%>%dplyr::filter(is.na(as.numeric(word)))

sentiment_by_brand_sun_product <-sunscreen_cleaned_for_wordcloud_product %>%inner_join(sunscReen::get_sunsentiments("sunscReen"), by =
"word") %>%group_by(productName, sentiment) %>%count()

sentiment_normalized_perbrand_sun_product <-sentiment_by_brand_sun_product %>%group_by(productName) %>%mutate(norm = n / sum(n))

plot_sentiment_pagenormalized_sun_product <- function(p) {ggplot(sentiment_normalized_perbrand_sun_product,aes(x = reorder(productName,-
n),y = norm,fill = sentiment)) +geom_bar(stat = "identity") +ggforce::facet_wrap_paginate(facets = ~ sentiment,nrow= 3,ncol = 4,page = p)
+theme_bw() +theme(axis.text.x = element_text(angle = 90, hjust = 1)) +labs(x = "", y = "frequency", fill = "Sentiment")}

plot_sentiment_pagenormalized_sun_product(1) %>% plot()

sunscreen_nostopword_corpus <- sunscreen %>%  mutate(review2 = review) %>%  as.tibble()

sunscreen_corpus <- corpus(sunscreen_nostopword_corpus$review) %>% quanteda::tokens(what = "word", remove_numbers =
TRUE,remove_punct = TRUE,remove_separators = TRUE,remove_twitter = TRUE, remove_hyphens = TRUE,remove_url = TRUE, ngrams = 1L,
skip = 0L,concatenator = "_",verbose = quanteda_options("verbose"),include_docvars = TRUE ) %>% tokens_select(pattern = stopwords('en'),
selection = 'remove')

dfmat <- dfm(sunscreen_corpus,remove_punct = TRUE,remove = my_stop_words$word)

 (tstat2 <-textstat_simil(dfmat, method = "cosine", margin = "documents"))

sum(dfmat[1, ] * dfmat[2, ]) / sqrt(sum(dfmat[1, ] ^ 2) * sum(dfmat[2, ] ^2))
```

```
clustering <-hclust(as.dist(1 - tstat2))

cluster.assignments<-cutree(clustering, k=20)

dtm <- convert(dfmat, to = "topicmodels")

lda <- LDA(dtm, k = 10) # build 10 topics

terms(lda, 5) topics(lda, 5)  lda@beta[, 1:10]  beta <- lda@beta[, 1:10] gam <- lda@gamma mlda <- gam %*% beta beta.td <- tidy(lda, matrix =
"beta")

filter(beta.td, topic == 1) ## all for topic 1

beta.top.terms <- beta.td %>%group_by(topic) %>% top_n(10, beta) %>% ungroup() %>% arrange(topic,-beta)

beta.top.terms %>%mutate(term = reorder_within(term, beta, topic)) %>%ggplot(aes(term, beta, fill = factor(topic))) +geom_col(show.legend =
FALSE) +facet_wrap( ~ topic, scales = "free") +  coord_flip() +  scale_x_reordered() +theme_minimal() +theme(text = element_text(size =
8),axis.text.x = element_text(angle = 90))

gamma.td <- tidy(lda, matrix = "gamma")

gettabletopicLDA <- function(gamma, i) {topic <- gamma %>% filter(topic == i)

  topic <- topic[order(-topic$gamma), ] %>% head()}

for (i in c(1:10)) {  gettabletopicLDA(gamma.td, i) %>%   kable() %>%    kable_styling(  bootstrap_options = c("striped", "hover", "condensed"),
full_width = FALSE  ) %>% print()}

augment(lda) %>% head() %>%kable() %>%  kable_styling(    bootstrap_options = c("striped", "hover", "condensed"),    full_width = FALSE  )
%>% print()

sunscreen.fcm <- fcm(  sunscreen_corpus,  context = "window",  count = "weighted",  weights = 1 / (1:5),  tri = TRUE)

get_similar <- function(w, df, k = 10) { if (is.character(w)) {    v <- as.numeric(df[w, ])  } else {    v <- as.numeric(w)  }

  nv <- sqrt(sum(v ^ 2))   m <- as.matrix(df)   rownorms <- sqrt(rowSums(m ^ 2))   sims <- m %*% v / (nv * rownorms)   names(sims) <-
rownames(df)

  return(head(sort(sims, decreasing = TRUE), n = k))}

glove <-  GlobalVectors$new(    word_vectors_size = 50,    vocabulary = featnames(sunscreen.fcm),    x_max = 10  )

sunscreen.glove <-  as.dfm(fit_transform(sunscreen.fcm, glove, n_iter = 100) + t(glove$components))

get_similar(sunscreen.glove["lotion", ] - sunscreen.glove["liquid", ] + sunscreen.glove["cream", ],   sunscreen.glove)

top50 <- names(get_similar("grail", sunscreen.glove, 50))[2:50]

distances <- textstat_dist(sunscreen.glove[top50, ])

clustering <- hclust(as.dist(distances))

cluster.assignments = cutree(clustering, k = 5)

pca <- prcomp(sunscreen.glove[top50,])

text(pca$x[,1:2], labels = rownames(pca$x), cex=0.5)

word_vectors <- glove$components

sun2.doc <- matrix(nr=nrow(word_vectors), nc=length(sunscreen_corpus))

colnames(sun2.doc) <- names(sunscreen_corpus)

for (i in 1:length(sunscreen_corpus)){  sun2.doc[,i] <- apply(word_vectors[,sunscreen_corpus[[i]], drop = F], 1, mean)}

sun2.doc <- as.matrix(sun2.doc) sun2.doc <- t(sun2.doc) sun2.doc <- as.data.frame(sun2.doc)  sun2.doc <- as.data.frame(sun2.doc) %>% mutate(text
= 1:nrow(sun2.doc))

sentiment_review <- sunscreen_cleaned_for_wordcloud %>%  inner_join(get_sunsentiments("sunscReen"), by = "word") %>%  group_by(reviewId,
sentiment) %>%  count()

positive <- c("comfort", "joy", "trust")

negative <- c("anger", "disgust", "fear", "sadness", "discomfort")

sentiment_review <- sentiment_review %>%  mutate(senplus = ifelse(sentiment %in% positive, n, 0)) %>% mutate(senmoins = ifelse(sentiment
%in% negative,-n, 0))

sentiment_per_text <-  sentiment_review %>% select(reviewId, senmoins, senplus) %>%  group_by(reviewId) %>%  summarize(sentimenttext =
sum(senmoins + senplus))

saveRDS(sentiment_per_text, file="../data/sentiment_per_text.RData"
brands <- c("australian", "biore", "bioré", "cerave", "clinique", "roche", "posay", "neutrogena", "paula")

brands_dummy <- data.frame(matrix(NA, nrow = nrow(sunscreen), ncol = length(brands)))

for(i in 1:length(brands)){  brands_dummy[,i] <- ifelse(str_detect(sunscreen$review, brands[i]),1,0)  names(brands_dummy)[i] <- paste("d_", brands[i])}

sunscreen.numbered <- sunscreen %>% cbind(brands_dummy) %>%  mutate(text = 1:nrow(sunscreen))

mlda.numbered <- mlda %>% as.data.frame() %>%  mutate(text = 1:nrow(mlda)) %>%   rename_at(vars(starts_with("V")),     funs(str_replace(., "V", "G")))

sun2.mlda <- inner_join(sun2.doc, mlda.numbered, by = "text")

sunscreen.sl.rating <- inner_join(sunscreen.numbered, sun2.mlda, by = "text") %>%  mutate(nwords = sapply(strsplit(review, " "), length)) %>% select(helpful, reviewId, votes, skinType, skinTone, brandId, starts_with("V"),starts_with("G"), starts_with("d_"),nwords, rating) %>% mutate(rating = as.factor(rating))

sunscreen.sl.rating <- inner_join(sentiment_per_text,sunscreen.sl.rating, by = "reviewId") %>%  select(-reviewId)

sunscreen.sl.brand <- inner_join(sunscreen.numbered, sun2.mlda, by = "text") %>%   mutate(nwords = sapply(strsplit(review, " "), length)) %>% select(reviewId, sentiments,  starts_with("V"),starts_with("G"),starts_with("d_"),nwords, brandId) %>%   mutate(brandId = as.factor(brandId))

sunscreen.sl.brand <- inner_join(sentiment_per_text,sunscreen.sl.brand, by = "reviewId") %>%   select(-reviewId)

## SUPERVISED LEARNING

index_b<-createDataPartition(sunscreen.sl.brand$brandId,p=0.75,list=F)

train.data  <- sunscreen.sl.brand[index_b,]

trainClasses <- sunscreen.sl.brand[index_b, ncol(train.data)]

test.data <- sunscreen.sl.brand[-index_b, ]

testClasses <- sunscreen.sl.brand[-index_b, ncol(test.data)]

tr <- trainControl(method = "repeatedcv", number = 10, repeats = 5)

system.time(rf_100_brand <- caret::train(brandId ~ ., data = train.data, method = "rf", ntree = 100,preProcess = "range", trControl = tr))

cm.rf_100_brand <- confusionMatrix(predict(rf_100_brand, test.data[,-ncol(test.data)]), test.data$brandId)

system.time(rf_500_brand <- caret::train(brandId ~ ., data = train.data, method = "rf", ntree = 500,preProcess = "range", trControl = tr))

cm.rf_500_brand <- confusionMatrix(predict(rf_500_brand, test.data[,-ncol(test.data)]), test.data$brandId)

system.time(rf_1000_brand <- caret::train(brandId ~ ., data = train.data, method = "rf", ntree = 1000,preProcess = "range", trControl = tr))

cm.rf_1000_brand <- confusionMatrix(predict(rf_1000_brand, test.data[,-ncol(test.data)]), test.data$brandId)

system.time(rf_1500_brand <- caret::train(brandId ~ ., data = train.data, method = "rf", ntree = 1500,preProcess = "range", trControl = tr))

cm.rf_1500_brand <- confusionMatrix(predict(rf_1500_brand, test.data[,-ncol(test.data)]), test.data$brandId)

system.time(rf_2000_brand <- caret::train(brandId ~ ., data = train.data, method = "rf", ntree = 2000,preProcess = "range", trControl = tr))

cm.rf_2000_brand <- confusionMatrix(predict(rf_2000_brand, test.data[,-ncol(test.data)]), test.data$brandId)

resamps <- resamples(list("RF with 100 trees" = rf_100_brand,    "RF with 500 trees" = rf_500_brand,  "RF with 1000 trees" = rf_1000_brand,  "RF with 1500 trees" = rf_1500_brand,   "RF with 2000 trees" = rf_2000_brand))

theme1 <- trellis.par.get()   theme1$plot.symbol$col = rgb(.2, .2, .2, .4)    theme1$plot.symbol$pch = 16  theme1$plot.line$col = rgb(1, 0, 0, .7) theme1$plot.line$lwd <- 2  trellis.par.set(theme1) bwplot(resamps, layout = c(2, 1))

Train_Features <- data.matrix(sunscreen.sl.brand[index_b,-ncol(sunscreen.sl.brand)])

Test_Features <- data.matrix(sunscreen.sl.brand[-index_b,-ncol(sunscreen.sl.brand)])

Train_Labels <- keras.dum[index_b,]

Test_Labels <- keras.dum[-index_b,]

model <- keras_model_sequential()

model %>%  layer_dense(units = 72, activation = "relu") %>% layer_dense(units = 40, activation = "relu") %>% layer_dense(units = 20, activation = "relu") %>%layer_dense(units = 15, activation = "relu") %>%layer_dense(units = 7, activation = "softmax")model %>% compile(loss = "categorical_crossentropy",optimizer = 'rmsprop',metrics = c('accuracy'))

history <- model %>% keras::fit(Train_Features,Train_Labels,validation_split = 0.15, epochs = 200,batch_size = 100,shuffle = T  )

score <- model %>% evaluate(Test_Features, Test_Labels, batch_size = 25)
```