Understanding an R corpus

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN R

Kasey Jones
Data Scientist

Corpora

- Collections of documents containing natural language text
- From the `tm` package as `corpus`
- `VCorpus` - most common representation

[1] https://www.rdocumentation.org/packages/tm/versions/0.7 [2] 6/topics/Corpus

Contents of a VCorpus: metadata

```
library(tm)
data("acq")
```

```
acq[[1]]$meta
```

```
  author       : character(0)
  datetimestamp: 1987-02-26 15:18:06
  heading      : COMPUTER TERMINAL SYSTEMS <CPML> COMPLETES SALE
  id           : 10
  language     : en
  origin       : Reuters-21578 XML
  ...          : ...
```

[1] http://www.daviddlewis.com/resources/testcollections/reuters21578/

Contents of a VCorpus: metadata

```r
library(tm)
data("acq")
```

```r
acq[[1]]$meta$places
```

```
[1] "usa"
```

Contents of a VCorpus: content

```
acq[[1]]$content
```

```
[1] "Computer Terminal Systems Inc said it has completed ...
```

```
acq[[2]]$content
```

```
[1] "Ohio Mattress Co said its first quarter, ending ...
```

Tidying a corpus

```
library(tm)
library(tidytext)
data("acq")
```

```
tidy_data <- tidy(acq)
tidy_data
```

```
# A tibble: 50 x 16
   author datetimestamp       description heading id    language origin
   <chr>  <dttm>              <chr>       <chr>   <chr> <chr>    <list>
 1 <NA>   1987-02-26 10:18:06 ""          COMPUT… 10    en         <chr …
```

Creating a corpus

Create the corpus

```
corpus <- VCorpus(VectorSource(tidy_data$text))
```

Add the meta information

```
meta(corpus, 'Author') <- tidy_data$author
meta(corpus, 'oldid') <- tidy_data$oldid
head(meta(corpus))
```

```
  Author oldid
1 <NA>   5553
2 <NA>   5555
```

Let's see this in action.

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN R

The bag-of-words representation

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN R

Kasey Jones
Research Data Scientist

The previous example

```
animal_farm %>%
  unnest_tokens(output = "word", token = "words",
                input = text_column) %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE)
```

```
# A tibble: 3,611 x 2
   word          n
   <chr>     <int>
 1 animals     248
 2 farm        163
 ...
```

The bag-of-words representation

```
text1 <- c("Few words are important.")
text2 <- c("All words are important.")
text3 <- c("Most words are important.")
```

Unique Words:

- few: only in `text1`
- all: only in `text2`
- most: only in `text3`
- words, are, important

Typical vector representations

```
# Lowercase, without stop words
word_vector <- c("few", "all", "most", "words", "important")
```

```
# Representation for text1
text1 <- c("Few words are important.")
text1_vector <- c(1, 0, 0, 1, 1)
# Representation for text2
text2 <- c("All words are important.")
text2_vector <- c(0, 1, 0, 1, 1)
# Representation for text3
text3 <- c("Most words are important.")
text3_vector <- c(0, 0, 1, 1, 1)
```

tidytext representation

```
words <- animal_farm %>%
    unnest_tokens(output = "word", token = "words", input = text_column) %>%
    anti_join(stop_words) %>%
    count(chapter, word, sort = TRUE)
words
```

```
# A tibble: 6,807 x 3
   chapter    word          n
   <chr>      <chr>     <int>
 1 Chapter 8  napoleon     43
 2 Chapter 8  animals      41
 3 Chapter 9  boxer        34
...
```

One word example

```
words %>%
  filter(word == 'napoleon') %>%
  arrange(desc(n))
```

```
# A tibble: 9 x 3
  chapter    word          n
  <chr>      <chr>     <int>
1 Chapter 8  napoleon     43
2 Chapter 7  napoleon     24
3 Chapter 5  napoleon     22
...
8 Chapter 3  napoleon      3
9 Chapter 4  napoleon      1
```

Sparse matrices

```r
library(tidytext); library(dplyr)
russian_tweets <- read.csv("russian_1.csv", stringsAsFactors = F)
russian_tweets <- as_tibble(russian_tweets)

tidy_tweets <- russian_tweets %>%
  unnest_tokens(word, content) %>%
  anti_join(stop_words)
tidy_tweets %>%
  count(word, sort = TRUE)
```

```
# A tibble: 43,666 x 2
...
```

Sparse matrices continued

Sparse Matrix

- 20,000 rows (the tweets)
- 43,000 columns (the words)
- 20,000 * 43,000 = 860,000,000
- Only 177,000 non-0 entries. About .02%

Sparse matrix example:

1	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0
0	1	0	0	0	3	0	0
1	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1
0	0	1	0	0	0	4	0

BoW Practice

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN R

The TFIDF

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN R

Kasey Jones
Research Data Scientist

Bag-of-word pitfalls

```
t1 <- "My name is John. My best friend is Joe. We like tacos."
t2 <- "Two common best friend names are John and Joe."
t3 <- "Tacos are my favorite food. I eat them with my friend Joe."
```

```
clean_t1 <- "john friend joe tacos"
clean_t2 <- "common friend john joe names"
clean_t3 <- "tacos favorite food eat buddy joe"
```

Sharing common words

```
clean_t1 <- "john friend joe tacos"
clean_t2 <- "common friend john joe names"
clean_t3 <- "tacos favorite food eat buddy joe"
```

Compare t1 and t2

- 3/4 words from t1 are in t2
- 3/5 words from t2 are in t1

Compare t1 and t3

- 2/4 words from t1 are in t3
- 2/6 words from t3 are in t1

Tacos matter

```
t1 <- "My name is John. My best friend is Joe. We like tacos."
t2 <- "Two common best friend names are John and Joe."
t3 <- "Tacos are my favorite food. I eat them with my friend Joe."
```

Words in each text:

- John: t1, t2, t3
- Joe: t1, t2, t3
- Tacos: t1, t3

TFIDF

```
clean_t1 <- "john friend joe tacos"
clean_t2 <- "common friend john joe names"
clean_t3 <- "tacos favorite food eat buddy joe"
```

- TF: Term Frequency
  - The proportion of words in a text that are that term
  - john is 1/4 words in `clean_t1` , tf = .25
- IDF: Inverse Document Frequency
  - The weight of how common a term is across all documents
  - john is in 3/3 documents, IDF = 0

IDF Equation

$$IDF = log\frac{N}{n_t}$$

- N: total number of documents in the corpus
- n_t: number of documents where the term appears

Example:

- Taco IDF: $log(\frac{3}{2}) = .405$
- Buddy IDF: $log(\frac{3}{1}) = 1.10$
- John IDF: $log(\frac{3}{3}) = 0$

TF + IDF

```
clean_t1 <- "john friend joe tacos"
clean_t2 <- "common friend john joe names"
clean_t3 <- "tacos favorite food eat buddy joe"
```

TFIDF for "tacos":

- clean_t1: TF * IDF = (1/4) * (.405) = 0.101
- clean_t2: TF * IDF = (0/4) * (.405) = 0
- clean_t3: TF * IDF = (1/6) * (.405) = 0.068

Calculating the TFIDF matrix

```
# Create a data.frame
df <- data.frame('text' = c(t1, t2, t3), 'ID' = c(1, 2, 3), stringsAsFactors = F)
```

```
df %>%
  unnest_tokens(output = "word", token = "words", input = text) %>%
  anti_join(stop_words) %>%
  count(ID, word, sort = TRUE) %>%
  bind_tf_idf(word, ID, n)
```

- word: the column containing the terms
- ID: the column containing document IDs
- n: the word count produced by `count()`

bind_tf_idf output

```
# A tibble: 15 x 6
        X word            n    tf   idf tf_idf
    <dbl> <chr>       <int> <dbl> <dbl>  <dbl>
 1      1 friend          1 0.25  0.405 0.101
 2      1 joe             1 0.25  0     0
 3      1 john            1 0.25  0.405 0.101
 4      1 tacos           1 0.25  0.405 0.101
 5      2 common          1 0.2   1.10  0.220
 6      2 friend          1 0.2   0.405 0.0811
 ...
```

TFIDF Practice

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN R

Cosine Similarity

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN R

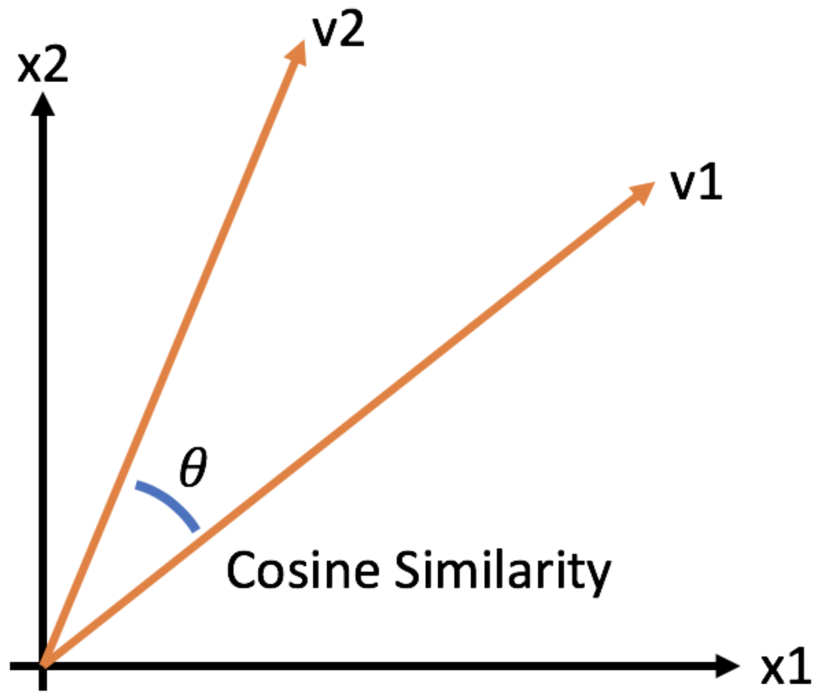
Kasey Jones
Research Data Scientist

TFIDF output

```
# A tibble: 1,498 x 6
        X word            n      tf    idf tf_idf
    <int> <chr>       <int>   <dbl> <dbl>   <dbl>
 1     20 january         4 0.0930  2.30   0.214
 2     15 power           4 0.0690  3.00   0.207
 3     19 futures         9 0.0643  3.00   0.193
 4      8 8               6 0.0619  3.00   0.185
 5      3 canada          2 0.0526  3.00   0.158
 6      3 canadian        2 0.0526  3.00   0.158
```

Cosine similarity

- a measure of similarity between two vectors
- measured by the angle formed by the two vectors



[1] https://en.wikipedia.org/wiki/Cosine_similarity

Cosine similarity formula

- similarity is calculated as the two vectors dot product

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Finding similarities part I

```
crude_weights <- crude_tibble %>%
  unnest_tokens(output = "word", token = "words", input = text) %>%
  anti_join(stop_words) %>%
  count(X, word) %>%
  bind_tf_idf(X, word, n)
```

```
# A tibble: 1,498 x 6
      X word          n    tf   idf tf_idf
  <int> <chr>     <int> <dbl> <dbl>  <dbl>
1     1 1.50          1 0.25   3.25  0.812
2     1 16.00         1 1      3.25  3.25
3     1 barrel        2 0.133  3.25  0.433
...
```

Pairwise similarity

```
pairwise_similarity(tbl, item, feature, value, ...)
```

- tbl: a table or tibble
- item: the items to compare (articles, tweets, etc.)
- feature: column describing the link between the items (i.e. words)
- value: the column of values (i.e. n or tf_idf)

Finding similarities part II

```
crude_weights %>%
  pairwise_similarity(X, word, tf_idf) %>%
  arrange(desc(similarity))
```

```
# A tibble: 380 x 3
   item1 item2 similarity
   <int> <int>      <dbl>
 1    17    16      0.663
 2    16    17      0.663
 3    13    10      0.311
 4    10    13      0.311
 ...
```

Cosine similarity use-cases

- find duplicate/similar pieces of text
- use in clustering and classification analysis
- ...

Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN R