`easytm`: an easy way for text mining in R.

Dr. Kamakshaiah Musunuru [1]

Abstract

Abstract goes here.

1 Background

R offers quite a few packages and a vast opportunities for text mining and analysis. Particularly, the `tm` package offers lot of functions for mining data from a given input corpus. However, there are a few trivial tasks such as managing paths, creating subsets of data frames which are repetitive, routine and tiring. For instance, we may have to create a couple of directories one for scripts, the other for outputs so on and so forth. Such jobs eats lot of time and often leaves the day with frustration.

[1] I am Dr. Kamakshaiah Musunuru, an academic of data science and analytics and freelance trainer of enterprise solutions. I provide training on Data science and analytics, Full-stack development, AI & ML, Big data, Fintech etc..., using C++, Java, Python, R, Hadoop, Hyperledger and a couple of other software tools. I also deal with real-time sensor based data analytics and dashboards using AVR uC, ARM processor, ESP8266, nodeMCU. Please visit my Github portal located at `https://github.com/kamakshaiah` to know about my interests. Please reach me on +91-9848396972 or write to `dr.m.kamakshaiah@gmail.com`.

The script `easytm` helps the users in performing trivial tasks rather painlessly. No need to use `readlines()` for paths, `grep()`, for subsetting data sets, and `file.path()` for making file paths etc. The script is easy for implementation, just execute `source(easytm.R)` in R console to use it for data mining and analysis. I have no plans to make a package in near by future because it is still evolving but the source code is available at `https://github.com/Kamakshaiah/easytm`. *It's free, however, no guarantee or warranties are applied for any damages or losses of what-so-ever, users discretion is advised and required.*

1.1 R Packages for Text Mining and Analysis

Text mining and analysis in R are facilitated by several powerful packages that provide tools for processing, analyzing, and visualizing textual data. These packages leverage R's robust statistical and graphical capabilities, making it a preferred choice for researchers and analysts working with text data. Here are some key R packages commonly used for text mining and analysis:

1. `tm`: The 'tm' package stands for "Text Mining Infrastructure in R." It provides a framework for managing and manipulating text documents. Functions in 'tm' allow for tasks such as cleaning text, creating term-document matrices, and applying various transformations necessary for text analysis.

2. `quanteda`: 'quanteda' is a package designed for quantitative analysis of textual data. It offers a fast and flexible framework for tokenizing texts, creating document-feature matrices, and performing various text mining tasks such as sentiment analysis, topic modeling, and more.

3. `tidytext`: Developed as part of the tidyverse ecosystem, 'tidytext' provides tools for text mining using tidy data principles. It facilitates integrating text mining with other tidyverse pack-

ages like ‘dplyr‘ and ‘ggplot2‘, making it easy to manipulate and visualize results from text analyses.

4. `textmineR`: This package is specifically focused on text mining tasks such as term frequency analysis, text clustering, and text similarity calculations. It provides an interface to many text mining algorithms and allows for easy extraction of insights from text data.

5. `topicmodels`: For topic modeling tasks such as Latent Dirichlet Allocation (LDA), the ‘topicmodels‘ package is widely used. It provides functions to fit topic models to text data and examine the resulting topics, making it valuable for exploring themes within large document collections.

6. `wordcloud`: The ‘wordcloud‘ package is useful for creating visually appealing word clouds from text data. It helps in visualizing word frequency and identifying prominent terms within a corpus.

7. `RWeka`: ‘RWeka‘ interfaces with the Weka machine learning toolkit, providing access to various machine learning algorithms that can be applied to text data. It extends R’s capabilities for classification, clustering, and other advanced text mining tasks.

8. `text2vec`: ‘text2vec‘ is designed for large-scale text mining tasks, offering efficient implementations of text processing algorithms such as tokenization, word embeddings, and document feature extraction. It’s particularly useful for handling large corpora that may not fit into memory.

9. `sentimentr`: For sentiment analysis, ‘sentimentr‘ is a valuable package that provides tools to analyze sentiment at the sentence level. It integrates with the ‘tidyverse‘ framework, allowing for easy integration into text analysis pipelines.

10. `lexicon`: The ‘lexicon‘ package offers various lexicons and dictionaries for sentiment analysis and other text mining tasks. It

provides access to predefined lists of words categorized by sentiment, emotion, or other attributes.

These packages collectively provide a comprehensive toolkit for performing various text mining and analysis tasks in R, catering to both basic exploratory analysis and advanced machine learning applications. These packages offer a wide range of functionality and cater to different aspects of text mining and analysis in R. Depending on specific needs and tasks, users can choose the appropriate package or combination of packages to effectively analyze textual data for insights and patterns.

1.2 A Note on NLP

In Natural Language Processing (NLP), word vectors are data-driven representations of words crucial for various tasks. Term Frequency-Inverse Document Frequency (TF-IDF) vectors are used to represent documents based on word frequencies.[**1**] [**2**]Document-Term Matrix (DTM) is a matrix where rows represent documents, columns represent terms, and cells contain the frequency of each term in each document. [**3**] TF-IDF is a common technique for text analytics, creating a sparse matrix of term frequencies.[**4**] In contrast, word vectors are dense representations of words, often derived from linguistic resources like WordNet, enhancing interpretability and performance in NLP tasks. [**5**] These vectors capture semantic relationships and are essential for tasks like document genre classification and disaster tweet classification.[**6**]

Term vectors can be treated as observed variables for statistical analysis. When textual data is converted into term vectors, these vectors represent numerical representations of the text, which can be analyzed using various statistical methods.[**14**] [**15**] Term vectors can be treated as study variables, especially in the context of analyzing the joint dynamics of bond yields and macroeconomic variables.[**7**] By redefining predictor variables as vectors instead of scalars, it becomes possible to handle variables that are not logically or physically separable, ensuring mathematical independence and statistical uncorrelation.[**8**] This approach allows for a more comprehensive analysis of the relationships between different factors, such as inflation, economic growth, and bond prices, leading to improved forecasting performance and a better understanding of the dynamics of the yield curve.[**9**] Additionally, vector autoregression models can provide valuable insights into the interactions between variables, offering a more complete treatment of policy endogeneity compared to other modeling strategies10. Therefore, leveraging term vectors as study variables can enhance the accuracy and depth of economic and financial analyses.

Statistical analysis can be conducted on Document-Term Matrices (DTMs) using techniques like Latent Semantic Analysis (LSA) and Correspondence Analysis (CA).[**11**] LSA employs singular value decomposition to reduce dimensionality and extract important relationships between terms and documents.[**12**] Additionally, incorporating semantic similarities from resources like WordNet can enhance the analysis, as shown in a study on similarity-reduced correspondence analysis.[**13**]

2 R practice

This section handle various tasks related to data mining and analysis through three different sub-sections viz., a trivial example, a moderate example, and non-trivial example. The first, explains processing text in paragraphs assuming each paragraph as a document. Second, explains how to process data from files, where the text is arranged in different rows, assuming text in each row represents a document. Third, explains how to process large amount of data which is obtained from online sources and as abstracts from certain research documents. As it is mentioned earlier, you need to execute `source(easytm.R)` in the R console to get all the functions of the script in to the cache.

2.1 A trivial example

You can switch to locations in your computer using `convertPath()`. This functions makes the task rather easy, it just replace a couple of R base functions i.e., `readline()` and `gsub()`.

```
> getwd()
[1] "C:/Users/kamak/OneDrive/Documents"
> path <- convertPath()
D:\demo
> setwd(path)
> getwd()
[1] "D:/demo"
```

Let's make text collection. I use two rows i.e., *row1* and *row2* and let's imagine that these two rows handles text from two independent documents.

```
row1 <- 'Wikipedia is a free online encyclopedia, created
and edited by volunteers around the world and
hosted by the Wikimedia Foundation.'
row2 <- 'Wikimedia is a global movement whose mission is
to bring free educational content to the world.'
```

2.1.1 Word vectors

R is known for a special function *data.frame()*, this saves quite a bit of time while performing analysis on data sets. All the base data structures such as vectors, factors, etc. need to be converted to data frame to make the job easy. The function `convertAbstractToDataSet()` of `easytm` can convert any piece of text into a *word vector*. The other function `searchWord()` finds the word of interest in the word vector structure.

```
> r1ab <- convertAbstractToDataSet(row1)
> head(r1ab)
     Var1 Freq
1       a    1
2     and    2
3  around    1
4      by    2
5 created    1
6  edited    1
> searchWord('online', r1ab)
     Var1 Freq
12 online    1
```

It is also possible to create DTM from the given pieces of text such as *row1* and *row2*. Let's make the above two rows i.e., `row1, row2` into a data frame. Word vectors can be processed for prepositions such as *a, an, the, ...* using `ceanData()`.

```
> r1ab_ <- cleanData(r1ab)
> head(r1ab_)
           Var1 Freq
3        around    1
5       created    1
6        edited    1
7 encyclopedia,    1
8   Foundation.    1
9          free    1
> searchWord('online', r1ab_)
     Var1 Freq
12 online    1
```

2.1.2 DTMs

It is also possible to make DTMs through data frame.

```
> rdf <- as.data.frame(rbind(row1, row2))
> is.data.frame(rdf)
[1] TRUE
> dim(rdf)
[1] 2 1
> rdf[1, ]
[1] 'Wikipedia is a free online encyclopedia, created
and edited by volunteers around the world and
hosted by the Wikimedia Foundation.'
> rdf[2, ]
[1] 'Wikimedia is a global movement whose mission is
to bring free educational content to the world.'
> names(rdf)
[1] "V1"
```

The data frame is 2×1 data matrix and the column name is `"V1"`, that's the default name R assigns to columns when no input is provided to the function `as.data.frame()`. Now the last step i.e., converting the text into a Document Term Matrix (DTM). That can be done using `cleanCorpusAndMakeDF()` function in `easytm`.

```
> absdf <- cleanCorpusAndMakeDF(rdf)
> names(absdf)
 [1] "around"       "bring"        "content"      "created"      "edited"
 [6] "educational"  "encyclopedia" "foundation"   "free"         "global"
[11] "hosted"       "mission"      "movement"     "online"       "volunteers"
[16] "whose"        "wikimedia"    "wikipedia"    "world"
```

If you see carefully, the output for the statement `names(absdf)` is the list of column names of the DTM. In case, if you doubt, you can verify using `typeof(), class()` function. These are R base functions.

```
> class(absdf)
[1] "data.frame"
> typeof(absdf)
[1] "list"
```

Performing analysis on this data frame is now easy. For instance, a simple summary for the column “online”, assuming it as variable, can be done as below.

```
> absdf <- cleanCorpusAndMakeDF(rdf)
> n_ <- searchPattern('online', absdf)
[1] "online"
> summary(absdf[, n_])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    0.25    0.50    0.50    0.75    1.00
```

2.2 A Moderate example

We don't deal with data within console but collect data in files such as spreadsheets. This section shows the procedure as how data can be obtained from Comma Separated Value (*.csv*) file. I scraped Google search data for a query "how to scrape Google search titles in R" using `rvest` package and the titles were saved in *gtitles.csv.* [2] [3]

```
> filepath <- convertPath()
D:\Work\R Scripts\textmining\easytm
> setwd(filepath)
> titles <- importDataFile(filepath = 'gtitles.csv')
[1] "Doc"
> titles
                                                    Doc
1  Web Scraping in R: The Complete Guide 2024 - ZenRows
2                Scraping Google Search Results Using R
3                     Web Scraping Google Result with R
4                                      google-scraper.R
5             Web Scraping With R: Step-by-Step Tutorial
6                      Web Scraping Google Scholar in R
7             Google News Scraping in R - Pieter E. Stek
8            How to Scrape Google Search Results in 2024
9                                       Web scraping in R
```

[2]You can find the data file in the article companion companion Github portal.

[3]Visit `https://cran.r-project.org/web/packages/rvest/index.html` for more information on web scrapping using `rvest` package.

What is the object type of “titles”?

```
> typeof(titles)
[1] "list"
> class(titles)
[1] "data.frame"
```

It’s a data frame. Good! Now let’s search for required columns I mean those columns in which our research interest lies in. [4]

```
> ttlsdf <- cleanCorpusAndMakeDF(titles["Doc"])
> head(ttlsdf)
  complete google googlescraperr guide news pieter result results scholar scrape
1        1      0              0     1    0      0      0       0       0      0
2        0      1              0     0    0      0      0       1       0      0
3        0      1              0     0    0      0      1       0       0      0
4        0      0              1     0    0      0      0       0       0      0
5        0      0              0     0    0      0      0       0       0      0
6        0      1              0     0    0      0      0       0       1      0
  scraping scrapping search stek stepstep tutorial using web zenrows
1        1         0      0    0        0        0     0   1       1
2        1         0      1    0        0        0     1   0       0
3        1         0      0    0        0        0     0   1       0
4        0         0      0    0        0        0     0   0       0
5        1         0      0    0        1        1     0   1       0
6        1         0      0    0        0        0     0   1       0
```

You don’t need to use whole bunch of functions of `tm` package for mining. Just use `cleanCorpusAndMakeDF()` for mining data sets. Let’s say I am interested in a column (variable) called “scrape”. I may be able to calculate summary statistics as shown below.

```
> names(ttlsdf)
 [1] "complete"       "google"         "googlescraperr" "guide"          "news"
 [6] "pieter"         "result"         "results"        "scholar"        "scrape"
[11] "scraping"       "scrapping"      "search"         "stek"           "stepstep"
[16] "tutorial"       "using"          "web"            "zenrows"
> grep('scrape', names(ttlsdf))
[1]  3 10
> names(ttlsdf[, c(3, 10)])
[1] "googlescraperr" "scrape"
> summary(ttlsdf[, c(3, 10)])
```

[4]Columns in DTM are called Term Frequency Vectors (TFV) and they are considered as variables for investigation and analysis. Usually variables are obtained using a *pattern*. A pattern is something close to the concept called “factor”

```
 googlescraperr      scrape
 Min.   :0.0    Min.   :0.0
 1st Qu.:0.0    1st Qu.:0.0
 Median :0.0    Median :0.0
 Mean   :0.1    Mean   :0.1
 3rd Qu.:0.0    3rd Qu.:0.0
 Max.   :1.0    Max.   :1.0
```

There are two data variables (TFVs), namely “googlescraperr”, “scrape”, which match the pattern “scrape” in the text corpus. Summaries shows that the maximum occurrence of the factor is one and there are a few tiles where the presence is zero (Min. 0).

2.3 Non trivial example

This section shows how a data file with significant size can be processed using `easytm`. I shall demonstrate a couple of worthy functions for obtaining the file, transform raw text into other formats which are amenable for analysis. I will be using a file consisting of a few abstracts retrieved from a known database such as Scopus and turning such abstract or abstracts into a data variables or matrices worthy for investigation. At first I will try a single abstract, for this I tried searching for abstract using a search string “marketing is fake”. Google scholar retrieved about 4,64,000 results in 0.07 seconds. I would like to know how many times words “marketing” and “fake” were repeated in the very first abstract.[5] [**16**]

```
firstab <- 'There is growing concern amongst policy makers, managers and
            academic researchers over the role that social media plays in
            spreading misinformation, widely described as ‘Fake News’.
            However, research to date has mainly focussed on the implications
            of fake news for political communication and debate. There has
            been less focus on the implications of social media misinformation
            upon marketing and consumers. Given the key role of social media
            as a communication platform, there is a gap in our understanding
            of fake news through a consumer lens. We address this gap by
            conducting an interdisciplinary systematic review of the relevant
            literature. Through critical evaluation and synthesis of the
            literature, we identify five themes that explain the fake news
            phenomenon: the dissemination process, spreading channel features,
            outcomes, fabricated legitimacy and attitudes. Finally,
            we propose a theoretical framework that highlights themes'
```

[5]The task was performed on 30th June, 2024

```
            relationships and research propositions to guide future research
            in this area.'
> opf <- convertAbstractToDataSet(firstab)
> opf <- cleanData(op_)
> searchWord('market', opf)
        Var1 Freq
59 marketing    1
```

The output for `convertAbstractToDataSet()` is a data frame of 101 × 2 order, the first column (Var1) is a vector of words (Freq) and the second column has the frequencies for each word in the first column. The word *fake* repeated 3 times in the abstract but the other word marketing occurred or mentioned only one time. Hence, it is clear that there is more emphasis on "fake" than "marketing" by the authors in this research article. Let's do the same exercise for the second abstract.[**17**]

```
> secondab <- 'We study the market for fake product reviews on Amazon.com.
+               Reviews are purchased in large private groups on Facebook
+               and other sites. We hand-collect data on these markets and
+               then collect a panel of data on these products’ ratings
+               and reviews on Amazon, as well as their sales rank,
+               advertising, and pricing policies. We find that a wide
+               array of products purchase fake reviews, including products
+               with many reviews and high average ratings. Buying fake
+               reviews on Facebook is associated with a significant but
+               short-term increase in average rating and number of reviews.
+               We exploit a sharp but temporary policy shift by Amazon to
+               show that rating manipulation has a large causal effect on
+               sales. Finally, we examine whether rating manipulation harms
+               consumers or whether it is mainly used by high-quality
+               products in a manner like advertising or by new products
+               trying to solve the cold-start problem. We find that after
+               firms stop buying fake reviews, their average ratings fall
+               and the share of one-star reviews increases significantly,
+               particularly for young products, indicating rating manipulation
+               is mostly used by low-quality products.'
> ops <- convertAbstractToDataSet(secondab)
> searchWord('market', ops)
      Var1 Freq
55  market    1
56 markets    1
> searchWord('fake', ops)
   Var1 Freq
29 fake    4
```

The function `cleanData()` can be used to clean unwanted words such as prepositions. It is also possible to make the word plot using `plotWordVec()`.1

```
> clnab <- cleanData(opf)
> dim(clnab)
[1] 19  2
> head(clnab)
       Var1 Freq
1            247
15  average    3
24     data    2
28 Facebook    2
29     fake    4
48    large    2
> plotWordVec(clnab[, 2], lbls = clnab[, 1])
```

It is also possible to compare insights from two different documents.

```
> makeDfFromWord('fake', clnabf, clnabs)
   Var1 Freq
d1 fake    3
d2 fake    4
```

Di Domenico et al. (2021),[**16**] mentioned the word "fake" less number of times compared to He et al., (2022)[**17**].

2.3.1 Processing large data sets

In this section the procedure for processing data from Scopus is demonstrated. The search query "marketing is fake"has retrieved 539 abstracts.[6] The data was saved in CSV file with a name "scopus.csv". Let's process the data file this time a bit differently. Differently I mean, I am planning to make a DTM instead of word/term vectors.

```
> datafile <- read.csv('scopus.csv')
> abs <- makeAbstracts(datafile)
data.frame with 539 rows and 1 columns were created
> df_ <- cleanCorpusAndMakeDF(abs, 0.99)
```

[6]30th June, 2024

```
> head(names(df_))
[1] "able"     "abnormal" "abortion" "abruptly" "absence"  "absorbed"
```

The function `makeAbstracts()` retrieves all those abstracts from the data file (*scopus.csv* in this case).[7] The most important function is perhaps `cleanCorpusAndMakeDF()` which not only creates DTM but ensures that all the unwanted text in the corpus will be removed.

2.4 Hypotheses and Errors

The textual patterns mined from abstracts can be used for testing a few hypotheses together with corresponding errors. The package `heplots` may be useful for such analysis. The following code shows about influence of *fintech* on *consumer satisfaction.*

```
tech <- searchVariable('tech', df_, output = T)
mark <- searchVariable('mareketing', df_, output = T)
sat <- searchVariable('satisfaction', df_, output = T)
cons <- searchVariable('consumer', df_, output = T)
mkttechsat <- data.frame(tech, mark, cons, sat)
names(mkttechsat)
lmfit <- lm(cbind(consumer, satisfaction)~fintech, data = mkttechsat)
summary(lmfit)
library(heplots)
heplot(lmfit)
```

The function `searchVariable()` can make data-frames for given word pattern also known as conceptual patterns. The objects *tech, mark, sat, cons* are data frames for conceptual patterns *technology, marketing, consumer*, and *satisfaction.* The summary of the above linear regression results are as follows.

```
> summary(lmfit)
Response consumer :
Call:
lm(formula = consumer ~ fintech, data = mkttechsat)
Residuals:
    Min      1Q  Median      3Q     Max
-0.2138 -0.2138 -0.2138 -0.2138  6.7862
Coefficients:
```

[7]The file is available at the companion Github portal.

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.21375    0.02956   7.232 1.65e-12 ***
fintech     -0.21375    0.68624  -0.311    0.756
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 0.6856 on 537 degrees of freedom
Multiple R-squared:  0.0001806,Adjusted R-squared:  -0.001681
F-statistic: 0.09702 on 1 and 537 DF,  p-value: 0.7556
Response satisfaction :
Call:
lm(formula = satisfaction ~ fintech, data = mkttechsat)
Residuals:
     Min       1Q   Median       3Q      Max
-0.00929 -0.00929 -0.00929 -0.00929  0.99071
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.009294   0.004141   2.244   0.0252 *
fintech     -0.009294   0.096133  -0.097   0.9230
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 0.09604 on 537 degrees of freedom
Multiple R-squared:  1.74e-05,Adjusted R-squared:  -0.001845
F-statistic: 0.009346 on 1 and 537 DF,  p-value: 0.923
```

Loot at Figure 2 for HE plot for above analysis. While the estimates are negative but insignificant. So, *Fintech* does not appear to be influential in the literature.

Suppose, the interest of the research is to test impact of information systems on consumer satisfaction, then the hypothesis can be tested as shown below.

```
info <- searchVariable('information', df_, output =T)
sys <- searchVariable('systems', df_, output = T)
consatinfosys <- data.frame(cons, sat, info, sys)
names(consatinfosys)
lmfit <- lm(cbind(consumer, satisfaction)~information+ systems, data = consatinfosys)
summary(lmfit)
Response consumer :
Call:
lm(formula = consumer ~ information + systems, data = consatinfosys)
Residuals:
    Min      1Q  Median      3Q     Max
-0.6667 -0.2056 -0.2056 -0.2056  6.7330
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.20555    0.03300   6.230 9.45e-10 ***
information  0.03075    0.02233   1.377    0.169
systems     -0.11230    0.07405  -1.517    0.130
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 0.6841 on 536 degrees of freedom
Multiple R-squared:  0.006479,Adjusted R-squared:  0.002772
```

```
F-statistic: 1.748 on 2 and 536 DF,  p-value: 0.1752
Response satisfaction :
Call:
lm(formula = satisfaction ~ information + systems,
data = consatinfosys)
Residuals:
     Min       1Q   Median       3Q      Max
-0.01163 -0.01163 -0.01163 -0.00860  0.98837
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.011632   0.004631   2.512   0.0123 *
information -0.003034   0.003135  -0.968   0.3336
systems     -0.003914   0.010393  -0.377   0.7067
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 0.09602 on 536 degrees of freedom
Multiple R-squared:  0.002378,Adjusted R-squared:  -0.001345
F-statistic: 0.6387 on 2 and 536 DF,  p-value: 0.5284
```

The influence of information systems in the literature seems to be negative despite of the fact that information affects consumer satisfaction positively. However, all the influences are statistically insignificant. Loot at the Figure 3 for visual. This shows that the opinion in the literature is positive for ideas related to *information* but negative for *systems*. So, while *information* seems to be important for consumer satisfaction but the same is not true for *information systems*.

2.5 Causal relationships

The package `lavaan` is highly usefull for testing causal relationships using Confirmatory Factor Analysis. The following codes deals with testing hypothesis for moderation effect from innovation.

```
library(lavaan)
inno <- searchVariable('innova', df_, output = T)
cfadf <- data.frame(cons, sat, info, sys, inno)

model <- '
  # regressions
  consumer + satisfaction ~ a * information + b * systems
  consumer + satisfaction ~ c * innovation
  # effects
  p := c * a
  q := c * b
  r := p * q
'
fit <- sem(model, scale(cfadf))
summary(fit)
```

The output for the above model is as follows.

```
> parameterestimates(fit)
            lhs op          rhs label    est    se      z pvalue ci.lower ci.upper
1      consumer  ~  information     a  0.006 0.032  0.191  0.848   -0.057    0.069
2      consumer  ~      systems     b -0.041 0.031 -1.320  0.187   -0.102    0.020
3  satisfaction  ~  information     a  0.006 0.032  0.191  0.848   -0.057    0.069
4  satisfaction  ~      systems     b -0.041 0.031 -1.320  0.187   -0.102    0.020
5      consumer  ~   innovation     c  0.011 0.032  0.338  0.736   -0.051    0.072
6  satisfaction  ~   innovation     c  0.011 0.032  0.338  0.736   -0.051    0.072
7      consumer ~~     consumer        0.994 0.061 16.416  0.000    0.875    1.113
8  satisfaction ~~ satisfaction        0.999 0.061 16.416  0.000    0.879    1.118
9      consumer ~~ satisfaction       -0.004 0.043 -0.085  0.932   -0.088    0.080
16            p :=          c*a     p  0.000 0.000  0.190  0.850   -0.001    0.001
17            q :=          c*b     q  0.000 0.001 -0.332  0.740   -0.003    0.002
18            r :=          p*q     r  0.000 0.000 -0.145  0.885    0.000    0.000
```

The results shows that the discussions on impact of information systems does not seems to be significant in the literature. However, there are a few positive relationships. For instance, consumer satisfaction appear to be influenced by information and innovation but such ideas are not strong in the literature.

3 Conclusion

References

[1] Wei, Li., Brian, Mak. (2017). Derivation of Document Vectors from Adaptation of LSTM Language Model. doi: 10.18653/V1/E17-2073.

[0] [2] ei, Li., Brian, Kan., Wing, Mak. (2016). Recurrent Neural Network Language Model Adaptation Derived Document Vector.. arXiv: Computation and Language.

[3] austubh, Keshav. (2022). Term Frequency Based Approach for Binary Classi?cations on Short Sentences. International Journal For Science Technology And Engineering, doi: 10.22214/ijraset.2022.47151

[4] amah, Senbel. (2021). Fast and Memory-Efficient TFIDF Calculation for Text Analysis of Large Datasets. doi: 10.1007/978-3-030-79457-6$_4$$7anaal, Faruqui., Chris, Dyer.(2015).Non - distributionalWordVectorRepresentations.arXiv : ComputationandLanguage,$

[5[6] uillaume, Desagulier. (2019). Can word vectors help corpus linguists. Studia Neophilologica, doi: 10.1080/00393274.2019.1616220.

[7] ndrew, Ang., Andrew, Ang., Monika, Piazzesi., Monika, Piazzesi. (2001). A No-Arbitrage Vector Autoregression of Term Structure Dynamics with Macroeconomic and Latent Variables. Research Papers in Economics,

[8] ohn, R., Freeman., John, T., Williams., Tse-min, Lin. (1989). Vector Autoregression and the Study of Politics. American Journal of Political Science, doi: 10.2307/2111112.

[9] ., T., Mcadams., R., W., Crawford., G., R., Hadder. (2000). A vector approach to regression analysis and its application to heavy-duty diesel emissions. SAE transactions, doi: 10.4271/2000-01-1961.

[10] eter, Reusens., Christophe, Croux. (2015). Real or nominal variables, does it matter for the impulse response?. Social Science Research Network, doi: 10.2139/SSRN.2577815.

[11] r., Deepali, , Jadhav, -Jagtap. (2023). A comparison of latent semantic analysis and correspondence analysis of document-term matrices. Natural Language Engineering, doi: 10.1017/s1351324923000244.

[12] ahrettin, Horasan., Hasan, Erbay., Fatih, Varçın., Emre, Deniz. (2019). Alternate Low-Rank Matrix Approximation in Latent Semantic Analysis. Scientific Programming, doi: 10.1155/2019/1095643.

[13] attia, Egloff., François, Bavaud. (2018). Taking into account semantic similarities in correspondence analysis.

[14] lei, D. M., Ng, A. Y., Jordan, M. I. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research, 3, 993-1022.

[15] umais, S. T. (2004). Latent semantic analysis. Annual Review of Information Science and Technology, 38(1), 188-230. https://doi.org/10.1002/aris.1440380105.

[16] i Domenico, G., Sit, J., Ishizaka, A., Nunan, D. (2021). Fake news, social media and marketing: A systematic review. Journal of Business Research, 124, 329-341.

[17] e, S., Hollenbeck, B., Proserpio, D. (2022). The market for fake reviews. Marketing Science, 41(5), 896-921.

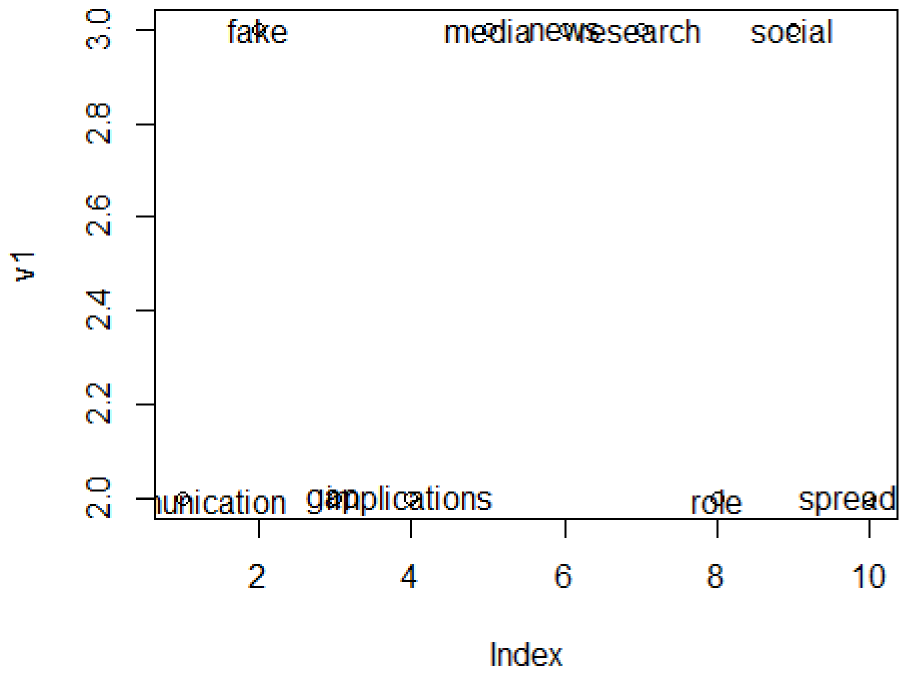

Figure 1: Word vector plot

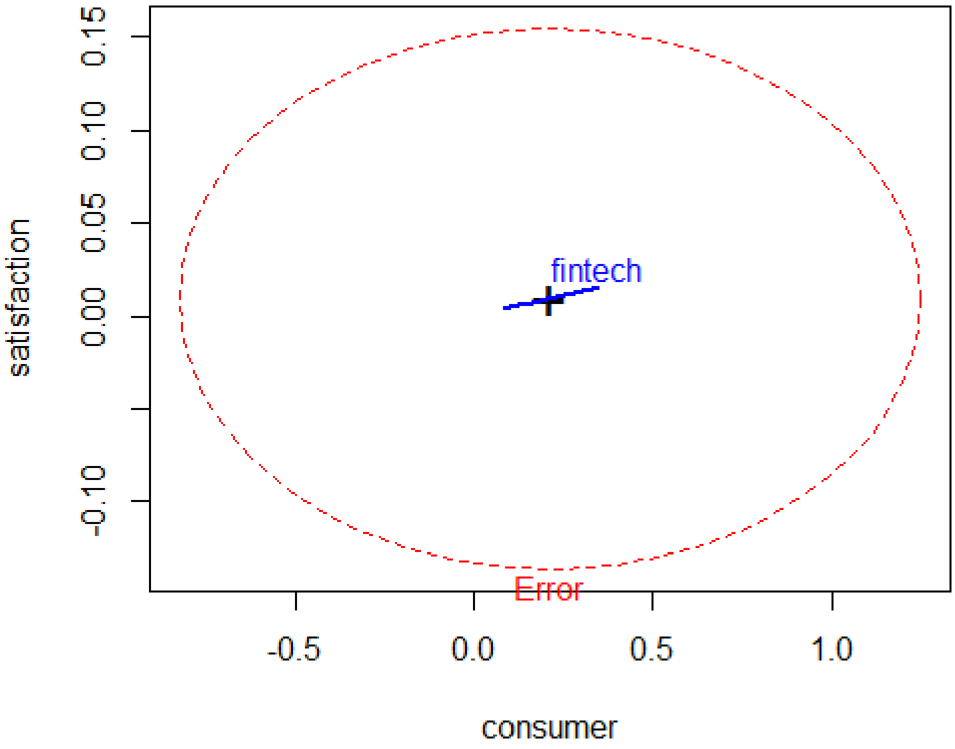

Figure 2: HE Plot for consumer satisfaction vs. fintech

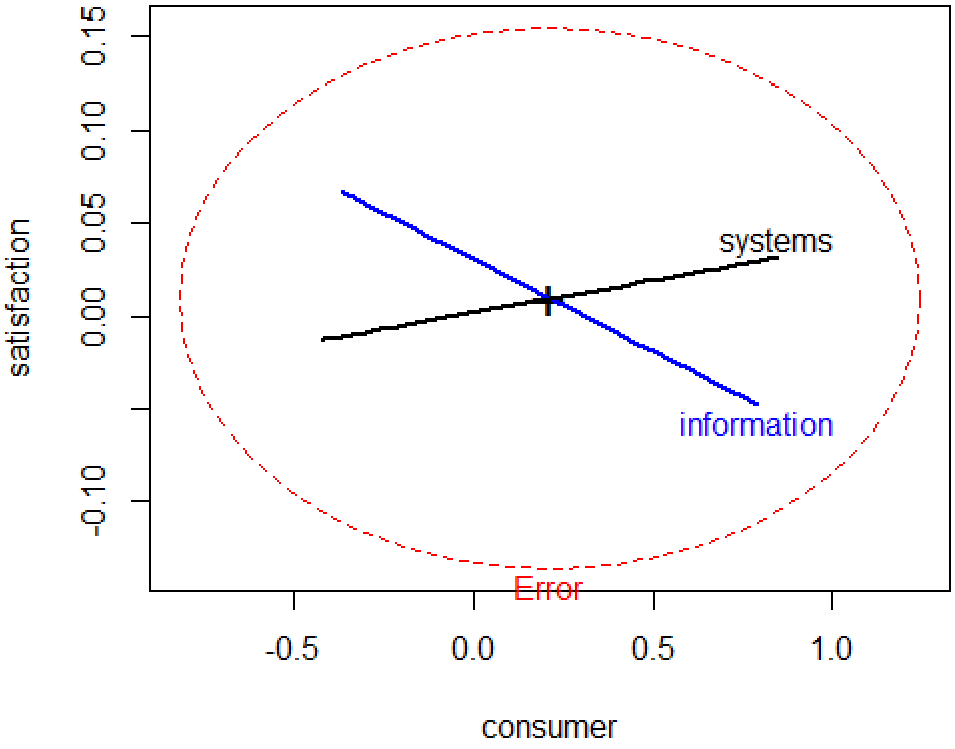

Figure 3: HE Plot for consumer satisfactin vs. information systems