Introduction to LSA– Use Cases

Promothesh Chatterjee*

Overview

Imagine 3 scenarios:

1) You have a corpus of reviews, your manager asks you to create a perceptual map of your brand vis-a-vis other brands.

2) Consider a large company that receives thousands of customer support tickets every day. Each ticket contains a description of the customer's issue written in natural language. The company wants to efficiently route each ticket to the appropriate support team in order to resolve issues as quickly as possible. However, manually reading and categorizing each ticket would be time-consuming and prone to errors.

3) For an emergency business situation, your manager asks you to provide a quick summary of some articles in 5 minutes.

In all three cases, we can use Latent Semantic Analysis to come up with some kind of answer.

R Code for scenario 1:

```
library(tidyverse)
library(quanteda)
library(quanteda.textmodels)
library(ggplot2)

# Sample dataset with brand names and associated reviews
reviews <- tibble(
    brand = c("Brand_A", "Brand_B", "Brand_C", "Brand_D", "Brand_E"),
    review = c(
        "Brand A is reliable and has excellent customer service.",
        "Brand B is innovative but has poor customer service.",
        "Brand C is affordable and has a good warranty.",
        "Brand D is expensive but offers high-quality products.",
        "Brand E is eco-friendly and has a great design."
    )
)

# Tokenize and preprocess the text data
tokens <- reviews$review %>%
    tokens(remove_punct = TRUE, remove_numbers = TRUE) %>%
    tokens_tolower() %>%
    tokens_remove(stopwords("en"))
```

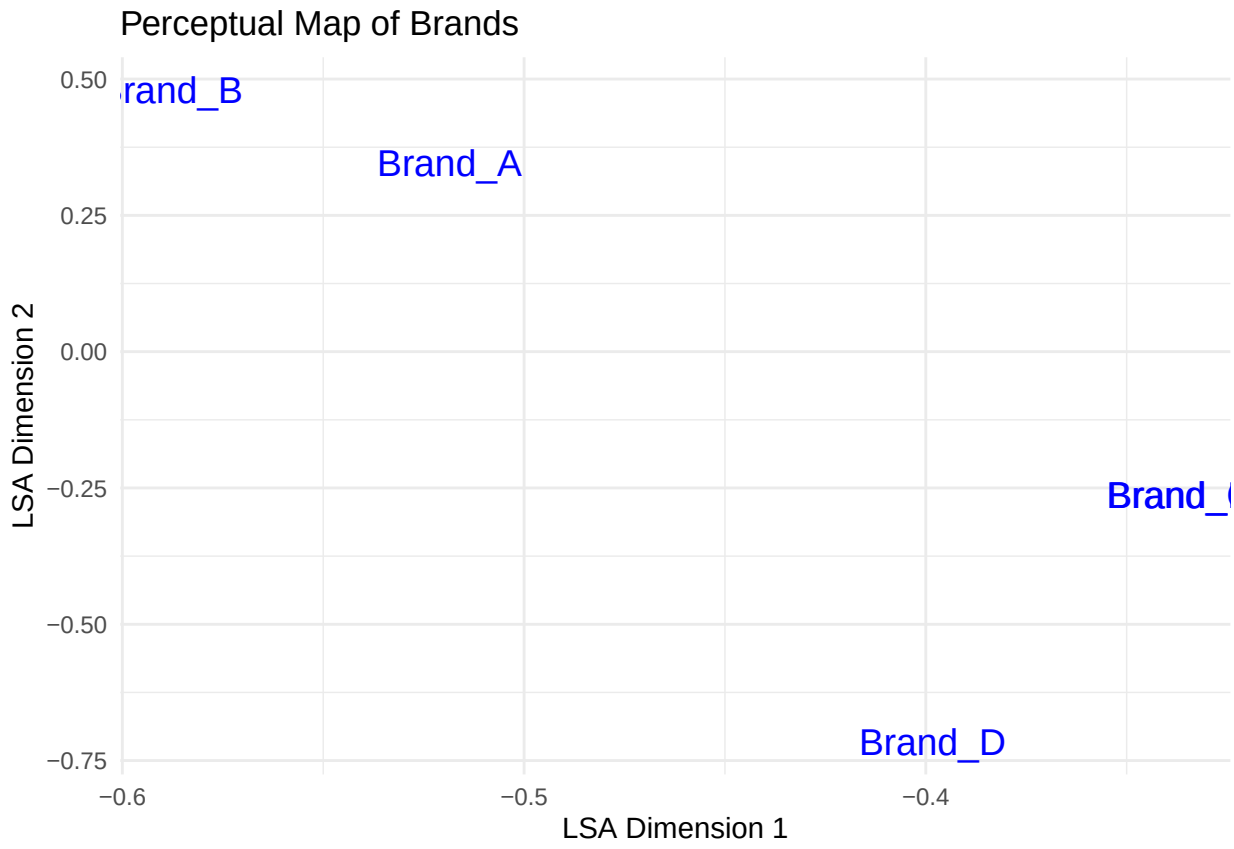
```r
# Create a Document-Feature Matrix (DFM)
dfm <- tokens %>%
    dfm()

# Perform LSA using quanteda.textmodels
lsa_space <- textmodel_lsa(dfm, nd = 2)

# View the LSA scores for the documents
lsa_scores <- as.data.frame(head(lsa_space$docs))

# Combine brand names and LSA scores
brand_scores <- cbind(reviews$brand, lsa_scores)

# Create the perceptual map
ggplot(brand_scores, aes(x = V1, y = V2, label = reviews$brand)) +
    geom_text(size = 5, color = "blue") +
    labs(title = "Perceptual Map of Brands", x = "LSA Dimension 1", y = "LSA Dimension 2") +
    theme_minimal()
```



```r
# View the document scores
head(lsa_space$docs)
```

```
##              [,1]       [,2]
## text1 -0.5185246  0.3459867
## text2 -0.5879511  0.4801588
## text3 -0.3367411 -0.2623184
## text4 -0.3983172 -0.7156249
## text5 -0.3367411 -0.2623184
```

```r
# View the feature scores
head(lsa_space$features)
```

```
##                 [,1]       [,2]
## brand     -0.6732347 -0.1753301
## reliable  -0.1602593  0.1464852
## excellent -0.1602593  0.1464852
## customer  -0.3419760  0.3497767
## service   -0.3419760  0.3497767
## b         -0.1817167  0.2032915
```

This script uses the quanteda and ggplot2 libraries to perform Latent Semantic Analysis (LSA) on a set of reviews for different brands and then visualizes the results in a perceptual map. Here's a breakdown of the steps:

1. The script first loads the necessary R packages: 'tidyverse' for data manipulation, 'quanteda' for text analysis, 'quanteda.textmodels' for text modelling, and 'ggplot2' for data visualization.

2. It then creates a tibble (a data frame variant from the tidyverse) named 'reviews' that contains brand names and associated reviews.

3. The reviews are tokenized, which means they are broken down into individual words or "tokens". The code also removes punctuation and numbers, converts all the text to lowercase, and removes common English "stop words" (like "the", "and", "is", etc.) that don't usually provide much information for text analysis.

4. The 'dfm()' function is used to create a Document-Feature Matrix (DFM). This is a matrix where each row represents a document (in this case, a review), each column represents a feature (a unique word), and each cell represents the frequency of a word in a particular document.

5. The 'textmodel_lsa()' function is used to perform LSA on the DFM, reducing the dimensionality of the data to 2 dimensions (as specified by 'nd = 2'). The result is a new space where each document is represented as a point, and the spatial relationships between points reflect the semantic relationships between the corresponding documents.

6. The LSA scores for the documents are extracted and combined with the brand names.

7. 'ggplot2' is used to create a perceptual map of the brands, which is a graphical representation of the brands' positions in the LSA space. Each brand is represented as a point, and the spatial relationships

between points reflect the semantic relationships between the corresponding reviews. The axes represent the two dimensions of the LSA space.

8. Finally, the script prints the document and feature scores from the LSA. These scores represent the positions of the documents and features in the LSA space.

In summary, this script uses LSA to analyze the semantic content of reviews for different brands, and then visualizes the results in a perceptual map. This can provide insights into how the brands are perceived in terms of different aspects (which are represented by the dimensions of the LSA space). For example, if the first dimension of the LSA space corresponds to the aspect of price, brands with higher scores on this dimension might be perceived as more expensive.

The use case of this script is to understand brand perceptions based on customer reviews using Latent Semantic Analysis. This could be helpful for market research, brand management, and competitive analysis. The perceptual map generated provides a visual representation of how different brands are perceived in relation to each other based on the semantic content of the reviews.

R code for scenario 2

```
library(quanteda)
library(quanteda.textmodels)
library(cluster)

# Synthetic dataset of customer support tickets
tickets <- tibble(
    id = 1:6,
    text = c(
        "I have a problem with my bill. The amount is incorrect.",
        "My phone is not working. It won't turn on.",
        "I was overcharged on my last bill. Can you help?",
        "My device is having issues. The screen is blank.",
        "I have a call-drop issue. It is really bad.",
        "The phone is broken. It's not charging."
    )
)
# Create a Document-Feature Matrix (DFM)
dfm <- tickets$text %>%
    tokens(remove_punct = TRUE, remove_numbers = TRUE) %>%
    tokens_tolower() %>%
    tokens_remove(stopwords("en"), padding = TRUE) %>%
    dfm()

# Perform LSA
lsa_space <- textmodel_lsa(dfm, nd = 2)
```

```r
# Get LSA scores for the documents
lsa_scores <- as.data.frame(lsa_space$docs[, 1:2])

# Perform k-means clustering
set.seed(123)  # for reproducibility
clusters <- kmeans(lsa_scores, centers = 2)

# Add cluster assignments to the ticket data
tickets$cluster <- clusters$cluster

# Print tickets with cluster assignments
print(tickets)
```

```
## # A tibble: 6 x 3
##      id text                                                    cluster
##   <int> <chr>                                                   <int>
## 1     1 I have a problem with my bill. The amount is incorrect.       1
## 2     2 My phone is not working. It won't turn on.                    2
## 3     3 I was overcharged on my last bill. Can you help?              1
## 4     4 My device is having issues. The screen is blank.              2
## 5     5 I have a call-drop issue. It is really bad.                   2
## 6     6 The phone is broken. It's not charging.                       2
```

The provided code is an example of how Latent Semantic Analysis (LSA) can be used in conjunction with k-means clustering to categorize customer support tickets into different issue types.

This is a simplified case study where a telecommunications company is receiving a number of customer support tickets. The tickets encompass a range of issues, from billing problems to device malfunctions. The goal is to automatically categorize these tickets into clusters based on the semantic content of the ticket text, which can then be used to route the tickets to the appropriate support teams.

Here's a detailed breakdown of the process:

1. Data Preparation: The script starts with a synthetic dataset of six customer support tickets, each containing a text description of the problem.

2. Text Preprocessing: The tickets are preprocessed using tokenization, removal of punctuation and numbers, conversion to lowercase, and removal of English stopwords. This ensures the analysis focuses on the meaningful parts of the text.

3. Document-Feature Matrix (DFM): The preprocessed tokens are transformed into a DFM, which represents the frequency of terms in each ticket.

4. Latent Semantic Analysis (LSA): LSA captures the semantic relationships between words and documents, and represents these relationships in a lower-dimensional space. In this case, the script reduces the data to two dimensions.

5. K-means Clustering: K-means clustering is applied to the LSA scores of the tickets, dividing them into two clusters. The K-means algorithm is a popular choice for clustering analyses due to its simplicity and efficiency. It partitions the tickets into clusters based on their similarity in the LSA space.

6. Cluster Assignment: The cluster assignments from the k-means algorithm are added back to the original ticket data. This assigns each ticket to one of the two clusters.

In this case study, the result would be two groups of tickets, each representing a different type of issue. For example, tickets about billing problems might be grouped into one cluster, while tickets about device problems might form another cluster. This kind of automated categorization can help to streamline the process of handling customer support tickets, as each ticket can be directly routed to the team that handles the relevant type of issue.

It should be noted that in practice, the number of clusters and dimensions might need to be tuned based on the specifics of the data and the problem at hand. Also, more complex preprocessing might be needed, especially for larger and noisier datasets. Finally, the interpretation of the clusters should be validated with domain knowledge or by inspecting the tickets in each cluster.

R code for scenario 3

```
library(ralger)
library(polite)
bow("https://www.nytimes.com/2023/05/29/opinion/inflation-groceries-pricing-walmart.html")
```

```
## <polite session> https://www.nytimes.com/2023/05/29/opinion/inflation-groceries-pricing-walmart.html
##     User-agent: polite R package
##     robots.txt: 69 rules are defined for 28 bots
##    Crawl delay: 5 sec
##   The path is scrapable for this user-agent
```

```
my_link <- "https://www.nytimes.com/2023/05/29/opinion/inflation-groceries-pricing-walmart.html"

my_node<-"p"# selector gadget will give these element ID


DF <- scrap(link = my_link, node = my_node)

library(LSAfun)

genericSummary(DF,k=3)# k is the number of sentences to be used in summary
```

```
## [1] "Like other independent grocers, Food Fresh buys through large national wholesalers that purchas
## [2] "This isn't competition. It's big retailers exploiting their financial control over suppliers to
## [3] "Like other independent grocers, Food Fresh buys through large national wholesalers that purchas
```

In this case study, the R script is being used to scrape an article from the New York Times and generate a summary of the article using Latent Semantic Analysis (LSA).

Here's a detailed breakdown of the process:

1. Web Scraping: The 'ralger' and 'polite' libraries are used to scrape the content of the webpage. The 'bow' function from the 'ralger' library is used to fetch the webpage, while the 'scrap' function from the 'polite' library is used to extract the text from the specific HTML nodes (in this case, the paragraph nodes denoted by "p").

2. Latent Semantic Analysis (LSA): The 'genericSummary' function uses LSA to generate a summary of the text. LSA captures the semantic relationships between words and sentences and represents these relationships in a lower-dimensional space.

3. Text Summarization: The 'genericSummary' function uses the LSA representation to select the most representative sentences for the summary. In this case, the function is set to select three sentences (as indicated by 'k=3').

This case study demonstrates how web scraping and LSA can be used together to automatically summarize online articles. This can be very useful for a variety of applications, including news aggregation, content curation, and information retrieval. For instance, a news aggregation website might use this approach to provide concise summaries of articles from various sources, helping users to quickly get the gist of the articles without having to read them in full.