Analyzing Amazon Food Reviews: Unveiling Consumer Insights

Gizem Güleli & Mehmet Tiryaki

449872 - 437988

2023-12-27

Overview

Delving into the intricate web of Amazon food reviews, we embarked on a journey to unravel customer sentiments, identify popular product categories, and uncover word associations that define the narrative. Our dataset, a co-purchasing network extracted from Amazon, comprised a staggering 262,111 nodes and 1,234,877 edges, offering a rich tapestry of interconnected insights.

The dataset, representing a co-purchasing network, was thoroughly preprocessed to ensure the integrity of the text data. The analysis involved examining word frequencies, bigrams, and skip-grams to unravel patterns and associations within the reviews.

Data

Amazon Food Reviews Dataset

The dataset represents an Amazon product co-purchasing network collected by crawling the Amazon website. It is based on the “Customers Who Bought This Item Also Bought” feature, forming directed edges between products frequently co-purchased. The data was collected in March 02, 2003.

Dataset Statistics:

Reviews from Oct 1999 - Oct 2012 568,454 reviews 256,059 users 74,258 products 260 users with > 50 reviews

```
data <- read.csv("Reviews.csv", stringsAsFactors = FALSE)

head(data)
```

```
##   Id  ProductId         UserId                      ProfileName
## 1  1 B001E4KFG0 A3SGXH7AUHU8GW                       delmartian
## 2  2 B00813GRG4 A1D87F6ZCVE5NK                           dll pa
## 3  3 B000LQOCH0  ABXLMWJIXXAIN Natalia Corres "Natalia Corres"
## 4  4 B000UA0QIQ A395BORC6FGVXV                             Karl
## 5  5 B006K2ZZ7K A1UQRSCLF8GW1T   Michael D. Bigham "M. Wassir"
## 6  6 B006K2ZZ7K  ADT0SRK1MGOEU                    Twoapennything
##   HelpfulnessNumerator HelpfulnessDenominator Score       Time
## 1                    1                       1     5 1303862400
## 2                    0                       0     1 1346976000
```

```
## 3                     1                       1     4 1219017600
## 4                     3                       3     2 1307923200
## 5                     0                       0     5 1350777600
## 6                     0                       0     4 1342051200
##                  Summary
## 1 Good Quality Dog Food
## 2     Not as Advertised
## 3 "Delight" says it all
## 4        Cough Medicine
## 5           Great taffy
## 6            Nice Taffy
##
Text
## 1
I have bought several of the Vitality canned dog food products and have found
them all to be of good quality. The product looks more like a stew than a
processed meat and it smells better. My Labrador is finicky and she
appreciates this product better than  most.
## 2
Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually
small sized unsalted. Not sure if this was an error or if the vendor intended
to represent the product as "Jumbo".
## 3 This is a confection that has been around a few centuries.  It is a
light, pillowy citrus gelatin with nuts - in this case Filberts. And it is
cut into tiny squares and then liberally coated with powdered sugar.  And it
is a tiny mouthful of heaven.  Not too chewy, and very flavorful.  I highly
recommend this yummy treat.  If you are familiar with the story of C.S.
Lewis' "The Lion, The Witch, and The Wardrobe" - this is the treat that
seduces Edmund into selling out his Brother and Sisters to the Witch.
## 4
If you are looking for the secret ingredient in Robitussin I believe I have
found it.  I got this in addition to the Root Beer Extract I ordered (which
was good) and made some cherry soda.  The flavor is very medicinal.
## 5
Great taffy at a great price.  There was a wide assortment of yummy taffy.
Delivery was very quick.  If your a taffy lover, this is a deal.
## 6
I got a wild hair for taffy and ordered this five pound bag. The taffy was
all very enjoyable with many flavors: watermelon, root beer, melon,
peppermint, grape, etc. My only complaint is there was a bit too much
red/black licorice-flavored pieces (just not my particular favorites).
Between me, my kids, and my husband, this lasted only two weeks! I would
recommend this brand of taffy -- it was a delightful treat.
```

Data Cleaning and Preprocessing

We performed data preprocessing and cleaning for a dataset. Initially, we focused on selecting relevant columns (non-user features) such as Id, ProductId, UserId, HelpfulnessNumerator, HelpfulnessDenominator, Score, Time, Summary, and Text while

excluding any rows with missing values. This step ensured that we retained essential information for our analysis while maintaining data integrity.

Subsequently, we prepared the text data within the reviews.df dataframe. Key transformations were applied to the Text column, including converting text to lowercase, removing unwanted characters such as newline characters (\n), and eliminating specific patterns like HTML entities (&). Additionally, we addressed URLs by removing them along with hashtags and account mentions from the text.

The aim of these preprocessing steps was to create a clean and standardized text corpus suitable for subsequent text mining and analysis tasks. we ensured a more consistent and focused textual dataset.

Later we created a text corpus based on the Text column of the reviews.df dataframe. The corpus was processed through a sequence of transformations such as converting all text to lowercase, removing Punctuation, removing numbers, removing English stop words Words, and eliminating the Extra whitespaces.

These preprocessing steps are essential for improving the quality of text data, making it conducive to tasks such as sentiment analysis, topic modeling, or network analysis. The resulting reviews.df dataframe serves as a refined foundation for gaining insights into the underlying textual content of the dataset.

Subsequently, the cleaned corpus was applied to the reviews.df dataframe, replacing the original Text column with the processed version (out)

```
#  Select non-user related data and Clean the missing data
cleaned_data <- data %>%
  select(Id, ProductId, HelpfulnessNumerator,HelpfulnessDenominator, Score,
Time, Summary, Text) %>%
  drop_na()  # Remove any rows with missing values

# text preparation
reviews.df <- cleaned_data %>%
  # Convert to lowercase.
  mutate(Text = Text %>% str_to_lower) %>%
  # Remove unwanted characters.
  mutate(Text= Text %>% str_remove_all(pattern = '\\n')) %>%
  mutate(Text = Text %>% str_remove_all(pattern = '&amp')) %>%
  mutate(Text = Text %>% str_remove_all(pattern = 'https://t.co/[a-z,A-Z,0-
9]*')) %>%
  mutate(Text = Text %>% str_remove_all(pattern = 'http://t.co/[a-z,A-Z,0-
9]*')) %>%
  mutate(Text = Text %>% str_remove_all(pattern = 'https')) %>%
  mutate(Text = Text %>% str_remove_all(pattern = 'http')) %>%
  # Remove hashtags.
  mutate(Text = Text %>% str_remove_all(pattern = '#[a-z,A-Z]*')) %>%
  # Remove accounts.
  mutate(Text = Text %>% str_remove_all(pattern = '@[a-z,A-Z]*'))
```

```r
#Creating corpus
corpus <-  VCorpus(x = VectorSource(x = reviews.df$Text))

# Perform additional text cleaning on corpus
clean_corpus <- corpus %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers) %>%
  tm_map(removeWords, stopwords("english")) %>%
  tm_map(stripWhitespace)

# Text from corpus to the vector out

out <- sapply(clean_corpus, function(x){x$content})
# out

is.vector(out)
```

```
## [1] TRUE
```

```r
#replacing the original Text column with the processed version

reviews.df %<>% mutate(Text = out)
```

Analyzing Word Frequency: What Words Echo Loudest?

In this section, our focus shifts to the textual content of the reviews, aiming to identify the most frequently occurring words. The initial step involves creating a word frequency table (word_count) that highlights the top words and their respective counts. It's worth noting that, during execution, a warning about outer names for unnamed scalar atomic inputs might appear. Despite this warning, the results remain accurate.

Upon analyzing the output, the top 10 words and their frequencies in the Amazon product reviews dataset are revealed. Notably, the term "br" stands out with 16,305 occurrences, raising a red flag for further investigation. Upon careful examination, it was determined that "br" was an artifact or noise introduced during the text cleaning and tokenization process, originating from HTML line break elements ("").

To address this issue, we carefully inspected the data and identified the source of these occurrences. Given that is an HTML line break element, we recognized the need to remove it from our analysis.

```r
# Counting the most popular words in the reviews
stopwords.df <- tibble(
  word = c(stopwords(kind = 'en'))
  )
words.df <- reviews.df %>%
  unnest_tokens(input = Text, output = word) %>%
  anti_join(y = stopwords.df, by = 'word')
```

```
## Warning: Outer names are only allowed for unnamed scalar atomic inputs
```

```
word.count <- words.df %>% count(word, sort = TRUE)

word.count %>% head(10)
```

```
##       word     n
## 1       br 16305
## 2     like 15446
## 3     good 12064
## 4   coffee 10579
## 5     just 10431
## 6      one 10366
## 7    great 10236
## 8    taste 10155
## 9   flavor  8775
## 10 product  8775
```

```
# Identify and remove <br /> occurrences
words.df <- words.df %>%
  filter(word != 'br')

# Count word frequencies
word.count <- words.df %>%
  count(word, sort = TRUE)

# Display the top 10 words
word.count %>% head(10)
```

```
##       word     n
## 1     like 15446
## 2     good 12064
## 3   coffee 10579
## 4     just 10431
## 5      one 10366
## 6    great 10236
## 7    taste 10155
## 8   flavor  8775
## 9  product  8775
## 10     can  8667
```

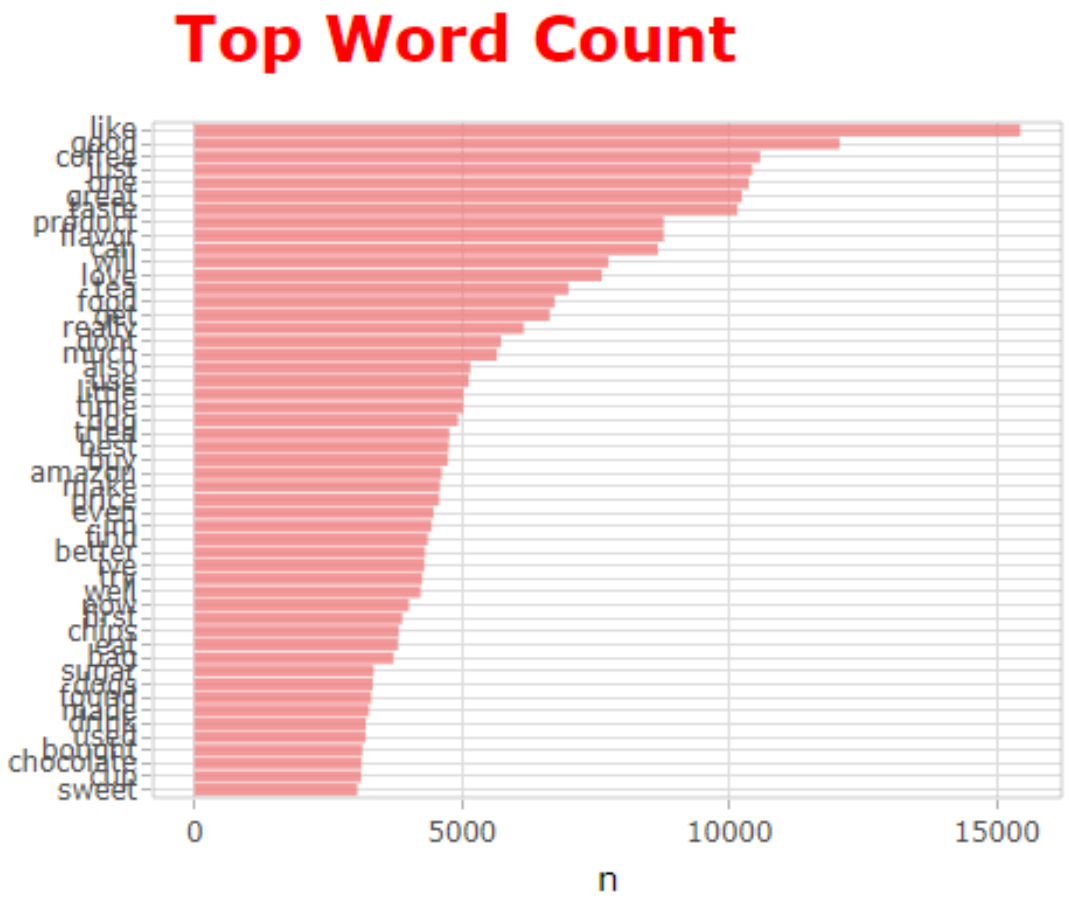
Visualization of the most frequently occurring words

In this section, our objective was to visually represent the most frequently occurring words in the reviews. We initiated the process by calculating the frequency of each word after tokenizing and eliminating stopwords from the review text, resulting in the creation of the word.count dataframe containing words and their respective frequencies. To highlight the most significant words, a count threshold of 3000 was established, guiding the subsequent analysis. The top word counts were then visualized through a bar plot (plt), where the x-axis denoted words and the y-axis represented their frequencies, showcasing only those

surpassing the defined threshold. To further enhance the interactive exploration of the data, the bar plot was transformed into an interactive plot using ggplotly(). Additionally, a word cloud was generated using the wordcloud package, presenting a visually compelling representation of the most frequent words. The color palette chosen for the word cloud aimed to improve visibility and overall aesthetics. Throughout these steps, the determination of thresholds was intricately linked to the analysis of the output data at each stage, ensuring a judicious balance between inclusivity and the emphasis on pertinent information in the reviews.

```
# visualize these counts in a bar plot
plt <- word.count %>%
  # Set count threshold.
  filter(n > 3000) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(x = word, y = n)) +
  theme_light() +
  geom_col(fill = 'lightcoral', alpha = 0.8) +
  xlab(NULL) +
  coord_flip() +
 ggtitle("Top Word Count") +
  theme(plot.title = element_text(size = 20, color = 'red', face = 'bold'))


plt %>% ggplotly()
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs().
If it is installed, please make sure the phantomjs executable can be found
via the PATH variable.
```




```
wordcloud(
  words = word.count$word,
  freq = word.count$n,
  min.freq = 3000,
  colors = brewer.pal(8, 'Dark2'),
)

## Warning in wordcloud(words = word.count$word, freq = word.count$n,
min.freq =
## 3000, : like could not be fit on page. It will not be plotted.
```

Network Analyze

Navigating Bigrams: Unveiling Word Companionships

In the bi-gram section, we analyze pairwise occurrences of words appearing together in the text. The goal is to identify meaningful associations between words and create a network representation of bigram relationships. We filter out stop words and white spaces, count the occurrences of each bigram, and visualize the distribution of their weights. To manage the skewed distribution, a log transformation is applied. A threshold is set to define the minimal weight allowed in the graph, and the resulting bigram network is visualized. The network is represented using a force-directed layout, with custom colors for vertex labels and a size scale based on the degree of each node.

The output displays the top bigrams based on their weights, representing the frequency of occurrence. Each row consists of two words (word1 and word2) forming a bigram, and the corresponding weight indicates the number of times that bigram appears in the text. For example, the bigram “highly recommend” has a weight of 898, indicating that this phrase is frequently used in the reviews. Similarly, other bigrams like “peanut butter,” “taste like,” and “gluten-free” are also common expressions. The table provides insights into meaningful word associations and recurring phrases in the analyzed text.

```
# count pairwise occurrences of words which appear together in the text
bi.gram.words <- reviews.df %>%
  unnest_tokens(
    input = Text,
    output = bigram,
```

```
    token = 'ngrams',
    n = 2
  ) %>%
  filter(! is.na(bigram))
```

```
## Warning: Outer names are only allowed for unnamed scalar atomic inputs
```

```
bi.gram.words %>%
  select(bigram) %>%
  head()
```

```
##             bigram
## 1   bought several
## 2 several vitality
## 3  vitality canned
## 4       canned dog
## 5         dog food
## 6    food products
```

```
# filter for stop words and remove white spaces
bi.gram.words %<>%
  separate(col = bigram, into = c('word1', 'word2'), sep = ' ') %>%
  filter(! word1 %in% stopwords.df$word) %>%
  filter(! word2 %in% stopwords.df$word) %>%
  filter(! is.na(word1)) %>%
  filter(! is.na(word2))

# group and count by bigram
bi.gram.count <- bi.gram.words %>%
  count(word1, word2, sort = TRUE) %>%
  # We rename the weight column so that the
  # associated network gets the weights (see below).
  rename(weight = n)

bi.gram.count %>% head()
```

```
##     word1     word2 weight
## 1  highly recommend    898
## 2  peanut    butter    830
## 3   taste      like    825
## 4  gluten      free    818
## 5     ive     tried    790
## 6 grocery     store    728
```

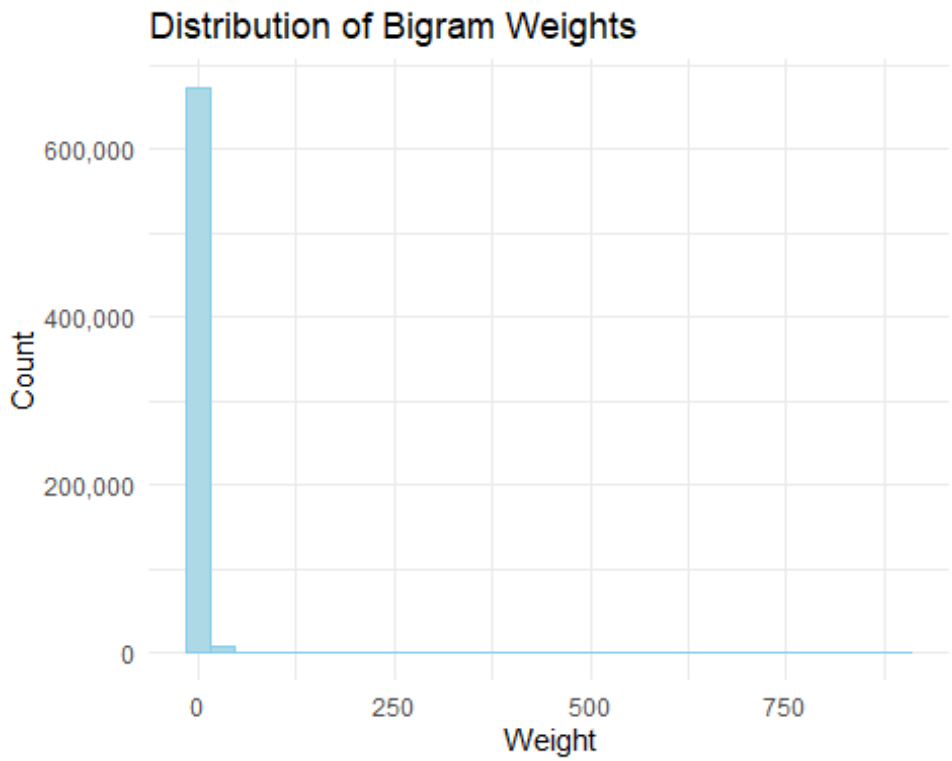
Visualization of Bi-gram Results

In the normal bigram weight distribution plot, in the beginning we observe that the y-axis displays counts in scientific notation (e.g., 2e+05), which can be visually challenging. To address this, we modify the y-axis to display counts in a more readable format. Additionally, we enhance the visual appeal of the plots by introducing vibrant and customized colors, deviating from the default settings to make the graphs more engaging.

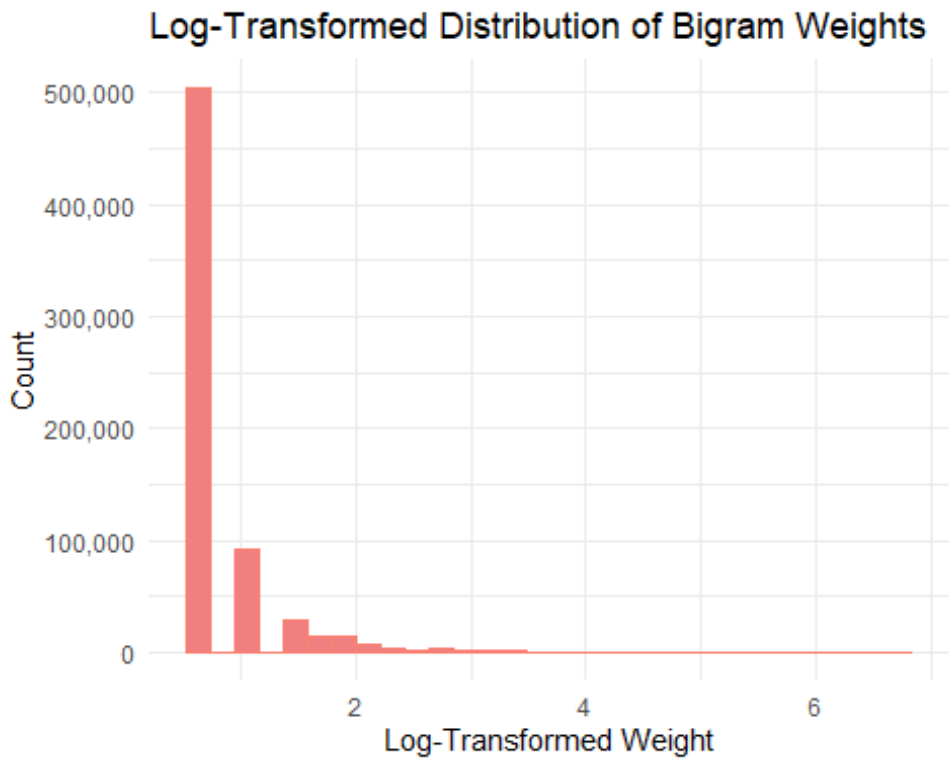
Having plotted the normal bigram weight distribution, we observed a right-skewed distribution that prompted us to address readability issues in the y-axis counts by modifying the format. Due to the skewed nature of the distribution, we opted for a logarithmic transformation in the log-transformed bigram weight distribution plot. This transformation normalize the data for better visualization and allowed better interpretation of the weight patterns in the bigrams.

From these results, we confronted a trade-off between network inclusivity and the strength of connections. A higher threshold (around 500-550) emphasized fewer bigrams with robust associations, fostering a focused network. Conversely, a lower threshold (around 100-150) prioritized a more extensive network but potentially weaker connections. Opting for a medium threshold (300-350) struck a balance, incorporating a moderate number of bigrams (41-58) with average weights around 0.27. This choice aimed to capture a meaningful yet comprehensive representation of word associations in the Amazon product reviews, offering a nuanced perspective that considers both network strength and inclusivity. The chosen threshold, set at 350, was applied to the weights. We normalized the weights for better visualization by scaling them with a global factor. Using igraph, we created a network object and visualized it, adjusting parameters such as vertex size, color, and edge width. The resulting force-directed network plot provides a comprehensive view of significant bigram relationships, contributing to a deeper understanding of patterns within the dataset.

```
# Plotting the distribution of bigram weights (Normal)
bi.gram.count %>%
  ggplot(mapping = aes(x = weight)) +
    theme_minimal() +
    geom_histogram(color = "skyblue", fill = "lightblue", bins = 30) +
    scale_y_continuous(labels = scales::comma) +  # Display counts with
commas
    labs(
      title = "Distribution of Bigram Weights",
      x = "Weight",
      y = "Count"
    )
```



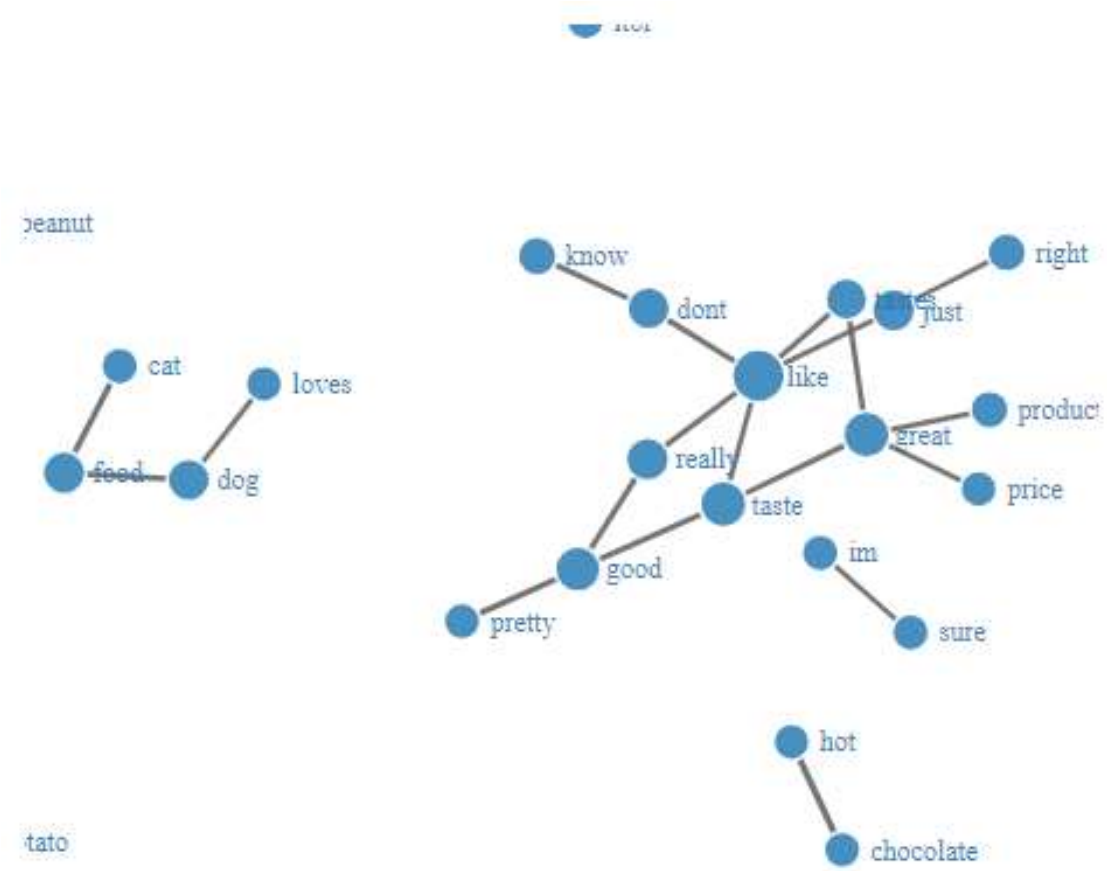

```
# very skewed, for visualization purposes it might be a good idea to perform
a transformation
# Plotting the distribution of log-transformed bigram weights
bi.gram.count %>%
  mutate(log_weight = log(weight + 1)) %>%
  ggplot(mapping = aes(x = log_weight)) +
    theme_minimal() +
    geom_histogram(color = "salmon", fill = "lightcoral", bins = 30) +
    scale_y_continuous(labels = scales::comma) +  # Display counts with
commas
    labs(
      title = "Log-Transformed Distribution of Bigram Weights",
      x = "Log-Transformed Weight",
      y = "Count"
    )
```




```
# Set the threshold

threshold <- 350

# For visualization purposes we scale by a global factor.
ScaleWeight <- function(x, lambda) {
  x / lambda
}

network <-  bi.gram.count %>%
  filter(weight > threshold) %>%
  mutate(weight = ScaleWeight(x = weight, lambda = 2E3)) %>%
  graph_from_data_frame(directed = FALSE)

network

## IGRAPH 1397488 UNW- 61 41 --
## + attr: name (v/c), weight (e/n)
## + edges from 1397488 (vertex names):
##  [1] highly --recommend peanut --butter    taste  --like      gluten --
free
##  [5] ive    --tried     grocery--store     dog    --food      much   --
better
##  [9] cup    --coffee    dont   --know      just   --right     really --
good
## [13] tastes --like      green  --tea       can    --get       year   --old
## [17] really --like      great  --product   just   --like      potato --
```

```
chips
## [21] first  --time      taste  --good      dont   --like      itbr   --br
## [25] im     --sure      cat    --food      dogs   --love      br     --br
## + ... omitted several edges

is.weighted(network)

## [1] TRUE

# Generate a palette of colors for vertex labels
label_colors <- rainbow(length(V(network)))

# Plot the network with custom colors for vertex labels
plot(
  network,
  vertex.size = 1,
  vertex.label.color = label_colors,
  vertex.label.cex = 0.7,
  vertex.label.dist = 1,
  vertex.color = 'white',  # Set vertex color to white
  edge.color = 'gray',
  main = 'Bigram Count Network',
  sub = glue('Weight Threshold: {threshold}'),
  alpha = 50
)
```

```r
#Set threshold
threshold <- 350

network <-  bi.gram.count %>%
  filter(weight > threshold) %>%
  graph_from_data_frame(directed = FALSE)

# Store the degree.
V(network)$degree <- strength(graph = network)
# Compute the weight shares.
E(network)$width <- E(network)$weight/max(E(network)$weight)

# Create networkD3 object.
network.D3 <- igraph_to_networkD3(g = network)
# Define node size.
network.D3$nodes %<>% mutate(Degree = (1E-2)*V(network)$degree)
# Degine color group (I will explore this feature later).
network.D3$nodes %<>% mutate(Group = 1)
# Define edges width.
network.D3$links$Width <- 10*E(network)$width

forceNetwork(
  Links = network.D3$links,
  Nodes = network.D3$nodes,
  Source = 'source',
  Target = 'target',
  NodeID = 'name',
  Group = 'Group',
  opacity = 0.9,
  Value = 'Width',
  Nodesize = 'Degree',
  # We input a JavaScript function.
  linkWidth = JS("function(d) { return Math.sqrt(d.value); }"),
  fontSize = 12,
  zoom = TRUE,
  opacityNoHover = 1
)
```



Skip-grams: Bridging Gaps in Word Connections

In this section, we extend our analysis by considering skipgrams, which allow for a "jump" in the word count. We extract skipgrams containing two words, count their occurrences, and visualize the resulting network. Similar to the bigram section, we apply a threshold to filter the network and focus on meaningful relationships. The dynamic plot is generated to highlight the most significant connections within the skipgram network, allowing for a more comprehensive exploration of word associations in the text data. The thresholds and parameters are chosen based on the outputs and the specific goals of the analysis.

Analyzing the results, a medium threshold of 300 appears to strike a reasonable balance between the number of skip-grams retained and the average weight. With this threshold, the network includes 80 skip-grams with an average weight of 480,88. This choice aims to provide a meaningful representation of associations in the skip-grams, capturing a moderate level of connection strength while still being inclusive enough to offer insights into the data

```
# consider skipgrams, which allow a "jump" in the word count

skip.window <- 2

skip.gram.words <- reviews.df %>%
  unnest_tokens(
```

```
    input = Text,
    output = skipgram,
    token = 'skip_ngrams',
    n = skip.window
  ) %>%
  filter(! is.na(skipgram))

## Warning: Outer names are only allowed for unnamed scalar atomic inputs

# consider the example review

reviews.df %>%
  slice(4) %>%
  pull(Text)

##
4
## " looking secret ingredient robitussin believe found got addition root
beer extract ordered good made cherry soda flavor medicinal"

# The skipgrams are
skip.gram.words %>%
  select(skipgram) %>%
  slice(10:20)

##          skipgram
## 1          canned
## 2      canned dog
## 3     canned food
## 4             dog
## 5        dog food
## 6    dog products
## 7            food
## 8   food products
## 9      food found
## 10       products
## 11 products found

# count the skipgrams containing two words
library(ngram)

## Warning: package 'ngram' was built under R version 4.3.2

skip.gram.words$num_words <- skip.gram.words$skipgram %>%
  map_int(.f = ~ ngram::wordcount(.x))

skip.gram.words %<>% filter(num_words == 2) %>% select(- num_words)

skip.gram.words %<>%
  separate(col = skipgram, into = c('word1', 'word2'), sep = ' ') %>%
  filter(! word1 %in% stopwords.df$word) %>%
  filter(! word2 %in% stopwords.df$word) %>%
```

```r
  filter(! is.na(word1)) %>%
  filter(! is.na(word2))

skip.gram.count <- skip.gram.words  %>%
  count(word1, word2, sort = TRUE) %>%
  rename(weight = n)

skip.gram.count %>% head()
```

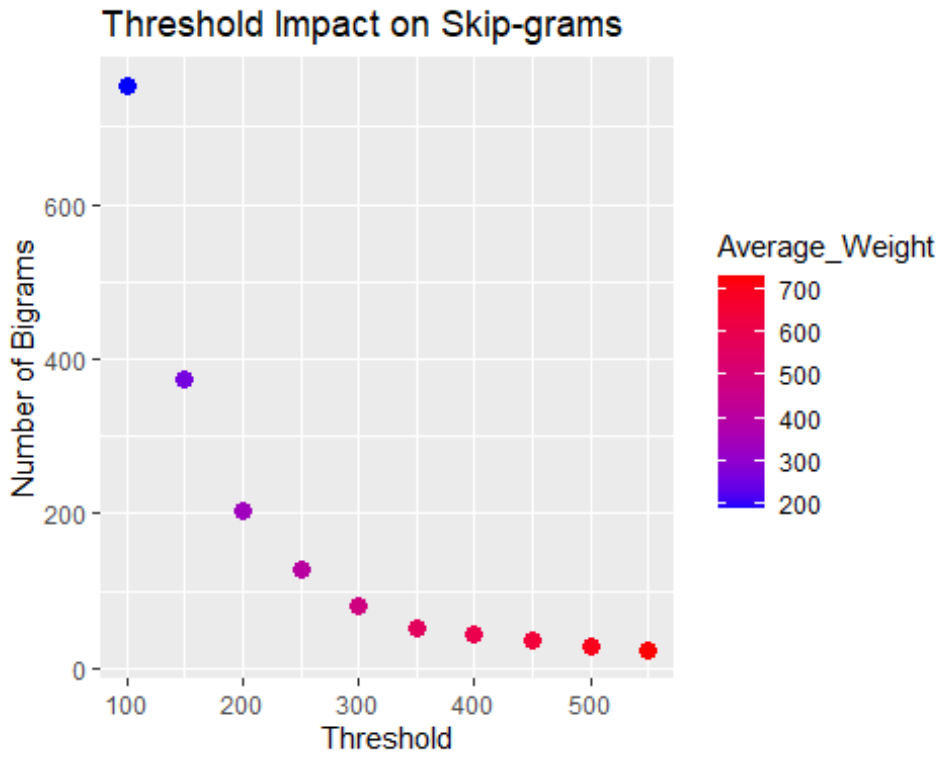
```
##    word1     word2 weight
## 1  taste      like   1156
## 2 highly recommend    901
## 3    ive     tried    870
## 4 gluten      free    840
## 5 peanut    butter    830
## 6 tastes      like    804
```

```r
# Create an empty data frame to store results
threshold_summary <- data.frame()

# Loop through different threshold values
for (threshold in c(100, 150, 200, 250, 300, 350, 400, 450, 500, 550)) {

  # Filter skip-gram count based on the threshold
  filtered_skip_grams <- skip.gram.count %>%
    filter(weight > threshold)

  # Calculate the number of bigrams and average weight
  num_bigrams <- nrow(filtered_skip_grams)
  avg_weight <- mean(filtered_skip_grams$weight)

  # Add results to the summary data frame
  threshold_summary <- rbind(threshold_summary, c(threshold, num_bigrams,
avg_weight))
}

# Rename columns
colnames(threshold_summary) <- c("Threshold", "Number_of_Bigrams",
"Average_Weight")

# Display the summary table
print(threshold_summary)
```

```
##    Threshold Number_of_Bigrams Average_Weight
## 1        100               753       190.3479
## 2        150               373       261.3592
## 3        200               203       336.9310
## 4        250               128       401.8906
## 5        300                80       480.8750
## 6        350                52       564.6346
```

```
## 7         400                 43        603.9767
## 8         450                 35        646.8857
## 9         500                 27        700.1481
## 10        550                 23        730.0870
```

```
# Visualize the results
library(ggplot2)

ggplot(threshold_summary, aes(x = Threshold, y = Number_of_Bigrams, color =
Average_Weight)) +
  geom_point(size = 3) +
  scale_color_gradient(low = "blue", high = "red") +
  labs(title = "Threshold Impact on Skip-grams", x = "Threshold", y = "Number
of Bigrams")
```




```
# dynamic plot (RStudio)
# Treshold
threshold <- 300

network <-  skip.gram.count %>%
  filter(weight > threshold) %>%
  graph_from_data_frame(directed = FALSE)

# Select biggest connected component.
V(network)$cluster <- clusters(graph = network)$membership

cc.network <- induced_subgraph(
```

```
  graph = network,
  vids = which(V(network)$cluster == which.max(clusters(graph =
network)$csize))
)

# Store the degree.
V(cc.network)$degree <- strength(graph = cc.network)
# Compute the weight shares.
E(cc.network)$width <- E(cc.network)$weight/max(E(cc.network)$weight)

# Create networkD3 object.
network.D3 <- igraph_to_networkD3(g = cc.network)
# Define node size.
network.D3$nodes %<>% mutate(Degree = (1E-2)*V(cc.network)$degree)
# Degine color group (I will explore this feature later).
network.D3$nodes %<>% mutate(Group = 1)
# Define edges width.
network.D3$links$Width <- 10*E(cc.network)$width

forceNetwork(
  Links = network.D3$links,
  Nodes = network.D3$nodes,
  Source = 'source',
  Target = 'target',
  NodeID = 'name',
  Group = 'Group',
  opacity = 0.9,
  Value = 'Width',
  Nodesize = 'Degree',
  # We input a JavaScript function.
  linkWidth = JS("function(d) { return Math.sqrt(d.value); }"),
  fontSize = 12,
  zoom = TRUE,
  opacityNoHover = 1
)
```

Compare the ngram results

Comparing the results of the top 10 bigrams and skip-grams, we observe some similarities and differences. In both cases, there are expressions related to taste (“taste like”), recommendations (“highly recommend”), and product experiences (“ive tried”, “much better”,“peanut butter”, “dog food”). However, there are unique combinations as well, such as “grocery store” , “dont know” in bigrams and “tastes like” , “really like” in skip-grams.

The weights indicate the strength of association between the words. For instance, “taste like” has a higher weight in skip-grams (1156) compared to “highly recommend” (898) in bigrams, suggesting a potentially stronger connection between these words in the context of reviews.

These comparisons help us understand how different n-gram approaches capture distinct patterns and associations in the dataset, providing valuable insights into the language used by reviewers. Adjusting thresholds and exploring various n-gram combinations allow for a more nuanced analysis of word relationships in the reviews.

```
bi.gram.count %>% head(10)
```

```
##      word1     word2 weight
## 1   highly recommend    898
## 2   peanut    butter    830
```

```
## 3    taste      like    825
## 4   gluten      free    818
## 5      ive     tried    790
## 6  grocery     store    728
## 7      dog      food    712
## 8     much    better    705
## 9      cup    coffee    639
## 10    dont      know    628
```

```
skip.gram.count %>% head(10)
```

```
##     word1     word2 weight
## 1   taste      like   1156
## 2  highly recommend    901
## 3     ive     tried    870
## 4  gluten      free    840
## 5  peanut    butter    830
## 6  tastes      like    804
## 7     dog      food    773
## 8  really      like    761
## 9     cup    coffee    744
## 10   much    better    737
```

Insights and Takeaways

Taste Dictates Trends

Expressions related to taste ("taste like," "really like") echoed prominently across bigrams and skip-grams. The emphasis on sensory experiences underlines the pivotal role taste plays in shaping consumer perceptions.

Recommendations Echo Loudly

"Highly recommend" resonated strongly, underscoring the influential power of recommendations in shaping consumer choices. The echo of positive endorsements reverberated throughout the dataset.

Product-Specific Conversations

Distinctive product references like "peanut butter" , "dog food" and "grocery store" surfaced, indicating specific product categories driving conversations. Understanding these nuances is crucial for businesses aiming to align with consumer preferences.

In Closing

Our exploration into the Amazon food reviews goes beyond the surface of individual words, revealing a intricate mosaic of consumer sentiments and preferences. Whether it's the flavor-centric vocabulary or the reverberation of recommendations, each word contributes to a narrative woven intricately in the vast tapestry of Amazon's food reviews. As businesses traverse this linguistic landscape, these insights act as a guiding compass,

providing a profound understanding of their customers. In the orchestration of reviews, every word assumes a pivotal role, and our analysis aims to decode the harmonious melody embedded within.

REFERENCES

J. Leskovec, L. Adamic, and B. Adamic. The Dynamics of Viral Marketing. ACM Transactions on the Web (ACM TWEB), 1(1), 2007. http://snap.stanford.edu/data/amazon0302.html J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. WWW, 2013.
https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews