



Basic Data Mining with n-Gram

Sin Ying Wong

10/30/2019

Why use this topic?

- During Project 3 “Which are the most valued data science skills”, I had my first experience in web scraping.
- Wondered how to get high-frequency phrases with different length from a long paragraph, instead of having “machine” and “learning” counted separately.
- N-gram was covered in Chapter 10: Representing and Mining Text.

Introductions

Introduction

- Bag-of-words:
A text, such as a sentence or a document, being as a bag of its words.
- N-gram: Context by breaking long text into sequences of adjacent words as terms, disregarding word order entirely.
e.g. bi-grams: sequence of 2 words, tri-grams: sequence of 3 words
- We would like to see phrase like “natural language processing” which is more meaningful than simply knowing the individual words “language”, “natural”, and “processing”.

RStudio - Part 1

Libraries and Functions

Libraries and Functions

Libraries used

- readtext
- tidyverse
- tm
- RWeka
- SnowballC

Functions used

- NGramTokenizer
- VCorpus (from tm ver.7.0)
- content_transform
- TermDocumentMatrix

Create functions for n-grams

```
BigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 2, max = 2))
TrigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 3, max = 3))
FourgramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 4, max = 4))
FivegramTokenizer <- function(x,n) NGramTokenizer(x, Weka_control(min = 5, max = 5))
removeURL <- function(x) str_replace_all(x,"http[[:alnum:]]*", "")
```

- BigramTokenizer: for phrase with length = 2
- TrigramTokenizer: for phrase with length = 3
- FourgramTokenizer: for phrase with length = 4
- FivegramTokenizer: for phrase with length = 5

Create a VCorpus for tidying data

```
text_corpus <- VCorpus(VectorSource(text)) %>%
  tm_map(removeNumbers) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeWords, stopwords("en")) %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(content_transformer(removeURL)) %>%
  tm_map(stripWhitespace)
```

- To remove numbers, punctuations,
- To remove stop words, e.g. myself, which, would, because, among, etc.
- To turn all alphabets to lower case
- To remove all URLs in the paragraph
- To strip extra whitespace from a text document. Multiple whitespace characters are collapsed to a single blank

Part 2

Obtaining n-grams

Tri-gram

Code:

```
key_n_gram <- character()
tdm.trigram <- TermDocumentMatrix(text_corpus,
                                   control = list(wordLengths = c(10, Inf),
                                                  tokenize = TrigramTokenizer))
#inspect(tdm.bigram)

freq.tdm.trigram <- data.frame(word = tdm.trigram$dimnames$Terms, frequency = tdm.trigram$v, stringsAsFactors
= FALSE) %>%
  arrange(-frequency)

freq.tdm.trigram
```

A phrase containing 3 words
while each has length ≥ 3

Result:

word <chr>	frequency <dbl>
natural language processing	110
nlp machine learning	47
learning nlp machine	46
machine learning nlp	46
association computational linguistics	24
language processing almost	23
almost from scratch	22
bottou karlen kavukcuoglu	22
collobert weston bottou	22
karlen kavukcuoglu and	22

1-10 of 10,000 rows

Previous 1 2 3 4 5 6 ... 1000 Next

- I took “natural language processing” in consideration.

Bi-gram

Code:

```
tdm.bigram <- TermDocumentMatrix(text_corpus,  
                                  control = list(wordLengths = c(8,Inf),  
                                                  tokenize = BigramTokenizer))  
  
#inspect(tdm.bigram)  
  
freq.tdm.bigram <- data.frame(word = tdm.bigram$dimnames$Terms, frequency = tdm.bigram$v, stringsAsFactors = F  
ALSE) %>%  
  arrange(-frequency)  
  
freq.tdm.bigram
```

Result:

word <chr>	frequency <dbl>
natural language	159
machine learning	141
neural networks	118
language processing	117
language model	88
nlp machine	47
learning nlp	46
computational linguistics	35
parse tree	30
lookup table	29

1-10 of 10,000 rows

Previous 1 2 3 4 5 6 ... 1000 Next

- I took top 10 phrases in consideration.

Four- and Five-gram

```
tdm.4gram <- TermDocumentMatrix(text_corpus,  
                                control = list(wordLengths = c(14,Inf),  
                                              tokenize = FourgramTokenizer))  
  
#inspect(tdm.bigram)  
  
freq.tdm.4gram <- data.frame(word = tdm.4gram$dimnames$Terms, frequency = tdm.4gram$v) %>%  
  arrange(-frequency)  
  
freq.tdm.4gram
```

word	frequency
<fctr>	<dbl>
learning nlp machine learning	46
machine learning nlp machine	46
nlp machine learning nlp	46
natural language processing almost	23
bottou karlen kavukcuoglu and	22
collobert weston bottou karlen	22
karlen kavukcuoglu and kuksa	22
language processing almost from	22
processing almost from scratch	22
weston bottou karlen kavukcuoglu	22

1-10 of 10,000 rows

Previous 1 2 3 4 5 6 ... 1000 Next

- 4-gram did not show meaningful results and so will 5-gram, therefore I skipped all these phrases.

Part 3

Concatenate n-grams by “_”

Concatenating n-grams by “_” in the paragraph

```
text_corpus_mod <- text_corpus

for (key in key_n_gram){
  text_corpus_mod <- text_corpus_mod[[1]]$content %>%
    str_replace_all(key, str_replace_all(key, ' ', '_')) %>%
    VectorSource() %>%
    VCorpus()
}
text_corpus_mod
```

- This code chunk will help me concatenating the 11 n-grams (1 tri-gram and 10 bi-gram) I put in `key_n_gram` in the tidied paragraph

Part 4

Output

Outputs:

Before concatenating:

```
tdm <- TermDocumentMatrix(text_corpus, control = list(wordLengths = c(4, Inf)))
#inspect(tdm2)
tdm.word <- data.frame(word = tdm$dimnames$Terms, frequency = tdm$v, stringsAsFactors = FALSE) %>%
  arrange(-frequency)

tdm.word
```

word <chr>	frequency <dbl>
language	326
learning	207
natural	168
machine	162
neural	158
word	157
networks	156
processing	146
using	129
model	125

1-10 of 3,653 rows Previous 1 2 3 4 5 6 ... 366 Next

After concatenating:

```
tdm.mod <- TermDocumentMatrix(text_corpus_mod, control = list(wordLengths = c(4, Inf)))
#inspect(tdm2)
tdm.mod.word <- data.frame(word = tdm.mod$dimnames$Terms, frequency = tdm.mod$v, stringsAsFactors = FALSE) %>%
  arrange(-frequency)

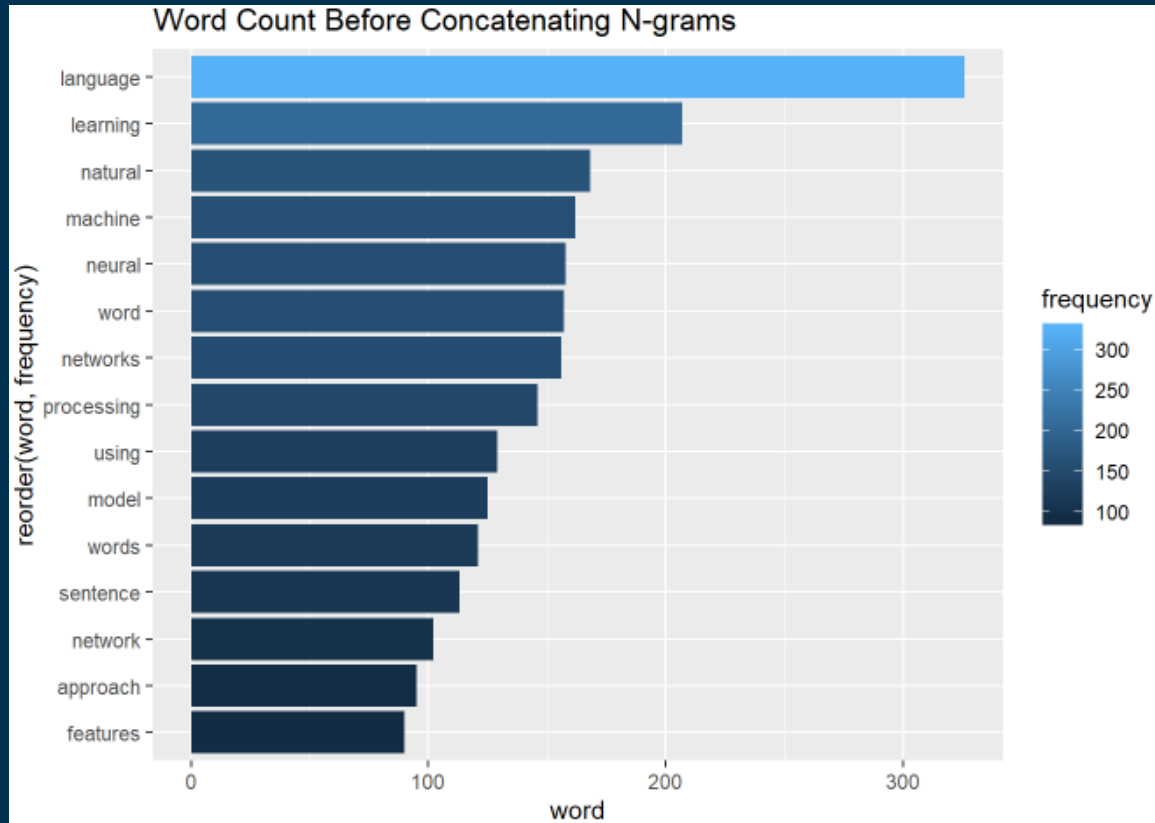
tdm.mod.word
```

word <chr>	frequency <dbl>
word	157
using	129
words	121
neural_networks	118
sentence	113
natural_language_processing	110
network	102
approach	95
machine_learning	94
features	90

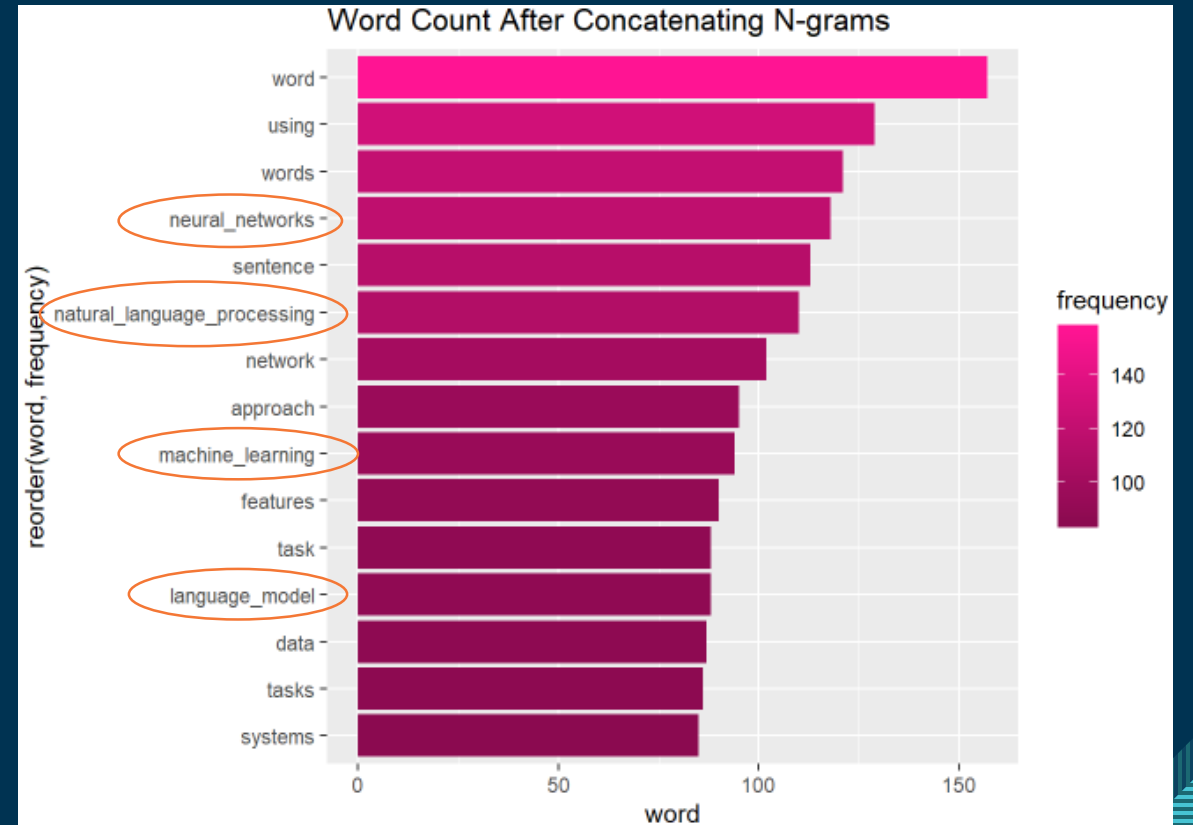
1-10 of 3,665 rows Previous 1 2 3 4 5 6 ... 367 Next

Charts

Before concatenating:



After concatenating:



Limitations

- As the outputs are scraped from paragraph, some words like “word”, “using”, “sentence” got into top 10 phrases. But they in fact are the words appeared with most frequencies.
- Further study is needed to omit terms above.
- If these codes are being applied to the skills listed in Project 3, the problem of having separated words can be solved.

Reference

- Introduction to the tm Package - Text Mining in R
<https://cran.r-project.org/web/packages/tm/vignettes/tm.pdf>
- Package 'tm'
<https://cran.r-project.org/web/packages/tm/tm.pdf>
- RWeka Odds and Ends
<https://cran.r-project.org/web/packages/RWeka/vignettes/RWeka.pdf>
- Package 'RWeka'
<https://cran.r-project.org/web/packages/RWeka/RWeka.pdf>

Thank You



The background features a diagonal split. The upper-left portion is light blue with thin, darker blue horizontal stripes. The lower-right portion is a solid dark blue. The text 'Q & A' is centered in the dark blue area.

Q & A