Text Analytics HW1

Marshall Tuck

10/26/2020

Preprocessing (20 pts)

a. (1 pt) Load in the "Sharktankpitchesdeals.csv" dataset in UTF-8 format.

```
#read file in utf-8 format
sharktank <- read.csv(file = "sharktankpitchesdeals.csv", encoding = "utf-8")
```

b. (2 pts) Change all text to lower-case

```
#Change text to lower case
sharktankpitches<- tolower(sharktank[,3])
```

c. Remove...

d. (2 pts) Stopwords

```
#Remove stop words
sharktankpitches.processed <- removeWords(sharktankpitches, stopwords())
```

ii. (2 pts) All numbers and punctuation

```
#Remove non-word characters
sharktankpitches.processed <- gsub("\\W", " ", sharktankpitches.processed)

#Remove numbers
sharktankpitches.processed<- gsub("\\d"," ",sharktankpitches.processed )
```

iii. (2 pts) Singular characters (characters like s and t)

```
#Remove singular characters surrounded by a space
sharktankpitches.processed<- gsub(" *\\b[[:alpha:]]{1}\\b *", " ", sharktankpitches.processed)
```

iv. (2 pts) All non-English characters (characters like ú, á, é, ñ)

```
#Remove non a-z characters
sharktankpitches.processed <- gsub("[^a-zA-Z]", " ", sharktankpitches.processed)
```

d. (2 pts) Remove excess whitespace and condense each character string so there is only 1 space between words and no whitespace at the beginning or end of a string.

```
#Replace multiple spaces with single space
sharktankpitches.processed<-stripWhitespace(sharktankpitches.processed)

#Remove leading and trailing white spaces
sharktankpitches.processed<-str_squish(sharktankpitches.processed)
```

e. (3 pts) Create a table of all the words, their counts, and their proportion. Sort the table so that the words with the highest frequency are first. Present the first 20 rows of the table.

```
#Split each string of words into a list of separate words
sharktank.tokenized<- str_split(sharktankpitches.processed, " ")

#Combine all words from all strings into one vector
sharktank.words<- unlist(sharktank.tokenized)

#Find unique words (minus stop words) in sharktank.words
sharktank.words.unique<- unique(sharktank.words)

#Create output vector where each of 4362 unique words are counted in all pitches
tf.sharktank.words<- sapply(1:length(sharktank.words.unique),
                            function (i){
                              sum(sharktank.words==sharktank.words.unique[i])
                              }
                            )

#Present table where each unique word is summarized with count and proportion
tf.sharktank.words.table <-
  data.frame("word" = sharktank.words.unique, "freq" = tf.sharktank.words)%>%
  as_tibble()%>%
  mutate(prop=freq/length(sharktank.words))%>%
  arrange(-freq)%>%
  head(20)%>%
  kable(caption="Shark Tank Pitch Words")%>%
  kable_styling(latex_options = "HOLD_position")

tf.sharktank.words.table
```

Table 1: Shark Tank Pitch Words

word	freq	prop
can	103	0.0077067
service	89	0.0066592
line	76	0.0056865
made	66	0.0049383
children	60	0.0044893
device	59	0.0044145
designed	54	0.0040404
like	48	0.0035915
products	46	0.0034418
app	42	0.0031425
clothing	41	0.0030677
kids	38	0.0028432
use	37	0.0027684
offers	37	0.0027684
high	37	0.0027684
perfect	37	0.0027684
company	36	0.0026936
allows	35	0.0026188
free	35	0.0026188
natural	35	0.0026188

f. (4 pts) Remove 2 words within the 20 most frequent words that you believe does not add any information. This is subjective, so just make a decision and justify why you remove those two words.

I choose to remove "can" and "like" from my included words, because there is little value to them in topic modeling investment pitches. "Can" is used frequently as an active word in advertising - my product "can" do something. "Like" is used in two contexts - similarity, and in affection. All companies that go on Shark Tank looking for money will be confident in their product's abilities, meaning that the words "like" and "can" will not differentiate between products.

```
#Remove "can" and "like"
sharktankpitches.processed <- removeWords(sharktankpitches.processed, c("can",
                                                                        "like"))

#Reprocess white spaces and squishing
#Replace multiple spaces with single space
sharktankpitches.processed<-stripWhitespace(sharktankpitches.processed)

#Remove leading and trailing white spaces
sharktankpitches.processed<-str_squish(sharktankpitches.processed)
```

```
#Recreate unique words, with word freq and proportion
#Split each string of words into a list of separate words
sharktank.tokenized<- str_split(sharktankpitches.processed, " ")

#Combine all words from all strings into one vector
sharktank.words<- unlist(sharktank.tokenized)

#Find unique words (minus stop words) in sharktank.words
```

```
sharktank.words.unique<- unique(sharktank.words)

#Create output vector where each of 4362 unique words are counted in all pitches
tf.sharktank.words<- sapply(1:length(sharktank.words.unique),
                            function (i){
                              sum(sharktank.words==sharktank.words.unique[i])
                              }
                            )

#Present table where each unique word is summarized with count and proportion
tf.sharktank.words.table <-
  data.frame("word" = sharktank.words.unique, "freq" = tf.sharktank.words)%>%
  as_tibble()%>%
  mutate(prop=freq/length(sharktank.words))%>%
  arrange(-freq)%>%
  head(20)%>%
  kable(caption="Shark Tank Words Minus 'Can' and 'Like'")%>%
  kable_styling(latex_options = "HOLD_position")
```

(20 pts) N-grams

a. (7 pts) From the preprocessed text in 1., create a table of all 2-gram combinations of words, in the same format as 1e. Present this table with the 20 most frequent 2-gram combinations.

```
#Create one string of all shark tank words
all.sharktank.words<- concatenate(sharktank.words)

#Present in ngram of 2
two.ngram.sharktank <- ngram(all.sharktank.words, 2, sep=" ")

#Output count and proportion
kable(head(get.phrasetable(two.ngram.sharktank), 20), caption="Shark Tank 2 Grams")%>%
  kable_styling(latex_options = "HOLD_position")
```

Table 2: Shark Tank 2 Grams

ngrams	freq	prop
shark tank	15	0.0011352
high quality	13	0.0009839
ice cream	13	0.0009839
you re	12	0.0009082
gluten free	11	0.0008325
mail order	11	0.0008325
eco friendly	11	0.0008325
specially designed	9	0.0006811
delivery service	8	0.0006055
clothing line	8	0.0006055
award winning	7	0.0005298
subscription service	7	0.0005298
rental service	7	0.0005298
belt buckle	7	0.0005298
real estate	7	0.0005298
pitches shark	6	0.0004541
app helps	6	0.0004541
device uses	6	0.0004541
natural ingredients	6	0.0004541
wine glass	6	0.0004541

b. (7 pts) Do the same as 2a), but with 3-gram combinations.

```
#Present in ngram of 3
three.ngram.sharktank <- ngram(all.sharktank.words, 3, sep=" ")

#Output count and proportion
kable(head(get.phrasetable(three.ngram.sharktank), 20), caption="Shark Tank 3 Grams")%>%
  kable_styling(latex_options = "HOLD_position")
```

Table 3: Shark Tank 3 Grams

ngrams	freq	prop
pitches shark tank	6	0.0004541
single serving wine	5	0.0003784
echo valley meats	4	0.0003028
made recycled materials	3	0.0002271
keep wine good	3	0.0002271
proprietary container brand	3	0.0002271
non gmo vegan	3	0.0002271
brand single serving	3	0.0002271
vegan gluten free	3	0.0002271
way keep wine	3	0.0002271
wine good longer	3	0.0002271
shark tank pitch	3	0.0002271
container brand single	3	0.0002271
serving wine glass	3	0.0002271
subscription box service	3	0.0002271
beef jerky made	3	0.0002271
service creates photobooks	2	0.0001514
clothing crawling babies	2	0.0001514
way take care	2	0.0001514
movie idea involving	2	0.0001514

c. (6 pts) Do you notice any differences between the 1, 2, and 3-gram combinations? Which words come up most frequently?

The first difference is that the frequency of 1 gram far exceeds the frequency of 2grams or 3grams. This is to be expected - as you add more and more words to a combination, that combination becomes less frequent in a conversation.

For 1 grams, it is most common to see what the product is. It's a service, device, or product, probably designed for children, that provides a good way (perfect, offers, high, natural) to do things. It is easy to see a narrative in 1 grams.

2grams offer less of an overall picture of all products, and more descriptors of individual products. For instance, these popular products might be gluten free, high quality, or eco friendly (all 2grams due to the removal of a hyphen in original words). Also included are non-value words like Shark Tank and Pitches Shark. The proportion of 20 most used 2grams is about 20% that of 1gram.

3grams offer even less insight into all products overall, and is only topical definition words on specific products. The top 20 3grams occur about half the time as the top twenty 2grams, so the 3grams tend to be very product specific, and have little to do with the most frequent 1grams (involving children's products). 3grams sometimes are exact matches (Echo Valley Meats) and sometimes are a product of removing stop words, single characters, etc. (made recycled materials, way keep wine).

It appears that as you increase your n in ngrams past 1 or 2, you lose insight into what the most popular shark tank topics are.

(20 pts) TF-IDF

a. (8 pts) Calculate the TF-IDF score for all words in each Shark pitch deal.

```
#Function that processes vector of words, and computes summary statistics on the word within each revie

tf_document <- lapply(1:length(sharktank.tokenized),
       function(i){
         unique_words <- unique(sharktank.tokenized[[i]])

         tf_review <- sapply(1:length(unique_words),
                             function(j){sum(sharktank.tokenized[[i]] ==
                                               unique_words[j])})

         tf_single_review <- (data.frame("review_num" = rep(i, length(tf_review)),
                                         "word" = unique_words,
                                         "count" = tf_review,
                                         "total_words" =
                                           rep(length(sharktank.tokenized[[i]]),
                                               length(tf_review))))
       })

#Bind the outputs of the function
tf_by_sharktank.pitch<- do.call("rbind", tf_document)
```

b. (4 pts) Create a table with the word, count, total words (from the word's respect pitch deal), TF (term frequency) score, IDF score, and TF-IDF score.

```
#Make table combining term freq, word, review number, and count
tfidf_sharktank <- bind_tf_idf(as_tibble(tf_by_sharktank.pitch), word, review_num, count)
```

c. (2 pts) Present the table sorted by TF-IDF score (highest TF-IDF score is the first row) with the first 20 rows.

```
#Present top 20 tf-idf scores
tfidf_sharktank%>%
  arrange(-tf_idf)%>%
  head(20)%>%
  kable(caption="Shark Tank Top 20 TF-IDF")%>%
  kable_styling(latex_options = "HOLD_position")
```

Table 4: Shark Tank Top 20 TF-IDF

review_num	word	count	total_words	tf	idf	tf_idf
355	peanuts	2	4	0.5	6.559615	3.279808
386	moccasins	2	4	0.5	6.559615	3.279808
437	breathalyzer	2	4	0.5	6.559615	3.279808
440	mango	2	4	0.5	6.559615	3.279808
441	pickles	2	4	0.5	6.559615	3.279808
448	translator	2	4	0.5	6.559615	3.279808
451	photobooths	2	4	0.5	6.559615	3.279808
461	gamers	2	4	0.5	6.559615	3.279808
485	muffins	1	2	0.5	6.559615	3.279808
522	slips	2	4	0.5	6.559615	3.279808
526	koozies	2	4	0.5	6.559615	3.279808
590	teas	2	4	0.5	6.559615	3.279808
705	pie	1	2	0.5	6.559615	3.279808
343	handmade	2	4	0.5	5.866468	2.933234
343	bowties	2	4	0.5	5.866468	2.933234
440	preserves	2	4	0.5	5.866468	2.933234
444	cake	2	4	0.5	5.866468	2.933234
461	furniture	2	4	0.5	5.866468	2.933234
522	modernized	2	4	0.5	5.866468	2.933234
553	band	2	4	0.5	5.866468	2.933234

d. (6 pts) Why is the TF-IDF score useful compared to using only word counts or n-grams?

The 'TF' portion in TF-IDF is what drives word counts or n-grams. The 'IDF' portion in TF-IDF is the inverse document frequency, which adds more insight than TF alone.

IDF measures the 'specialness' of a word in the overall corpus. Therefore, given two terms with the same term frequency in a document, a term that occurs in fewer documents in the overall corpus will have a higher TF-IDF score.

It is useful because it gives us a measure of the most common words weighted by the uniqueness of a word in the corpus. These two together show us the importantness of a word.

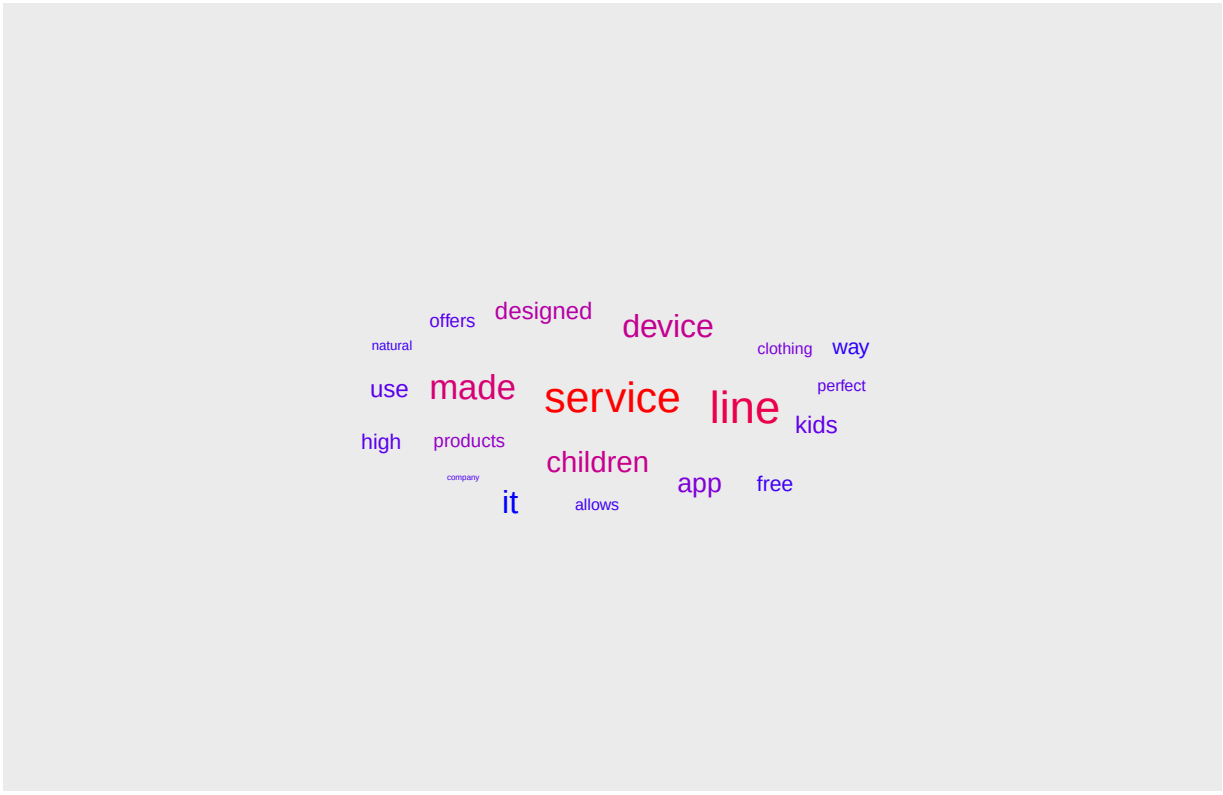
(20 pts) Visualizations, for these 2 questions, feel free to play around with color combinations that look appealing

a. Create a word cloud of...
b. (2 pts) Word counts of the preprocessed information from 1e.

```
#GGplot word cloud by 1 grams
sharktank.tf.wordcloud<-
  data.frame("word" = sharktank.words.unique, "freq" = tf.sharktank.words)%>%
  as_tibble()%>%
  mutate(prop=freq/length(sharktank.words))%>%
  arrange(-freq)%>%
  head(20)

ggplot(sharktank.tf.wordcloud,
```

```
       aes(label = word, size = freq, color = freq)) +
            geom_text_wordcloud_area() +
            scale_color_gradient(low = "blue", high = "red")+
            labs(title = "Term Frequency: Shark Tank Words")
```

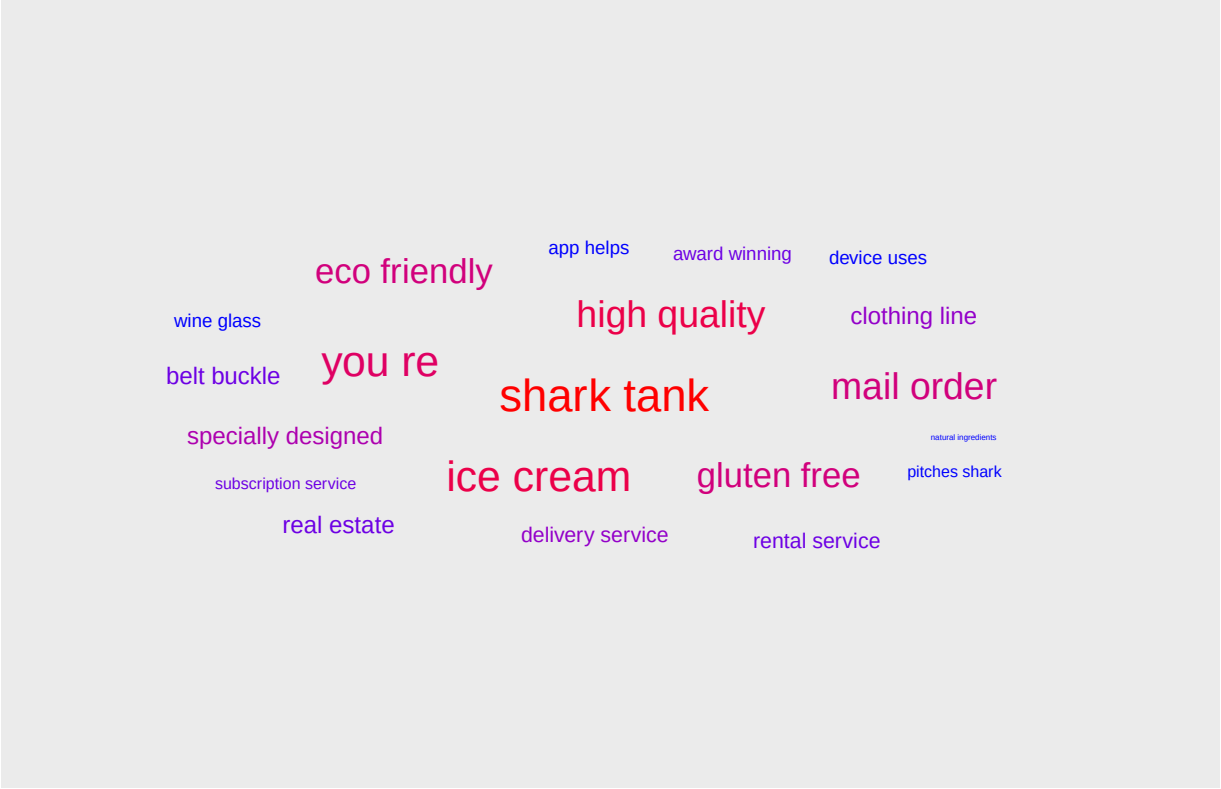


ii. (2 pts) 2-grams calculated in 2a.

```
# GGplot word cloud 2 gram
sharktank.2gram<-head(get.phrasetable(two.ngram.sharktank),20)

ggplot(sharktank.2gram,
       aes(label = ngrams, size = freq, color = freq)) +
            geom_text_wordcloud_area() +
            scale_color_gradient(low = "blue", high = "red")+
            labs(title = "Term Frequency: Shark Tank 2-grams")
```

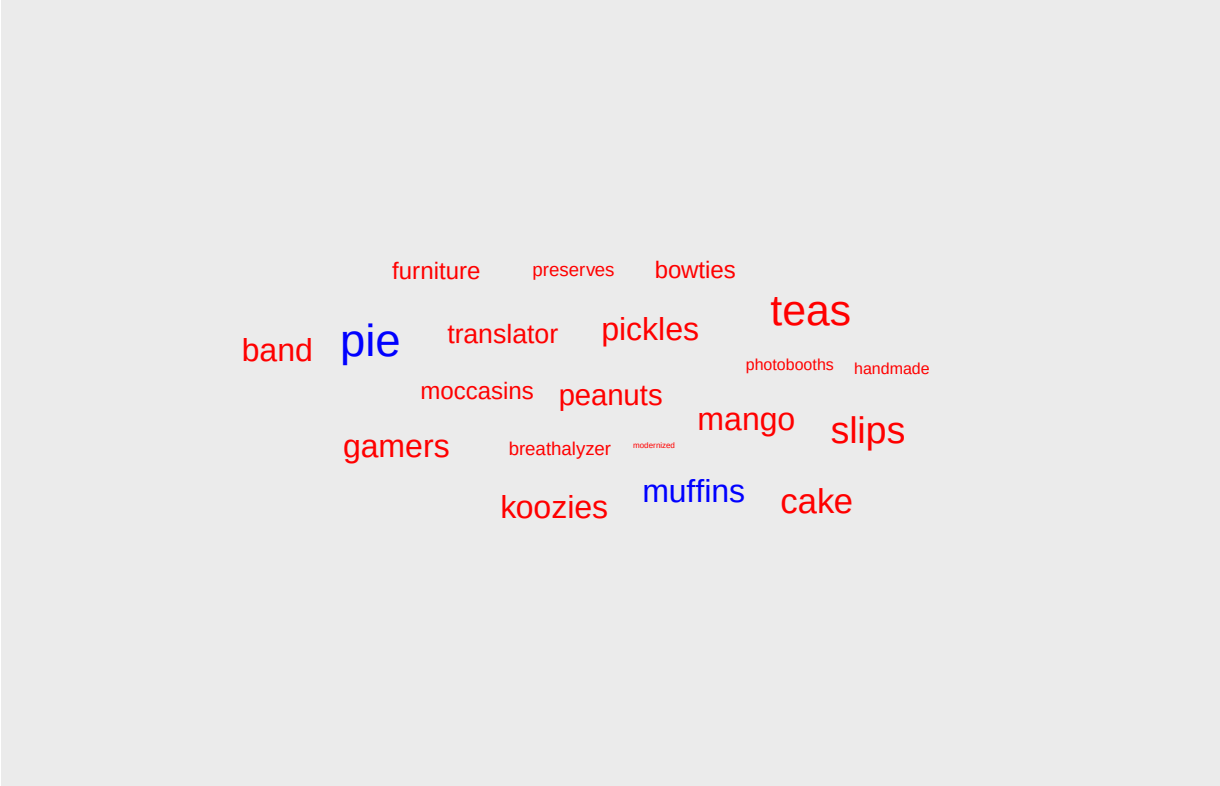
Term Frequency: Shark Tank 2−grams



iii. (2 pts) TF-IDF scored words calculated in 3b.

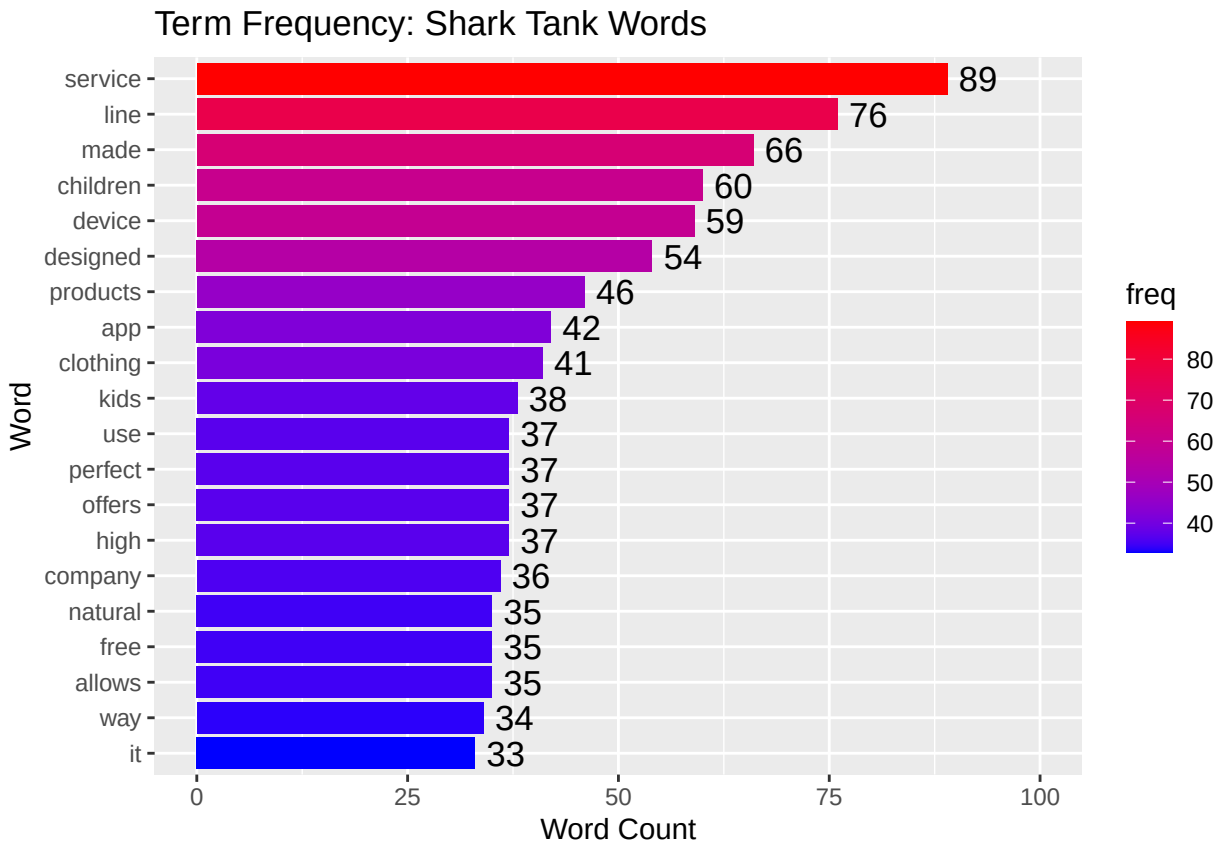
```
#GGplot word cloud tf-idf
tfidf_sharktank%>%
  arrange(-tf_idf)%>%
  head(20)%>%
  ggplot(aes(label = word, size = tf_idf, color = count)) +
           geom_text_wordcloud_area() +
           scale_color_gradient(low = "blue", high = "red")+
           labs(title = "TF-IDF: Shark Tank Words")
```

TF−IDF: Shark Tank Words



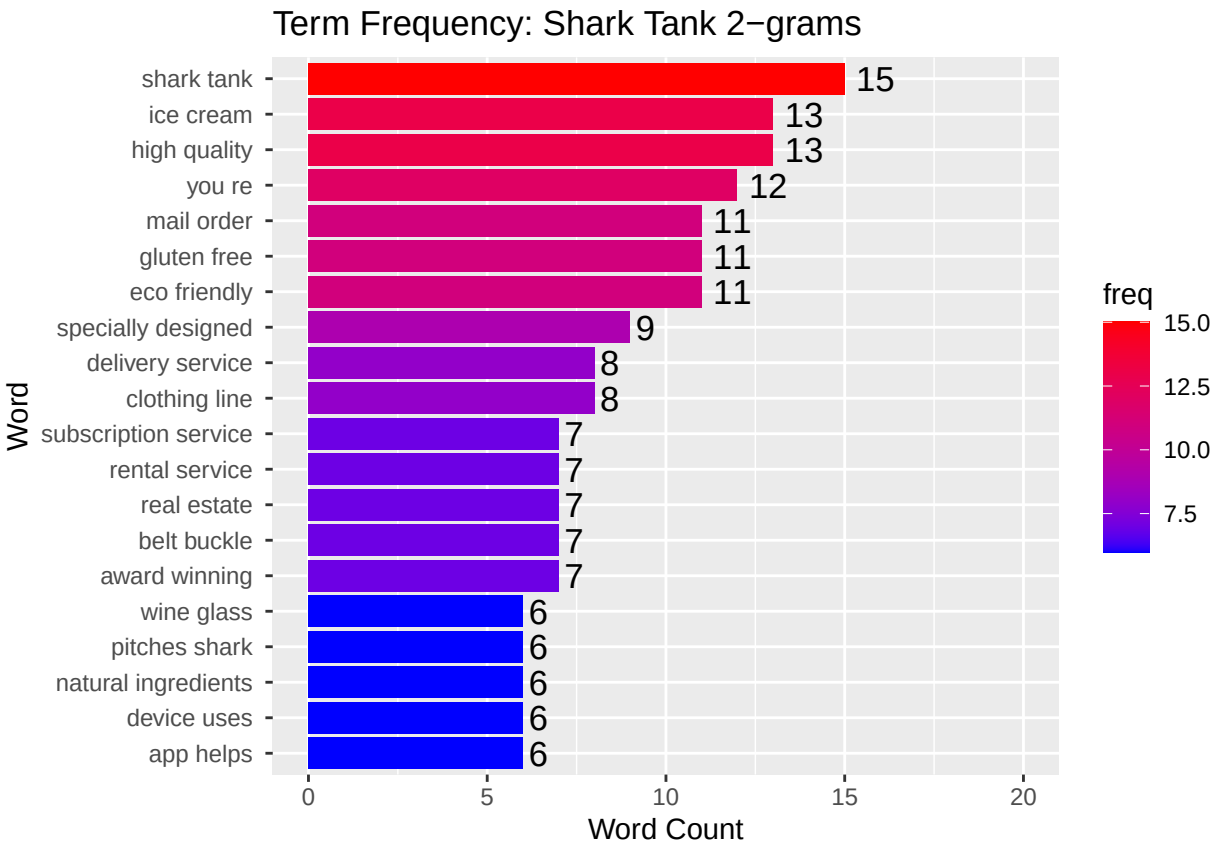
b. Create a bar plot of...
c. (2 pts) Word counts of the preprocessed information from 1e.

```
#Barplot 1 gram
ggplot(sharktank.tf.wordcloud, aes(x = reorder(word, freq), y = freq)) +
            geom_bar(stat = "identity", aes(fill = freq)) +
            labs(title = "Term Frequency: Shark Tank Words", x = "Word",
                 y = "Word Count") +
            coord_flip() +
            geom_text(stat = "identity", aes(label = freq), hjust = -0.3, size = 4.5) +
            scale_fill_gradient(low = "blue", high = "red") +
            expand_limits(y = 100)
```



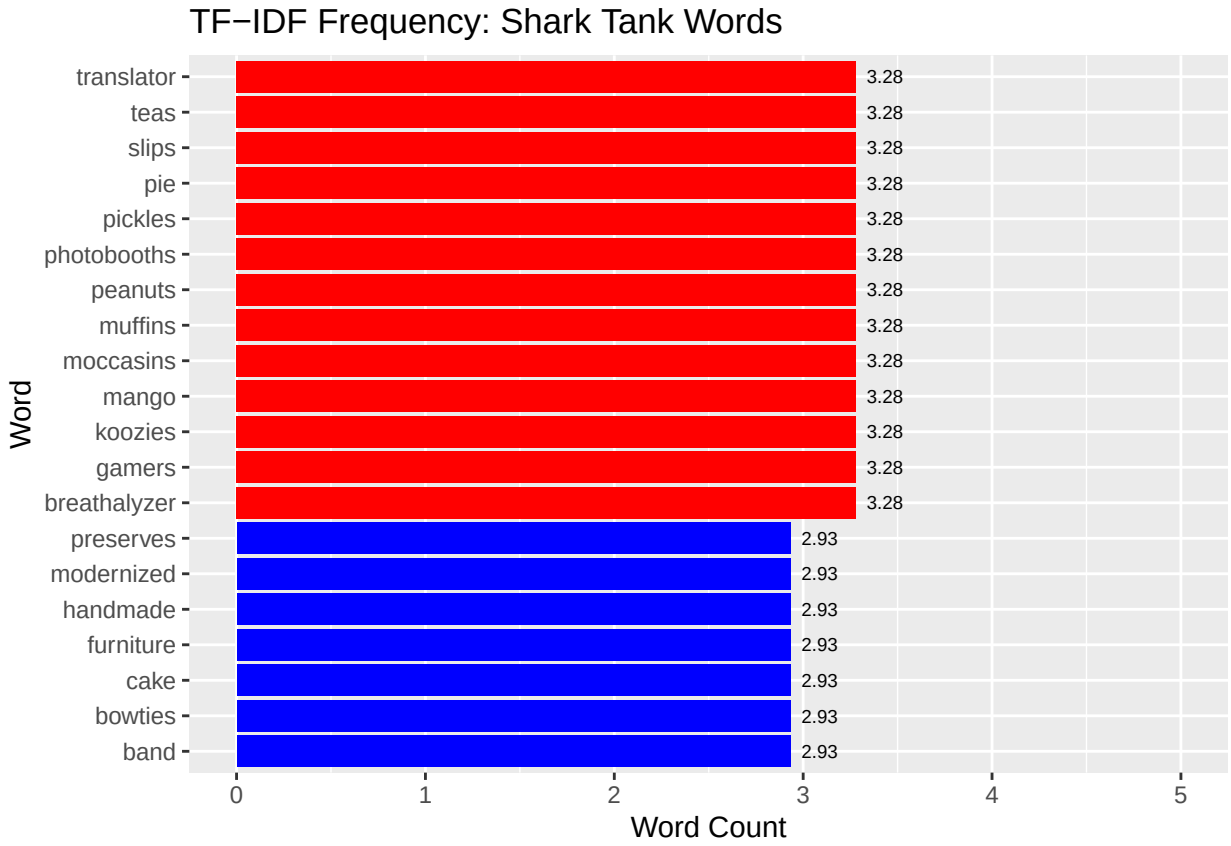
ii. (2 pts) 2-grams calculated in 2a.

```
#Barplot 2 gram
ggplot(sharktank.2gram, aes(x = reorder(ngrams, freq), y = freq)) +
            geom_bar(stat = "identity", aes(fill = freq)) +
            labs(title = "Term Frequency: Shark Tank 2-grams", x = "Word",
                 y = "Word Count") +
            coord_flip() +
            geom_text(stat = "identity", aes(label = freq), hjust = -0.3, size = 4.5) +
            scale_fill_gradient(low = "blue", high = "red") +
            expand_limits(y = 20)
```



iii. (2 pts) TF-IDF scored words calculated in 3b.

```
#Barplot tf-idf
tfidf_sharktank%>%
  arrange(-tf_idf)%>%
  head(20)%>%
  ggplot(aes(x = reorder(word, tf_idf), y = tf_idf)) +
            geom_bar(stat = "identity", aes(fill = tf_idf)) +
            labs(title = "TF-IDF Frequency: Shark Tank Words", x = "Word",
            y = "Word Count") +
            coord_flip() +
            geom_text(stat = "identity", aes(label = round(tf_idf,2)), hjust = -0.3, size = 2.5) +
            scale_fill_gradient(low = "blue", high = "red") +
            expand_limits(y = 5)+
  theme(legend.position = "none")
```



c. (8 pts) Do you notice any differences between any of the word counts, 2-grams, and TF-IDF counts? What information can you infer from all of the shark pitches through these bar plots and word clouds?

The most blatant observation is that as you link more words (going from 1 gram to 2 gram), the frequencies decrease understandably as it is harder to use a 2 gram in as many contexts as a 1 gram.

Also, I notice that the TF-IDF scores are all less than the word count scores - which makes sense, as the IDF portion scales down the value.

TF-IDF values are shared within the top 20 words - most are 3.28, and some are 2.93. This tells me that a bigger corpus may be helpful, or some other scaling mechanism, as these "tied" TF-IDF scores may be hiding useful information.

All together, we see that children products and services are common words, but not all that unique as showed by low TF-IDF scores. Our 2 grams are mainly descriptive phrases. And the top 20 TF-IDF terms are truly unique words associated with unique products - teas, pickles, photobooths, moccasins.

(20 pts) Accepted pitches vs rejected pitches

a. (4 pts) From the information obtained from 1d (before condensing all pitches into one character string), separate the pitches into 2 groups. One group for pitches that were accepted (Deal_Status = 1) and one group for pitches that were not accepted (Deal_Status = 0).

```
#Separate into accepted=yes and accepted=no
pitch.yes<- sharktank%>%
  cbind(sharktankpitches.processed)%>%
```

```
  filter(Deal_Status==1)%>%
  select(sharktankpitches.processed)

pitch.no<- sharktank%>%
  cbind(sharktankpitches.processed)%>%
  filter(Deal_Status==0)%>%
  select(sharktankpitches.processed)
```

b. (4 pts) Create a table for each group of pitches, similar to the one created in 1e.

```
#Split each string of words into a list of separate words
sharktank.tokenized.yes<- str_split(pitch.yes, " ")
```

```
## Warning in stri_split_regex(string, pattern, n = n, simplify = simplify, :
## argument is not an atomic vector; coercing
```

```
#Combine all words from all strings into one vector
sharktank.words.yes<- unlist(sharktank.tokenized.yes)

#Find unique words (minus stop words) in sharktank.words
sharktank.words.unique.yes<- unique(sharktank.words.yes)

#Create output vector where each of 4362 unique words are counted in all pitches
tf.sharktank.words.yes<- sapply(1:length(sharktank.words.unique.yes),
                            function (i){
                              sum(sharktank.words.yes==sharktank.words.unique.yes[i])
                              }
                            )

#Present table where each unique word is summarized with count and proportion
tf.sharktank.words.table.yes <-
  data.frame("word" = sharktank.words.unique.yes, "freq" = tf.sharktank.words.yes)%>%
  as_tibble()%>%
  mutate(prop=freq/length(sharktank.words.yes))%>%
  arrange(-freq)%>%
  head(20)

tf.sharktank.words.table.yes%>%
  kable(caption="Top 20 Words by Frequency in Accepted Deals")%>%
  kable_styling(latex_options = "HOLD_position")
```

Table 5: Top 20 Words by Frequency in Accepted Deals

word	freq	prop
designed	36	0.0045455
made	36	0.0045455
service	29	0.0036616
kids	28	0.0035354
perfect	27	0.0034091
children	27	0.0034091
free	26	0.0032828
allows	25	0.0031566
use	24	0.0030303
line	24	0.0030303
high	22	0.0027778
phone	22	0.0027778
also	22	0.0027778
offers	21	0.0026515
just	21	0.0026515
wine	20	0.0025253
it	20	0.0025253
way	20	0.0025253
device	19	0.0023990
app	19	0.0023990

```
#Split each string of words into a list of separate words
sharktank.tokenized.no<- str_split(pitch.no, " ")
```

```
## Warning in stri_split_regex(string, pattern, n = n, simplify = simplify, :
## argument is not an atomic vector; coercing
```

```
#Combine all words from all strings into one vector
sharktank.words.no<- unlist(sharktank.tokenized.no)

#Find unique words (minus stop words) in sharktank.words
sharktank.words.unique.no<- unique(sharktank.words.no)

#Create output vector where each of 4362 unique words are counted in all pitches
tf.sharktank.words.no<- sapply(1:length(sharktank.words.unique.no),
                               function (i){
                                 sum(sharktank.words.no==sharktank.words.unique.no[i])
                                 }
                               )

#Present table where each unique word is summarized with count and proportion
tf.sharktank.words.table.no <-
  data.frame("word" = sharktank.words.unique.no, "freq" = tf.sharktank.words.no)%>%
  as_tibble()%>%
  mutate(prop=freq/length(sharktank.words.no))%>%
  arrange(-freq)%>%
  head(20)

tf.sharktank.words.table.no%>%
```

```
  kable(caption="Top 20 Words by Frequency in NOT Accepted Deals")%>%
  kable_styling(latex_options = "HOLD_position")
```

Table 6: Top 20 Words by Frequency in NOT Accepted Deals

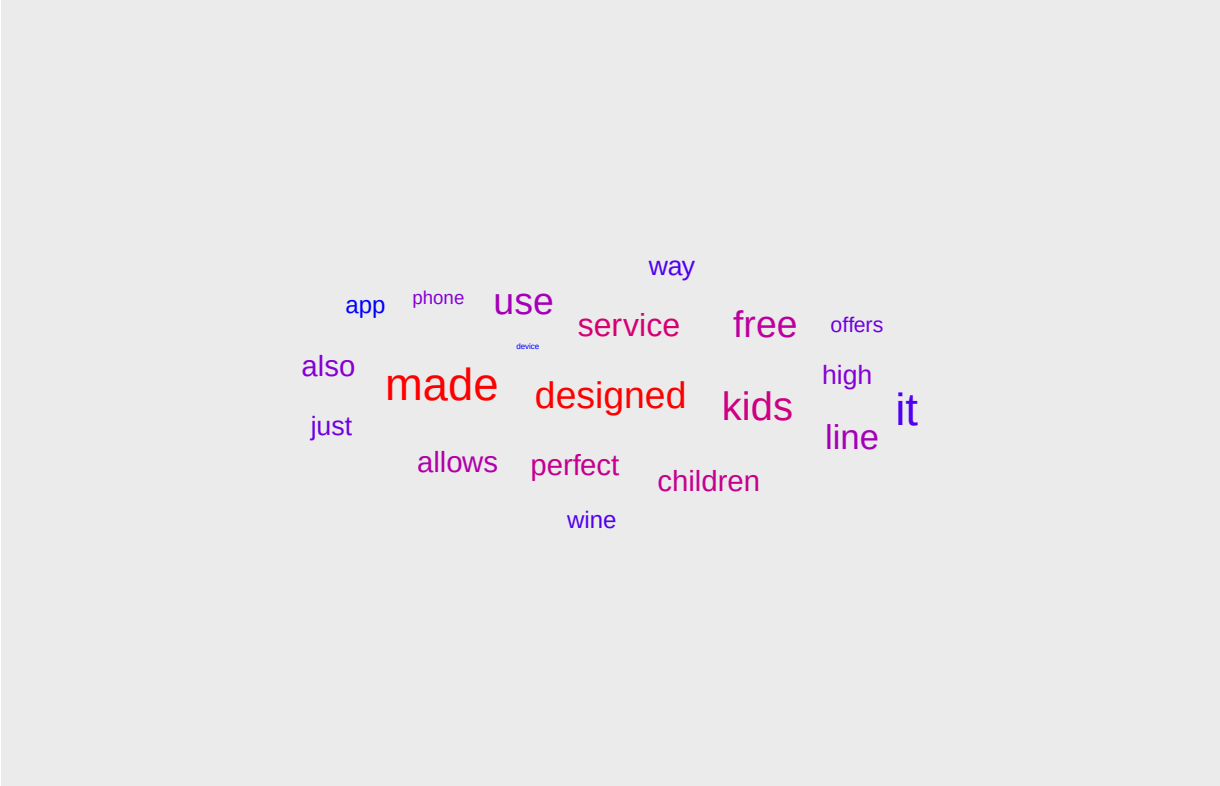
word	freq	prop
service	34	0.0064224
made	30	0.0056668
clothing	23	0.0043445
products	21	0.0039668
device	20	0.0037779
designed	18	0.0034001
uses	18	0.0034001
children	17	0.0032112
offers	16	0.0030223
ice	15	0.0028334
company	15	0.0028334
hair	14	0.0026445
help	14	0.0026445
quality	14	0.0026445
it	13	0.0024556
water	13	0.0024556
friendly	13	0.0024556
men	13	0.0024556
"line	13	0.0024556
line	13	0.0024556

c. (4 pts) Create a word cloud and bar plot for each group of pitches.

```
#GGplot word cloud by 1 grams

ggplot(tf.sharktank.words.table.yes,
       aes(label = word, size = freq, color = freq)) +
            geom_text_wordcloud_area() +
            scale_color_gradient(low = "blue", high = "red")+
            labs(title = "Term Frequency: Shark Tank Words for Accepted Deals")
```

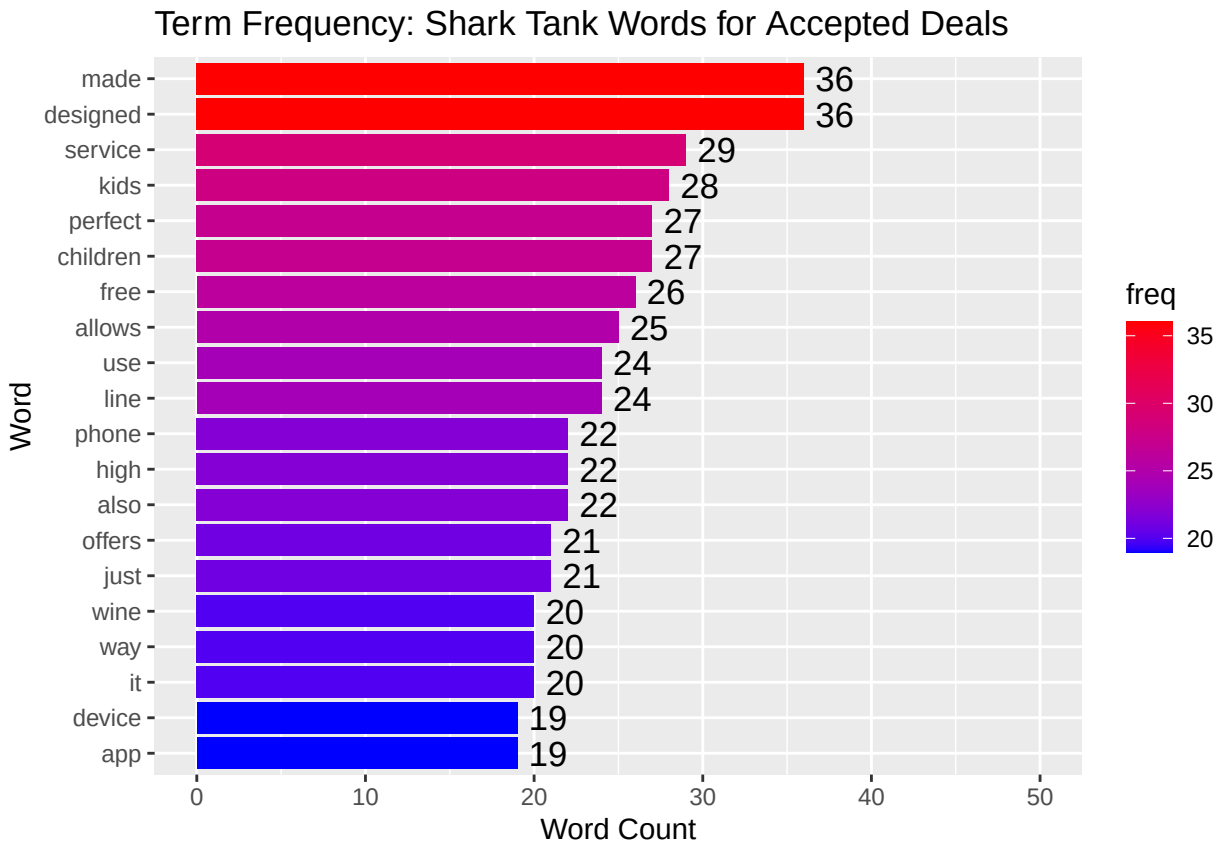
Term Frequency: Shark Tank Words for Accepted Deals



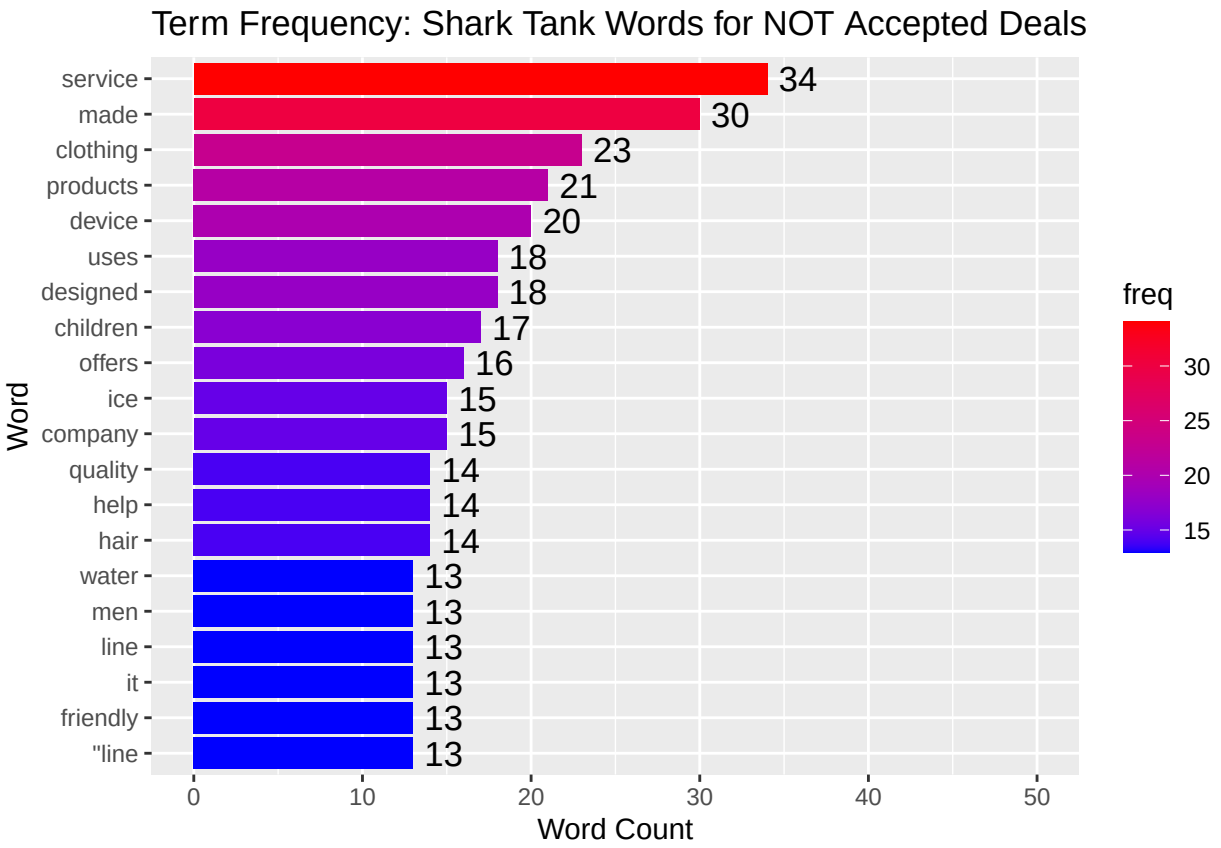

```
ggplot(tf.sharktank.words.table.no,
       aes(label = word, size = freq, color = freq)) +
            geom_text_wordcloud_area() +
            scale_color_gradient(low = "blue", high = "red")+
            labs(title = "Term Frequency: Shark Tank Words for NOT Accepted Deals")
```

Term Frequency: Shark Tank Words for NOT Accepted Deals


```
#GGplot bar graphs
ggplot(tf.sharktank.words.table.yes, aes(x = reorder(word, freq), y = freq)) +
            geom_bar(stat = "identity", aes(fill = freq)) +
            labs(title = "Term Frequency: Shark Tank Words for Accepted Deals", x = "Word", y = "Word C
            coord_flip() +
            geom_text(stat = "identity", aes(label = freq), hjust = -0.3, size = 4.5) +
            scale_fill_gradient(low = "blue", high = "red") +
            expand_limits(y = 50)
```




```
ggplot(tf.sharktank.words.table.no, aes(x = reorder(word, freq), y = freq)) +
            geom_bar(stat = "identity", aes(fill = freq)) +
            labs(title = "Term Frequency: Shark Tank Words for NOT Accepted Deals", x = "Word", y = "Wo
            coord_flip() +
            geom_text(stat = "identity", aes(label = freq), hjust = -0.3, size = 4.5) +
            scale_fill_gradient(low = "blue", high = "red") +
            expand_limits(y = 50)
```



d. (8 pts) Do you see any differences between the groups of pitches? What differentiates successful pitches from less successful ones? What would you suggest to a future entrepreneur to try to increase their chances of being approved (from a word usage standpoint)?

The first difference I see is that the 'no' bar chart contains more specific products, which we can now assume are considered bad products. For instance, stay away from any pitch around the words ice, hair, water, or men. In contrast, successful pitches tend to be about children, phones, apps, and wine.

What could be happening is the application of the product to a visual pitch "show and tell" - it is easy to bring children to a pitch, and get the sharks' interest. It is harder to show and tell water, and make that pitch interesting.

To a future entrepreneur, I would recommend bringing on a product that is tangible and easily explained. Also, I would suggest bringing interesting props to a demonstration, which could get the sharks' attention and make it more likely to get a deal.