Text Mining

Bag of Words

Term Frequency

Inverse Term Frequency

Reviews of Online article

- Review 1: This page is lengthy.
- Review 2: This article is good and presented well.
- Review 3: This article gave insight about the subject.
- Reviews : Are they positive or negative?
  - Requires text preprocessing
  - Bag-of-Words and TF-IDF are two examples of how to do this

Creating Vectors from Text

- High computation cost
  - It should not result in a sparse matrix
  - sparse matrices result in high computation cost.
- Able to retain most of the linguistic information.
  - Word Embedding is one such technique where we can represent the text using vectors.
  - BoW, which stands for Bag of Words
  - TF-IDF, which stands for Term Frequency-Inverse Document Frequency

Bag of Words (BoW) Model

- The Bag of Words (BoW) model is the simplest form of text representation in numbers.
- Like the term, Represent a sentence as a bag of words vector (a string of numbers).
  - Review 1: This page is lengthy.
  - Review 2: This article is good and presented well.
  - Review 3: This article gave insight about the subject.
- First build a vocabulary from all the unique words in the above three reviews.

- Review 1: This page is lengthy.
- Review 2: This article is good and presented well.
- Review 3: This article gave insight about the subject.

- The vocabulary are
  - This, page, is, lengthy, good, article, and, presented, well, gave, insight, about, the, subject.

– Review 1: This page is lengthy.

– Review 2: This article is good and presented well.

– Review 3: This article gave insight about the subject.

- This will give us 3 vectors for 3 reviews:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	This	page	is	lengt hy	good	articl e	and	prese nted	well	gave	insig ht	abou t	the	subje ct
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
2	1	0	1	0	1	1	1	1	1	0	0	0	0	0
3	1	0	0	0	0	1	0	0	0	1	1	1	1	1

Take each of these words and mark their occurrence in the three reviews above with 1s and 0s

The three vectors

- Vector for Review -1
- Vector for Review -1
- Vector for Review -1

Drawbacks

- issues arises when new sentences comes.
- If the new sentences contain new words, then vocabulary size would increase and thereby, the length of the vectors would increase too.
- Additionally, the vectors would also contain many 0s, thereby resulting in a sparse matrix.
- The information on the grammar of the sentences nor on the ordering of the words in the text is retained.

Term Frequency-Inverse Document Frequency (TF-IDF)

- “Term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.”

Term Frequency (TF)

- It is a measure of how frequently a term, t, appears in a document, d:

$$tf_{t,d} = \frac{n_{t,d}}{Number\ of\ terms\ in\ the\ document}$$

Here, in the numerator, n is the number of times the term "t" appears in the document "d". Thus, each document and term would have its own TF value.

We will again use the same vocabulary we had built in the Bag-of-Words model to show how to calculate the TF for Review #2:

Term Frequency (TF)

- Here,
  - Vocabulary**:** This, page, is, lengthy, good, article, and, presented, well, gave, insight, about, the, subject.
  - Review 2: This article is good and presented well.
- Number of words in Review 2 = 7
- TF for the word ‘this’ = (number of times ‘this’ appears in review 2)/(number of terms in review 2) = 1/7

Review 1: This page is lengthy.
Review 2: This article is good and presented well.
Review 3: This article gave insight about the subject.

Term	Review1	Review2	Review3	TF1	TF2	TF3
This	1	1	1	¼	1/7	1/7
page	1	0	0	1/4	0	0
is	1	1	0	¼	1/7	0
lengthy	1	0	0	1/4	0	0
good	0	1	0	0	1/7	0
article	0	1	1	0	1/7	1/7
and	0	1	0	0	1/7	0
presented	0	1	0	0	1/7	0
gave	0	0	1	0	0	1/7
Insight	0	0	1	0	0	1/7
about	0	0	1	0	0	1/7
the	0	0	1	0	0	1/7
subject	0	0	1	0	0	1/7
well	0	1	0	0	1/7	0

Inverse Document Frequency (IDF)

$$idf_t = \log \frac{number\ of\ documents}{number\ of\ documents\ with\ term\ 't'}$$

- IDF is a measure of how important a term is. We need the IDF value because computing just the TF alone is not sufficient to understand the importance of words
- We can calculate the IDF values for the all the words in Review 2:
- IDF('this') = log(number of documents/number of documents containing the word 'this') = log(3/3) = log(1) = 0

Review 1: This page is lengthy.
Review 2: This article is good and presented well.
Review 3: This article gave insight about the subject.

- We can calculate the IDF values for the all the words in Review 2:
- IDF(‘this’) = log(number of documents/number of documents containing the word ‘this’) = log(3/3) = log(1) = 0

$$idf_t = \log \frac{number\ of\ documents}{number\ of\ documents\ with\ term\ 't'}$$

Inverse Document Frequency (IDF)

Term	Review1	Review2	Review3	IDF
This	1	1	1	0.0
page	1	0	0	log(3/1)
is	1	1	0	log(3/2)
lengthy	1	0	0	
good	0	1	0	
article	0	1	1	
and	0	1	0	
presented	0	1	0	
gave	0	0	1	
Insight	0	0	1	
about	0	0	1	
the	0	0	1	
subject	0	0	1	
well	0	1	0	

- **Hence, we see that words like “is”, “this”, “and”, etc., are reduced to 0 and have little importance; while words like “presented”, “insight”, “good”, etc. are words with more importance and thus have a higher value.**
- We can now compute the TF-IDF score for each word in the corpus. Words with a higher score are more important, and those with a lower score are less important:

$$(tf\_idf)_{t,d} = tf_{t,d} * idf_t$$

$$(tf\_idf)_{t,d} = tf_{t,d} * idf_t$$

- We can now calculate the TF-IDF score for every word in Review 2:
- TF-IDF(‘this’, Review 2) = TF(‘this’, Review 2) * IDF(‘this’) = (1/7) * 0 = 0

Review 1: This page is lengthy.
Review 2: This article is good and presented well.
Review 3: This article gave insight about the subject.

Term	Review1	Review2	Review3	IDF	TF-IDF1	TF-IDF2	TF-IDF3
This	1	1	1	0.0			
page	1	0	0	log(3/1)			
is	1	1	0	log(3/2)			
lengthy	1	0	0				
good	0	1	0				
article	0	1	1				
and	0	1	0				
presented	0	1	0				
gave	0	0	1				
Insight	0	0	1				
about	0	0	1				
the	0	0	1				
subject	0	0	1				
well	0	1	0				

Conclusion

- obtained the TF-IDF scores for our vocabulary.
- *TF-IDF also gives larger values for less frequent words and is high when both IDF and TF values are high
- *i.e the word is rare in all the documents combined, but frequent in a single document.

Summary

- Bag of Words just creates a set of vectors containing the count of word occurrences in the document (reviews), while the TF-IDF model contains information on the more important words and the less important ones as well.
- Bag of Words vectors are easy to interpret.
- However, TF-IDF usually performs better in machine learning models.

- While both Bag-of-Words and TF-IDF have been popular in their own regard, there still remained a void where understanding the context of words was concerned.
- Detecting the similarity between the words ‘spooky’ and ‘scary’, or translating our given documents into another language, requires a lot more information on the documents.

- Hence Word Embedding techniques such as Word2Vec,

  Continuous Bag of Words (CBOW),

  Skipgram.

Text Mining

- highlights the most frequently used keywords in a paragraph of texts.
- Used to create a word cloud, also referred as text cloud or tag cloud, which is a visual representation of text data.
- R Packages: To analyze texts and Visualize keywords as word clouds
  - The text mining package (tm), and
  - The word cloud generator package (wordcloud)

Benefits

- Add simplicity and clarity. The most used keywords stand out better in a word cloud
- A potent communication tool. They are easy to understand, to be shared and are impactful
- Are visually engaging than a table data

Applications

- **Researchers :** for reporting qualitative data
- **Marketers :** for highlighting the needs and pain points of customers
- **Educators :** to support essential issues
- Politicians and journalists
- **social media sites :** to collect, analyze and share user sentiments

The steps involved

- Step 1: Create a text file
- Step 2 : Install and load the required packages
- Step 3 : Text mining
- Step 4 : Build a term-document matrix
- Step 5 : Generate the Word cloud

R Packages

- # Install
- install.packages("tm") # for text mining
- install.packages("SnowballC") # for text stemming
- install.packages("wordcloud") # word-cloud generator
- install.packages("RColorBrewer") # color palettes
- # Load
- library("tm")
- library("SnowballC")
- library("wordcloud")
- library("RColorBrewer")

Load Data

- **Method – 1 – File dialogue**

```
text <- readLines(file.choose())
```

- Read the text file from internet
- **Method - 2**

```
filePath <-
"http://www.sthda.com/sthda/RDoc/example-
files/martin-luther-king-i-have-a-dream-speech.txt"
text <- readLines(filePath)
```

- **Method – 3**

```
setwd("F:/R/Day-4")
getwd()
text <- readLines("Blockchain.txt")
text
```

VectorSource()

- Description: Create a vector source.

- Usage

VectorSource(x)

Arguments

x : A vector giving the texts.

Details

A vector source interprets each element of the vector x as a document.

Interpretation of input text

- interprets each element of the vector x as a document
  - docs <- Corpus(VectorSource(text))
  - inspect(docs)

Text transformation

- Transformation is performed using tm_map() function to replace,

for example, special characters from the text.

- Replacing “/”, “@” and “|” with space:

```
toSpace <- content_transformer(function (x , pattern
) gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "/")
docs <- tm_map(docs, toSpace, "@")
docs <- tm_map(docs, toSpace, "\\|")
```

Cleaning Text

```
# Convert the text to lower case
docs <- tm_map(docs, content_transformer(tolower))
# Remove numbers
docs <- tm_map(docs, removeNumbers)
# Remove english common stopwords
docs <- tm_map(docs, removeWords, stopwords("english"))
# Remove your own stop word
# specify your stopwords as a character vector
docs <- tm_map(docs, removeWords, c("blabla1", "blabla2"))
# Remove punctuations
docs <- tm_map(docs, removePunctuation)
# Eliminate extra white spaces
docs <- tm_map(docs, stripWhitespace)
```

Build a term-document matrix

- ?TermDocumentMatrix()

Description :

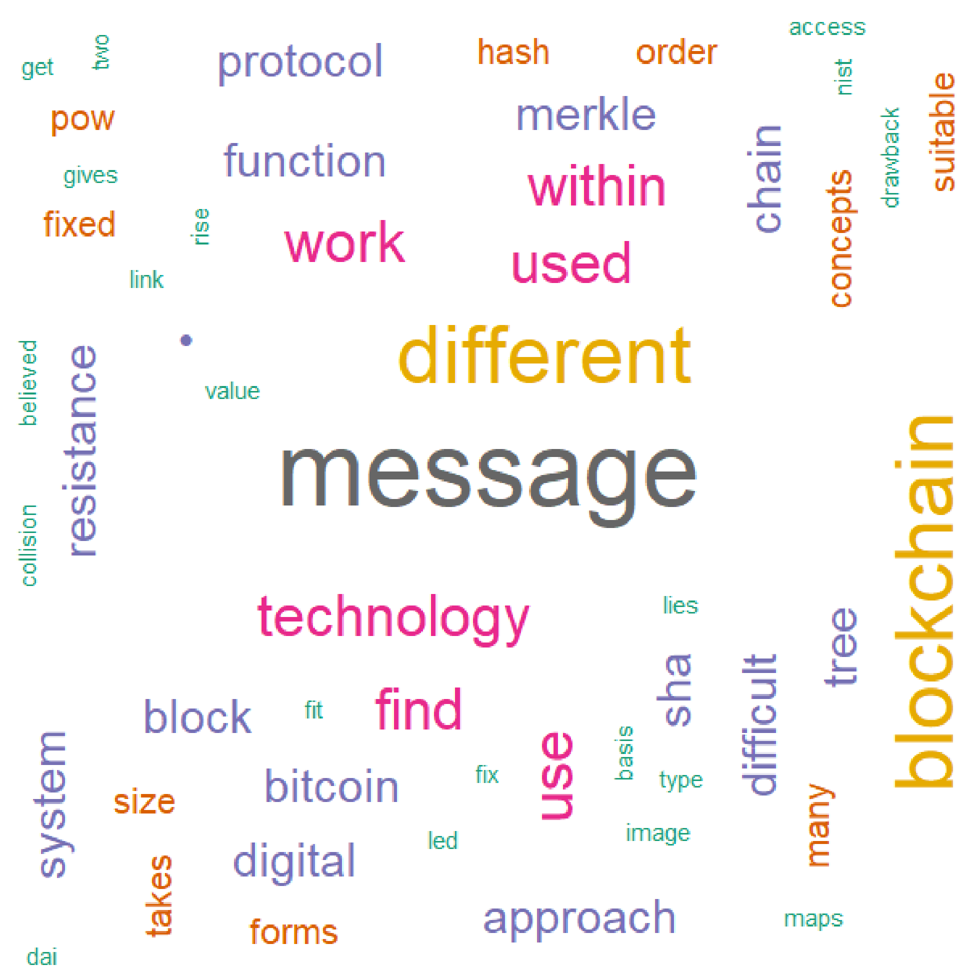
Constructs or coerces to a term-document matrix or a document-term matrix.

WORDCLOUD

```
?wordcloud(): Plot a word cloud
wordcloud(words = d$word, freq = d$freq, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```

Arguments of the word cloud generator function :
words : the words to be plotted
freq : their frequencies
min.freq : words with frequency below min.freq will not be plotted
max.words : maximum number of words to be plotted
random.order : plot words in random order. If false, they will be plotted in decreasing frequency
rot.per : proportion words with 90 degree rotation (vertical text)
colors : color words from least to most frequent. Use, for example, colors =“black” for single color.

Wordcloud



message
blockchain
different
technology
within
used
work
find
use
protocol
function
merkle
chain
resistance
block
bitcoin
digital
approach
system
sha
difficult
tree
hash
order
pow
fixed
size
takes
forms
concepts
suitable
many
access
get
two
gives
rise
link
nist
drawback
value
believed
collision
lies
fit
fix
basis
type
led
image
maps
dai

Wordcloud

tomorrow
slopes
selfevident
thank
oppression
hilltops
must
pennsylvania
hamlet
injustice
lookout
south
character
still
jews
meaning
rise
curvaceous
beautiful
struggle
york
mountainside
brotherhood
new
every
american
owners
discords
sing
made
equal
great
join
god
live
places
nullification
dripping
racists
truths
molehill
despair
table
able
little
free
boys
low
back
rough
knowing
children
difficulties
crooked
interposition
skin
exalted
almighty
work
today
village
alleghenies
pray
freedom
new
sons
snowcapped
tennessee
jangling
gentiles
hope
alabama
nation
true
jail
mountain
rockies
deeply
dream
will
let
state
glory
brothers
colorado
valley
mighty
former
land
lord
see
girls
right
stone
tis
lips
symphony
sit
hold
allow
ring
day
last
creed
justice
white
even
fathers
sweet
words
hill
heat
liberty
happens
transformed
flesh
color
plain
become
california
spiritual
negro
old
thee
one
faith
black
created
face
died
four
prodigious
hands
georgia
america
city
revealed
though
together
shall
pilgrim
judged
vicious
men
hills
stand
red
country
mississippi
content
slaves
governor
sweltering
catholics
pride
rooted
oasis
protestants
hampshire
slave
sisters
heightening
transform
mountains
speed
straight