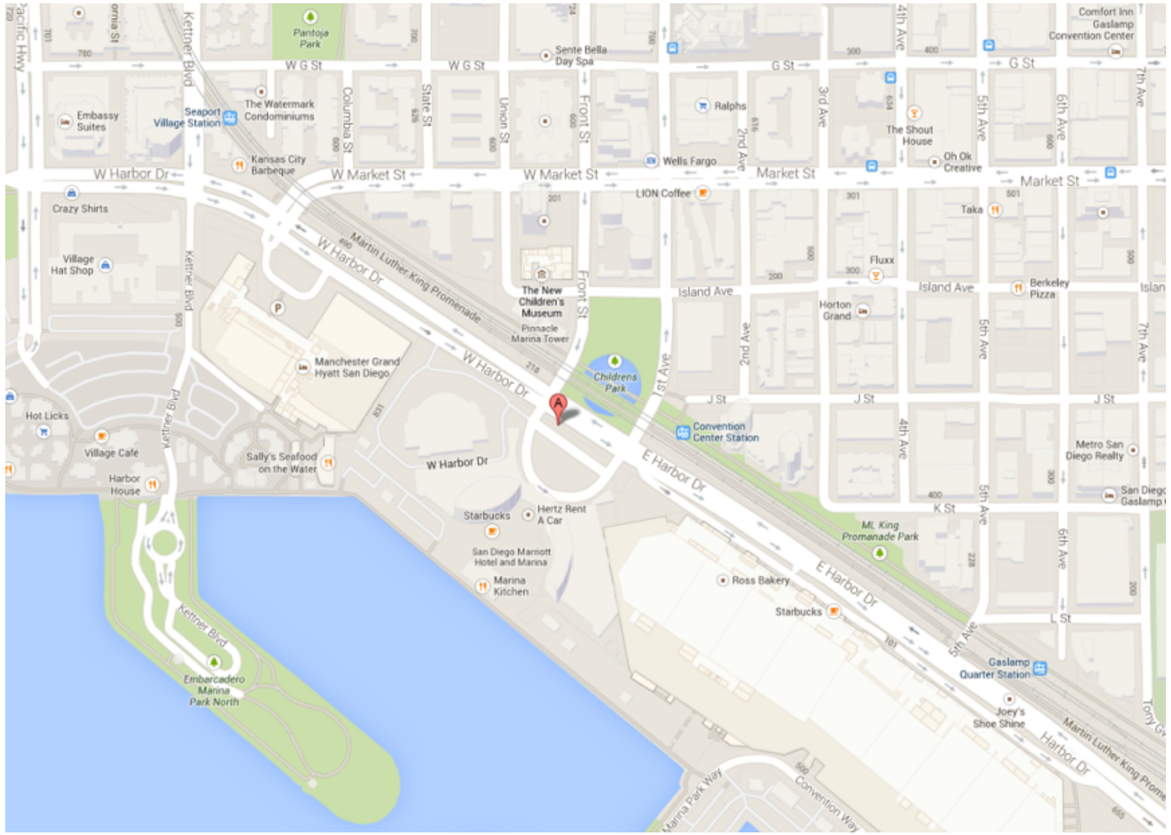
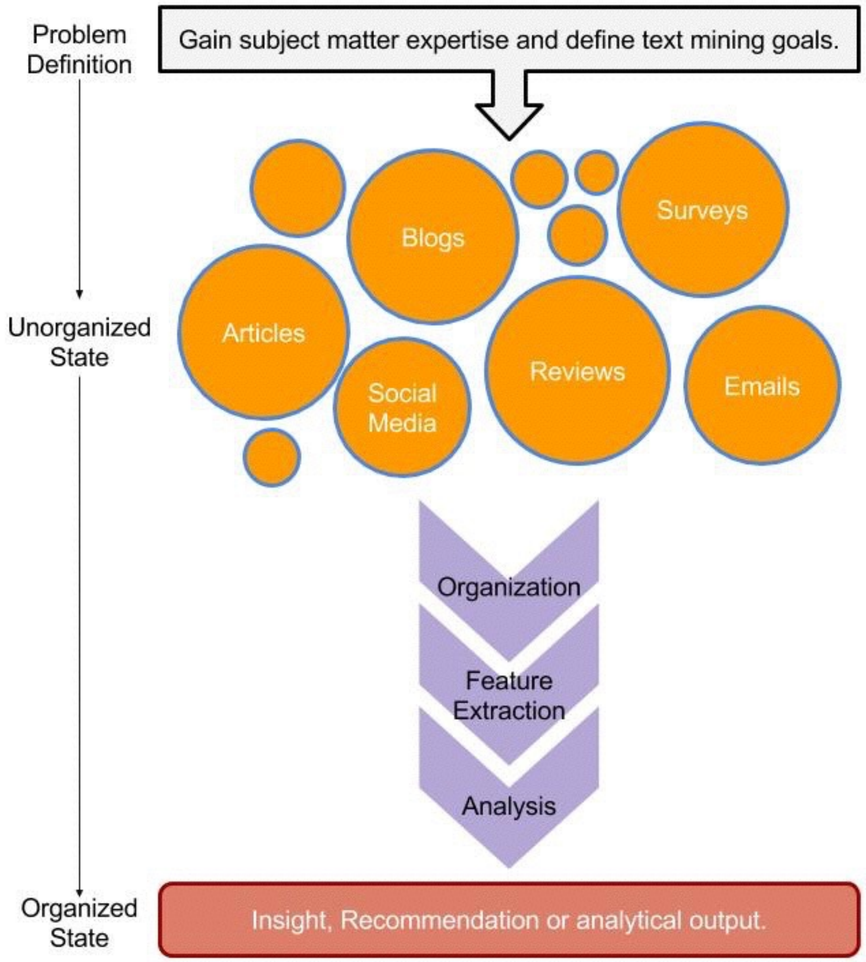
What is text mining?

What is text mining?

The process of distilling actionable insights from text



Text mining workflow



1 - Problem definition & specific goals

2 - Identify text to be collected

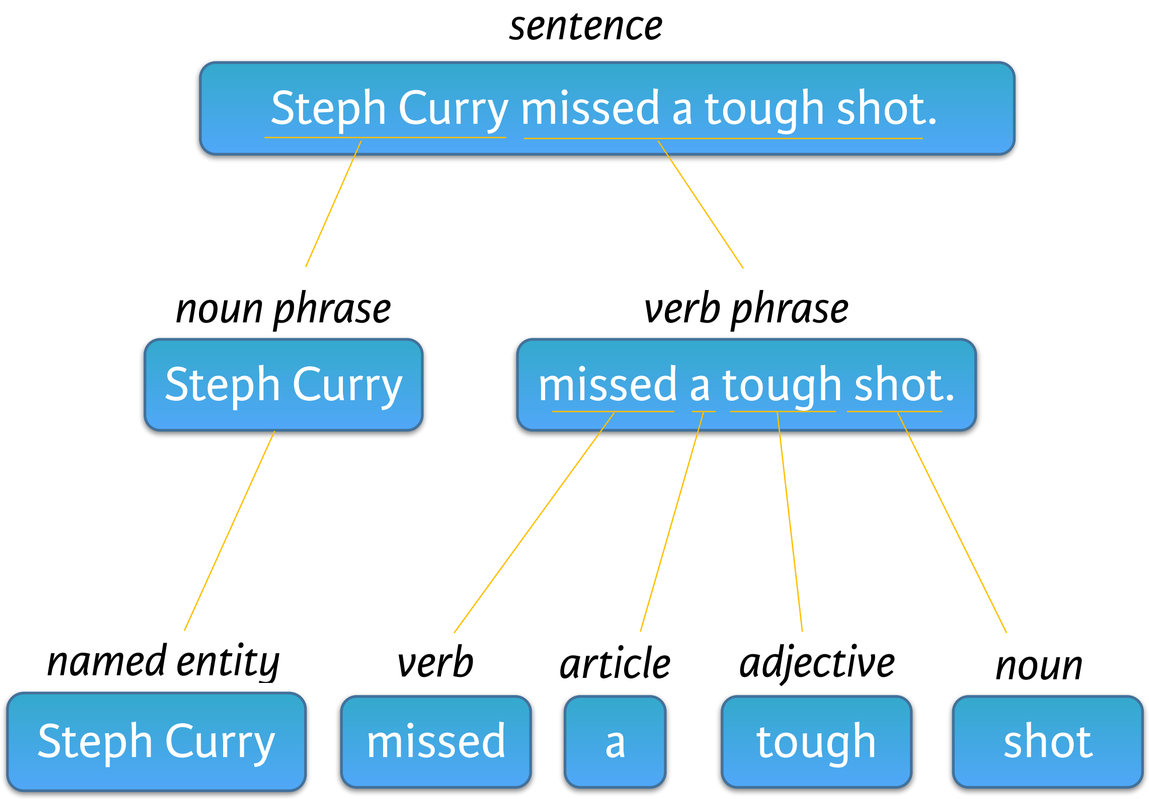
3 - Text organization

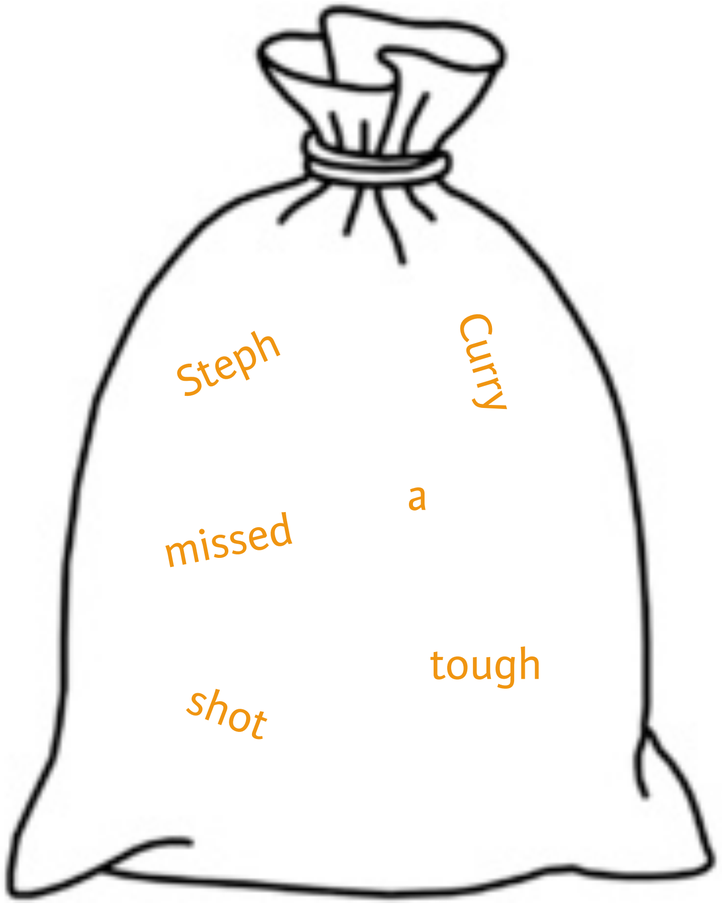
4 - Feature extraction

5 - Analysis

6 - Reach an insight, recommendation or output

Semantic parsing vs. bag of words





INTRO TO TEXT MINING: BAG OF WORDS

Let’s practice!

INTRO TO TEXT MINING: BAG OF WORDS

Getting started

Building our first corpus

```
> # Load corpus
> coffee_tweets <- read.csv("coffee.csv", stringsAsFactors = FALSE)

> # Vector of tweets
> coffee_tweets <- coffee_tweets$text

> # View first 5 tweets
> head(coffee_tweets, 5)
[1] "@ayyytylerb that is so true drink lots of coffee"
[2] "RT @bryzy_brib: Senior March tmw morning at 7:25 A.M. in the
SENIOR lot. Get up early, make yo coffee/breakfast, cus this will
only happen ?"
[3] "If you believe in #gunsense tomorrow would be a very good day
to have your coffee any place BUT @Starbucks Guns+Coffee=#nosense
@MomsDemand"
[4] "My cute coffee mug. http://t.co/2udvMU6XIG"
[5] "RT @slaredo21: I wish we had Starbucks here... Cause coffee
dates in the morning sound perff!"
```

INTRO TO TEXT MINING: BAG OF WORDS

Let’s practice!

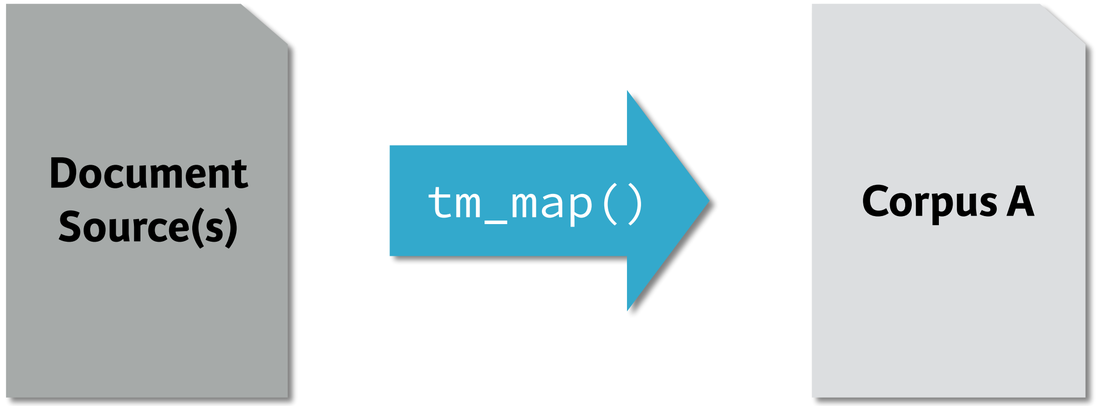
INTRO TO TEXT MINING: BAG OF WORDS

Cleaning and preprocessing text

Common preprocessing functions

TM Function	Description	Before	After
`tolower()`	Makes all text lowercase	`Starbucks is from Seattle.`	`starbucks is from seattle.`
`removePunctuation()`	Removes punctuation like periods and exclamation points	`Watch out!  That coffee is going to spill!`	`Watch out  That coffee is going to spill`
`removeNumbers()`	Removes numbers	`I drank 4 cups of coffee 2 days`	`I drank  cups of coffee  days ago.`
`stripWhiteSpace()`	Removes tabs and extra spaces	`I like    coffee.`	`I like coffee.`
`removeWords()`	Removes specific words (e.g. "the", "of") defined by the data scientist	`The coffee house and barista he visited were nice, she said hello.`	`The coffee house barista  visited nice,  said hello.`

Preprocessing in practice




```
> # Make a vector source: coffee_source
> coffee_source <- VectorSource(coffee_tweets)

> # Make a volatile corpus: coffee_corpus
> coffee_corpus <- VCorpus(coffee_source)

> # Apply various preprocessing functions
> tm_map(coffee_corpus, removeNumbers)
> tm_map(coffee_corpus, removePunctuation)
> tm_map(coffee_corpus, content_transformer(replace_abbreviation))
```

Another preprocessing step: word stemming

```
> # Stem words
> stem_words <- stemDocument(c("complicatedly", "complicated",
                               "complication"))
> stem_words
[1] “complic” “complic” “complic”

> # Complete words using single word dictionary
> stemCompletion(stem_words, c("complicate"))
     complic      complic      complic
"complicate" "complicate" "complicate"

> # Complete words using entire corpus
> stemCompletion(stem_words, my_corpus)
```

INTRO TO TEXT MINING: BAG OF WORDS

Let’s practice!

INTRO TO TEXT MINING: BAG OF WORDS

The TDM & DTM

TDM vs. DTM

	Tweet 1	Tweet 2	Tweet 3	...	Tweet N
Term 1	0	0	0	0	0
Term 2	1	1	0	0	0
Term 3	1	0	0	0	0
...	0	0	3	1	1
Term M	0	0	0	1	0

Term Document Matrix (TDM)

	Term 1	Term 2	Term 3	...	Term M
Tweet 1	0	1	1	0	0
Tweet 2	0	1	0	0	0
Tweet 3	0	0	0	3	0
...	0	0	0	1	1
Tweet N	0	0	0	1	0

Document Term Matrix (DTM)

```
> # Generate TDM
> coffee_tdm <- TermDocumentMatrix(clean_corp)

> # Generate DTM
> coffee_dtm <- DocumentTermMatrix(clean_corp)
```

Word Frequency Matrix (WFM)

```
> # Load qdap package
> library(qdap)

> # Generate word frequency matrix
> coffee_wfm <- wfm(coffee_text$text)
```

	Tweet 1
Term 1	0
Term 2	1
Term 3	1
...	0
Term M	0

Word Frequency Matrix (WFM)

INTRO TO TEXT MINING: BAG OF WORDS

Let’s practice!

INTRO TO TEXT MINING: BAG OF WORDS

Common text mining visuals

Why make visuals?

- Good visuals lead to quick conclusions
- The brain efficiently processes visual information

Setting the scene

Term Document Matrix (TDM)

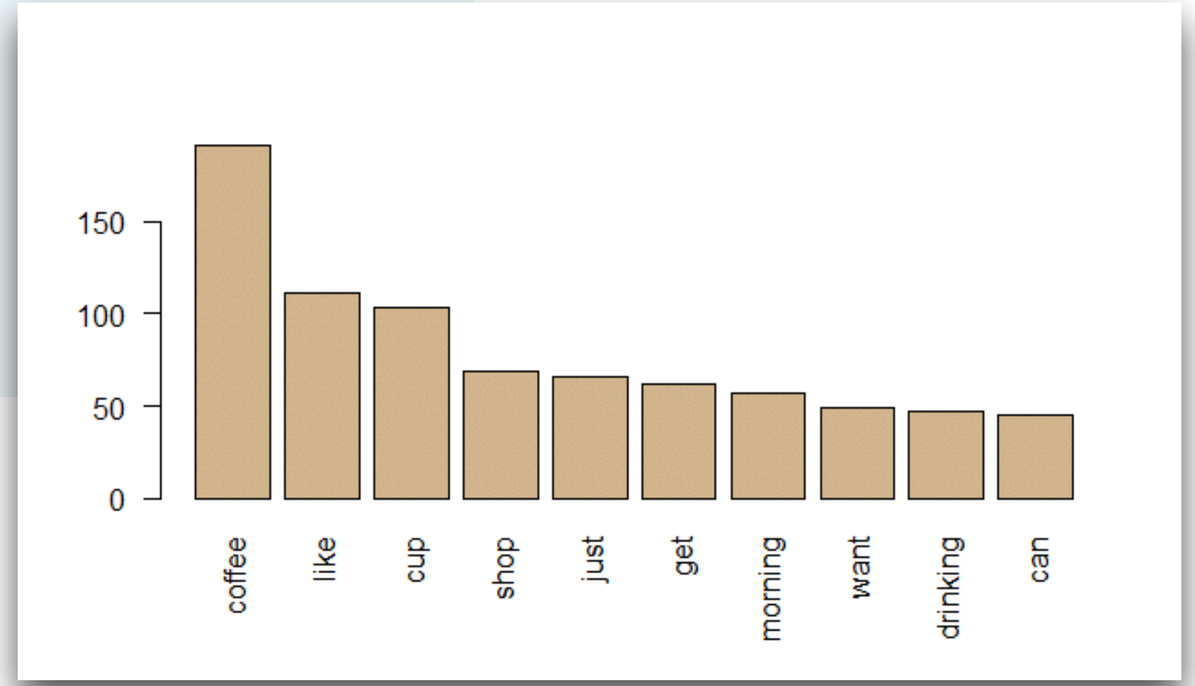
	Tweet1	Tweet2	Tweet3	...	Tweet_N
Term1	0	0	0	0	0
Term2	1	1	0	0	0
Term3	1	0	0	2	0
...	0	0	3	1	1
Term_N	0	0	1	1	0

Summed vector

Sum
0
2
3
5
2

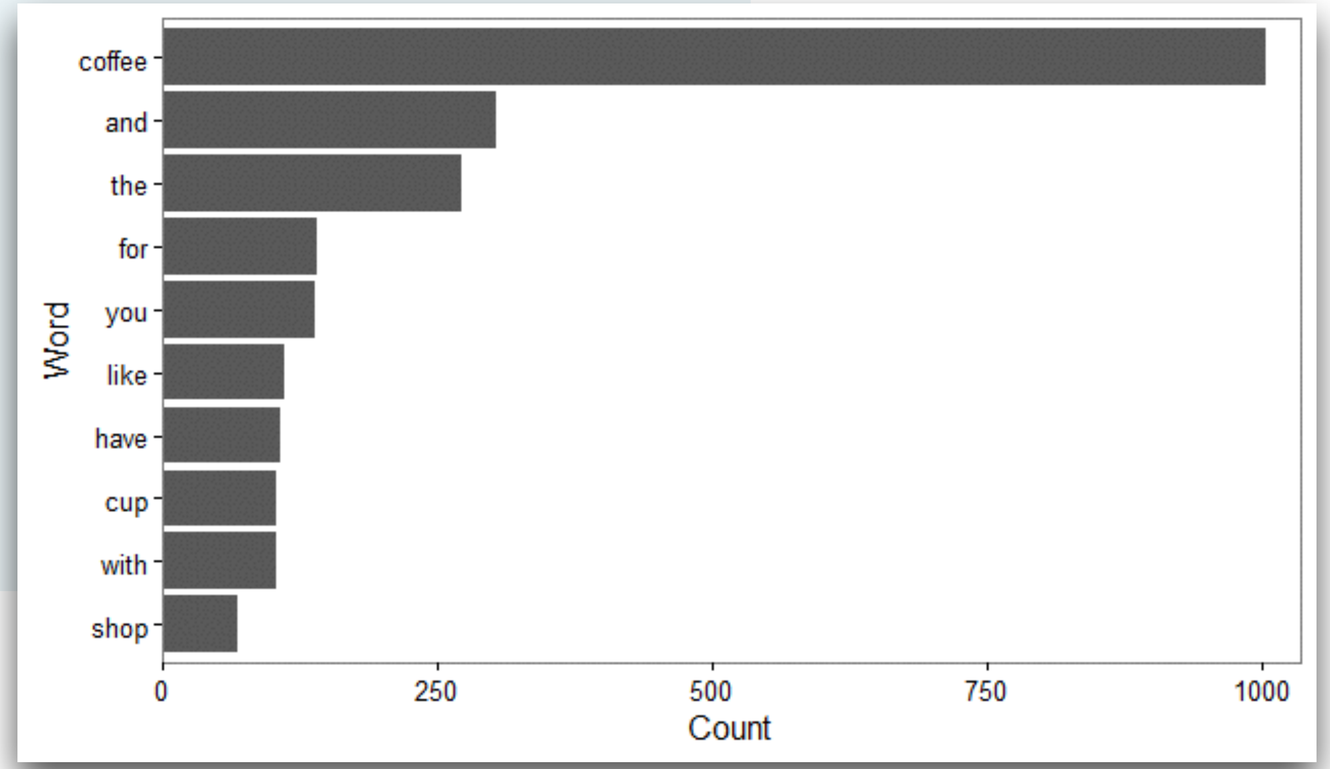
Term frequency plots with tm

```
> # Convert TDM to matrix
> coffee_m <- as.matrix(coffee_tdm)

> # Sum rows and sort by frequency
> term_frequency <- rowSums(coffee_m)
> term_frequency <- sort(term_frequency,
                         decreasing = TRUE)

> # Create a barplot
> barplot(term_frequency[1:10],
          col = "tan", las = 2)
```



Term frequency plots with qdap

```
> # Load qdap package
> library(qdap)

> # Find term frequencies
> frequency <- freq_terms(
    tweets$text,
    top = 10,
    at.least = 3,
    stopwords = "Top200Words"
  )

> # Plot term frequencies
> plot(frequency)
```



INTRO TO TEXT MINING: BAG OF WORDS

Let’s practice!

INTRO TO TEXT MINING: BAG OF WORDS

Intro to word clouds

A simple word cloud

```
> # Convert TDM to matrix
> chardonnay_tdm <- TermDocumentMatrix(clean_chardonnay)
> chardonnay_m <- as.matrix(chardonnay_tdm)

> # Sum rows and sort by frequency
> term_frequency <- rowSums(chardonnay_m)
> word_freqs <- data.frame(term = names(term_frequency),
                           num = term_frequency)

> # Make word cloud
> wordcloud(word_freqs$term, word_freqs$num,
            max.words = 100, colors = "red")
```

The impact of stop words

```
clean_corpus <- function(corpus){
  corpus <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, stripWhitespace)
  corpus <- tm_map(corpus, removeNumbers)
  corpus <- tm_map(corpus, content_transformer(tolower))
  corpus <- tm_map(corpus, removeWords,
                   c(stopwords("en"), "amp"))
  return(corpus)
}
```

Removing uninformative words

```
clean_corpus <- function(corpus){
  corpus <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, stripWhitespace)
  corpus <- tm_map(corpus, removeNumbers)
  corpus <- tm_map(corpus, content_transformer(tolower))
  corpus <- tm_map(corpus, removeWords,
                   c(stopwords("en"), "amp",
                     "chardonnay", "wine", "glass"))
  return(corpus)
}
```

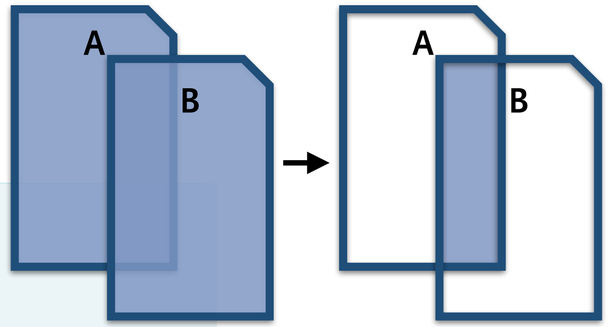
INTRO TO TEXT MINING: BAG OF WORDS

Let’s practice!

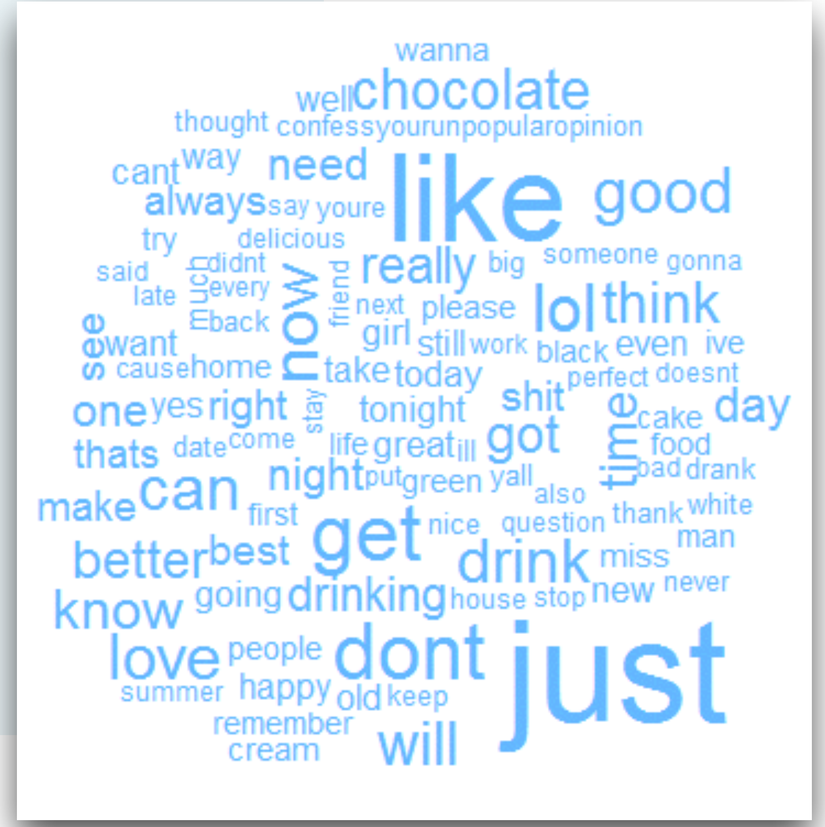
INTRO TO TEXT MINING: BAG OF WORDS

Other word clouds and word networks

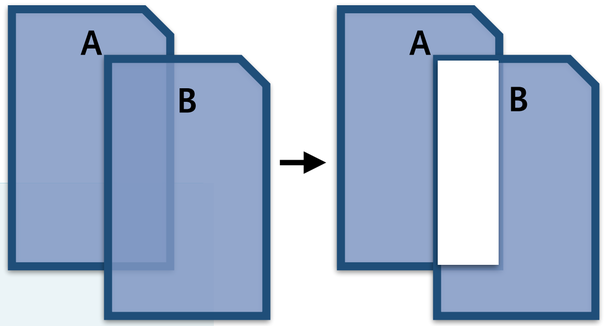
Commonality clouds



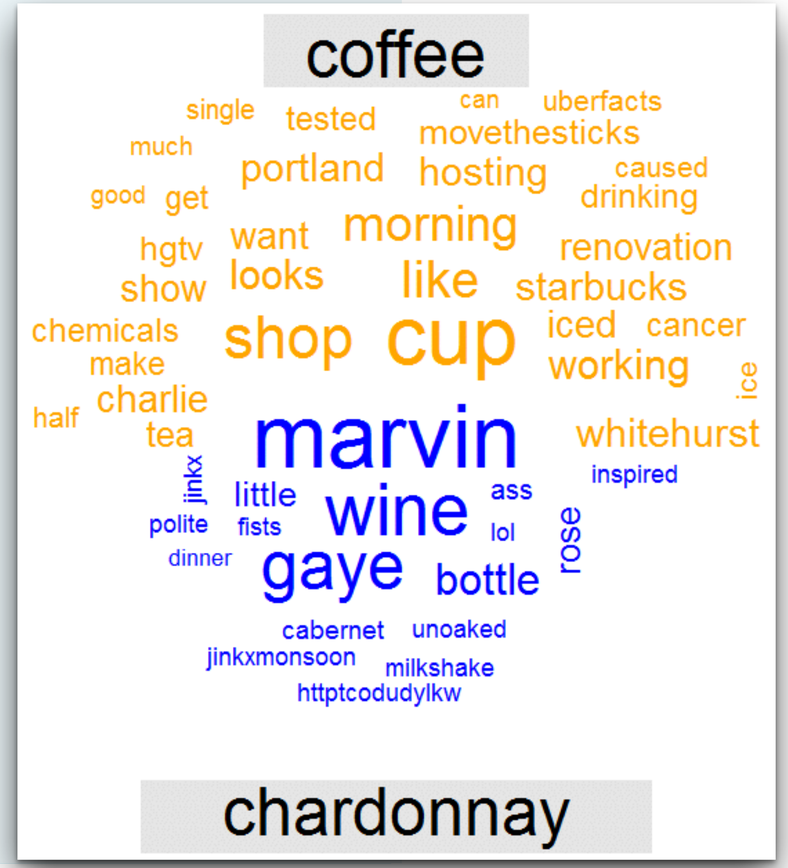

```
> # Combine both corpora: all_tweets
> all_coffee <- paste(coffee_tweets$text, collapse = "")
> all_chardonay <- paste(chardonnay_tweets$text,
                         collapse = "")
> all_tweets <- c(all_coffee, all_chardonnay)

> # Clean all_tweets
> all_tweets <- VectorSource(all_tweets)
> all_corpus <- VCorpus(all_tweets)
> all_clean <- clean_corpus(all_corpus)
> all_dm <- TermDocumentMatrix(all_clean)
> all_m <- as.matrix(all_tdm)

> # Make commonality cloud
> commonality.cloud(all_m, colors = "steelblue1",
                    max.words = 100)
```



Comparison clouds




```
> # Combine both corpora: all_tweets
> all_coffee <- paste(coffee_tweets$text, collapse = "")
> all_chardonay <- paste(chardonnay_tweets$text,
                         collapse = "")
> all_tweets <- c(all_coffee, all_chardonnay)

> # Clean all_tweets
> all_tweets <- VectorSource(all_tweets)
> all_corpus <- VCorpus(all_tweets)
> all_clean <- clean_corpus(all_corpus)
> all_tdm <- TermDocumentMatrix(all_clean)
> colnames(all_tdm) <- c("coffee", "chardonnay")
> all_m <- as.matrix(all_tdm)

> # Make comparison cloud
> comparison.cloud(all_m,
                   colors = c("orange", "blue"),
                   max.words = 50)
```

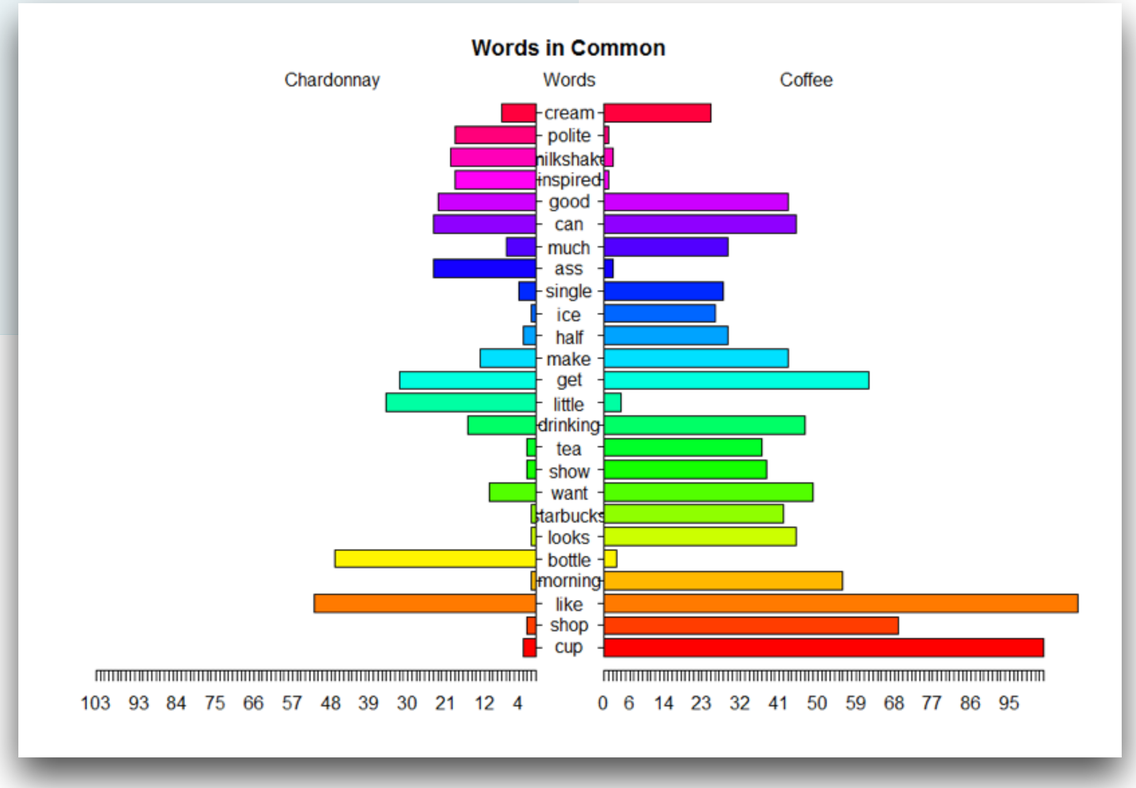


Pyramid plots

```
> # Identify terms shared by both documents
> common_words <- subset(
    all_tdm_m,
    all_tdm_m[, 1] > 0 & all_tdm_m[, 2] > 0
  )

> # Find most commonly shared words
> difference <- abs(common_words[, 1] - common_words[, 2])
> common_words <- cbind(common_words, difference)
> common_words <- common_words[order(common_words[, 3],
                                     decreasing = TRUE), ]
> top25_df <- data.frame(x = common_words[1:25, 1],
                         y = common_words[1:25, 2],
                         labels = rownames(common_words[1:25, ]))
```

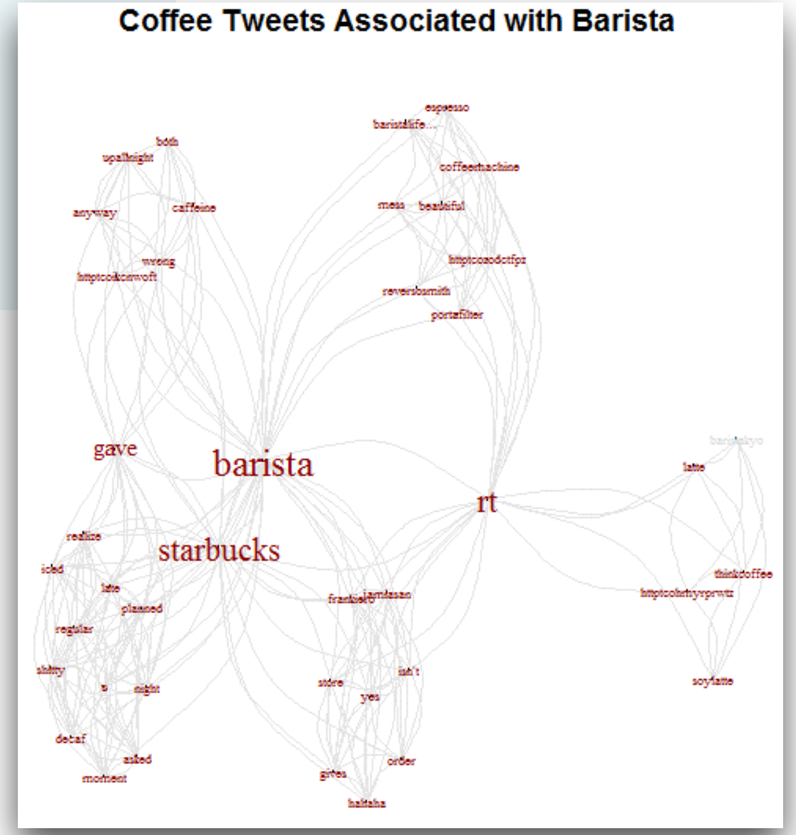
Pyramid plots

```
> # Make pyramid plot
> pyramid.plot(top25_df$x, top25_df$y,
               labels = top25_df$labels,
               main = "Words in Common",
               gap = 8, laxly = NULL,
               raxlab = NULL, unit = NULL,
               top.labels = c("Chardonnay",
                              "Words",
                              "Coffee"))
```



Word networks

```
# Create word network
> word_associate(coffee_tweets$text, match.string = c("barista"),
                 stopwords = c(Top200Words, "coffee", "amp"),
                 network.plot = TRUE,
                 cloud.colors = c("gray85", "darkred"))

# Add title
title(main = "Barista Coffee Tweet Associations")
```



INTRO TO TEXT MINING: BAG OF WORDS

Let’s practice!

INTRO TO TEXT MINING: BAG OF WORDS

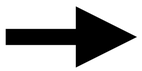
Simple word clustering

Hierarchical clustering example

```
> dist_rain <- dist(rain[, 2])
```

The data

City	Annual rainfall (in.)
Cleveland	39.14
Portland	39.14
Boston	43.77
New Orleans	62.45

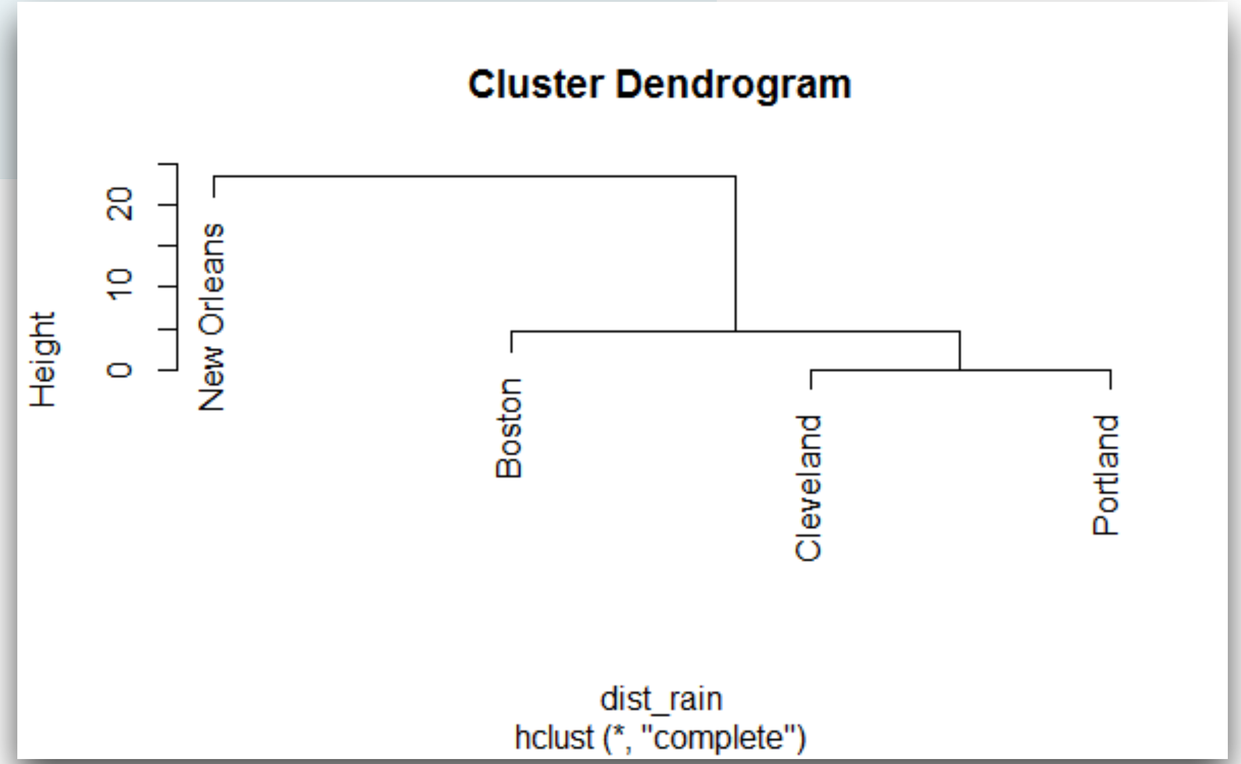


Distance matrix

	Cleveland	Portland	Boston
Portland	0.00		
Boston	4.63	4.63	
New Orleans	23.31	23.31	18.69

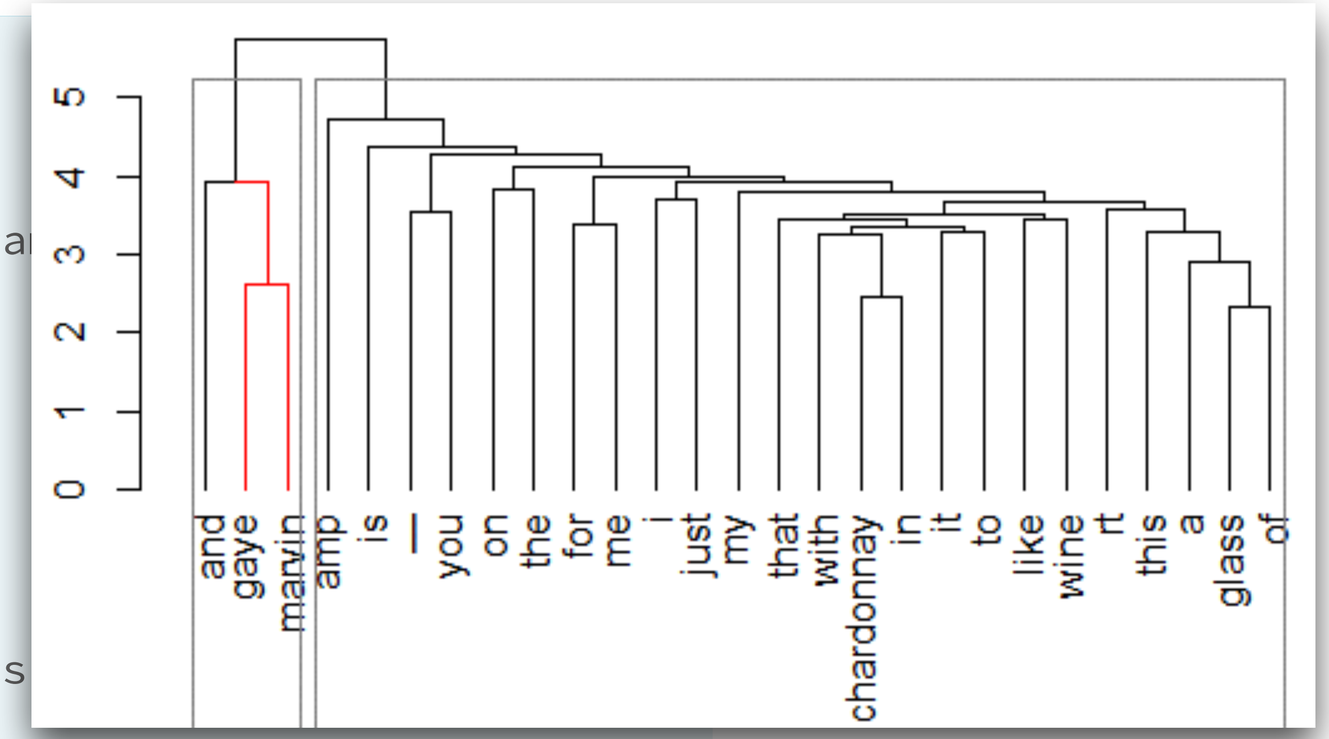
A simple dendrogram

```
> # Reclassify distances as hierarchical cluster object
> hc <- hclust(dist_rain)

> # Plot dendrogram with city labels
> plot(hc, labels = rain$city)
```



Dendrogram aesthetics

```
> # Load dendextend package
> library(dendextend)

> # Convert distance matrix to dendrogra
> hc <- hclust(tweets_dist)
> hcd <- as.dendrogram(hc)

> # Color branches
> hcd <- branches_attr_by_labels(hcd,
            c("marvin", "gaye"), "red")

> # Plot dendrogram with some aesthetics
> plot(hcd, main = "Better Dendrogram")
> rect.dendrogram(hcd, k = 2, border = "grey50")
```



INTRO TO TEXT MINING: BAG OF WORDS

Let’s practice!

INTRO TO TEXT MINING: BAG OF WORDS

Getting past single words

Unigrams, bigrams, trigrams, oh my!

```
> # Use only first 2 coffee tweets
> tweets$text[1:2]
[1] @ayyytylerb that is so true drink lots of coffee
[2] RT @bryzy_brib: Senior March tmw morning at 7:25 A.M. in the
SENIOR lot. Get up early, make yo coffee/breakfast, cus this will
only happen…

> # Make a unigram DTM on first 2 coffee tweets
> unigram_dtm <- DocumentTermMatrix(text_corp)
> unigram_dtm
<<DocumentTermMatrix (documents: 2, terms: 18)>>
Non-/sparse entries: 18/18
Sparsity           : 50%
Maximal term length: 15
Weighting          : term frequency (tf)
```

Unigrams, bigrams, trigrams, oh my!

```
> # Load RWeka package
> library(RWeka)

> # Define bigram tokenizer
> tokenizer <- function(x)
    NGramTokenizer(x, Weka_control(min = 2, max = 2))

> # Make a bigram TDM
> bigram_tdm <- TermDocumentMatrix(
    clean_corpus(text_corp),
    control = list(tokenize = tokenizer)
  )
> bigram_tdm
<<DocumentTermMatrix (documents: 2, terms: 21)>>
Non-/sparse entries: 21/21
Sparsity           : 50%
Maximal term length: 19
Weighting          : term frequency (tf)
```

INTRO TO TEXT MINING: BAG OF WORDS

Let’s practice!

INTRO TO TEXT MINING: BAG OF WORDS

Different frequency criteria

Term weights

- Default term frequency = simple word count
- Frequent words can mask insights
- Adjust term weighting via TfIdf

Term frequency-inverse document frequency

- Words appearing in many documents are penalized

Term weights

```
> # Standard term weighting
> tf_tdm <- TermDocumentMatrix(text_corp)
> tf_tdm_m <- as.matrix(tf_dtm)
> tf_tdm_m[505:510, 5:10]

> # TfIdf weighting
> tf_idf_tdm <- TermDocumentMatrix(text_corp,
                    control = list(weighting = weightTfIdf))
> tf_idf_tdm_m <- as.matrix(tf_idf_dtm)
> tf_tdm_m <- as.matrix(tf_dtm)
```

	Docs					
Terms	5	6	7	8	9	10
cocoa	0	0	0	0	0	0
cocobear	0	0	0	0	0	0
coconut	0	0	0	0	0	0
codagogy	0	0	0	0	0	0
code-alan	0	0	0	0	0	0
coffee	1	1	1	1	1	1

	Docs					
Terms	5	6	7	8	9	10
cocoa	0.00	0.000	0.000	0.000	0.000	0.000
cocobear	0.00	0.000	0.000	0.000	0.000	0.000
coconut	0.00	0.000	0.000	0.000	0.000	0.000
codagogy	0.00	0.000	0.000	0.000	0.000	0.000
code-alan	0.00	0.000	0.000	0.000	0.000	0.000
coffee	0.01	0.014	0.008	0.043	0.022	0.029

Retaining document metadata

```
> # Create mapping to metadata
> custom_reader <- readTabular(mapping = list(content = "text",
                               id = "num", author = "screenName",
                               date = "created"))

> # Create VCorpus including metadata
> test_corpus <- VCorpus(DataframeSource(tweets),
                   readerControl = list(reader = custom_reader))

> # Clean and view results
> text_corpus <- clean_corpus(text_corpus)
> text_corpus[[1]][1]
$content
[1] "ayyytylerb true drink lots coffee"
> text_corpus[[1]][2]
$meta
  id       : 1
  author   : thejennagibson
  date     : 8/9/2013 2:43
  language: en
```

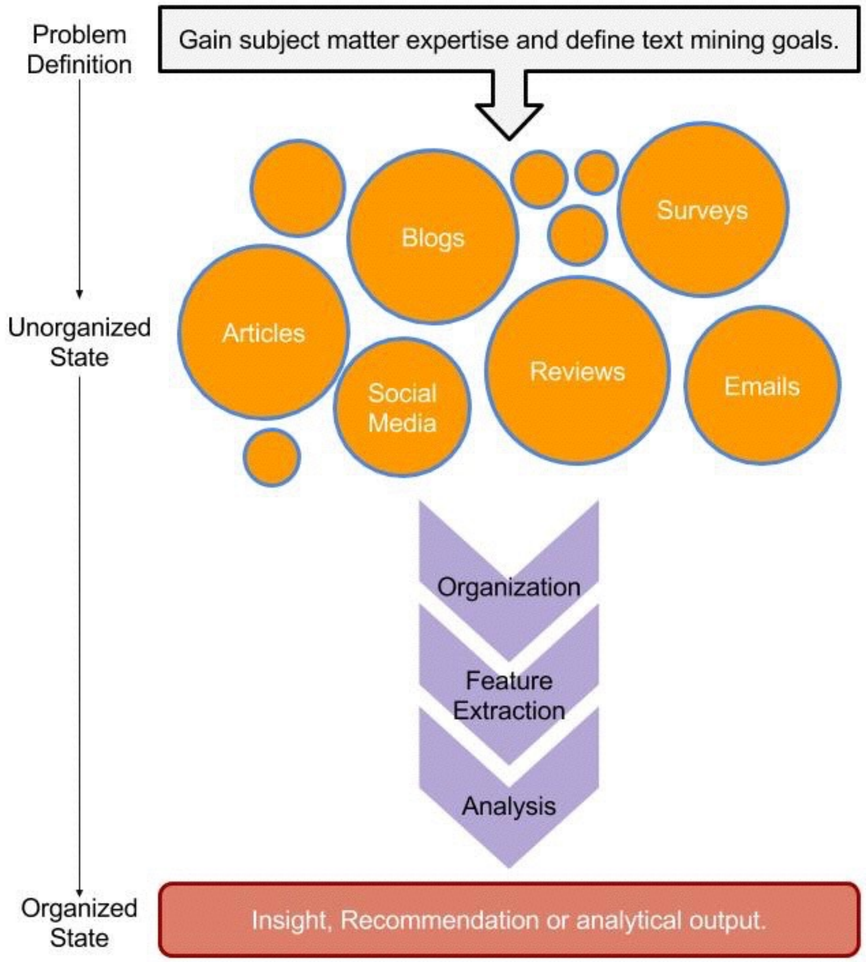
INTRO TO TEXT MINING: BAG OF WORDS

Let’s practice!

INTRO TO TEXT MINING: BAG OF WORDS

Amazon vs. Google

Remember the workflow?



1 - Problem definition & specific goals

2 - Identify text to be collected

3 - Text organization

4 - Feature extraction

5 - Analysis

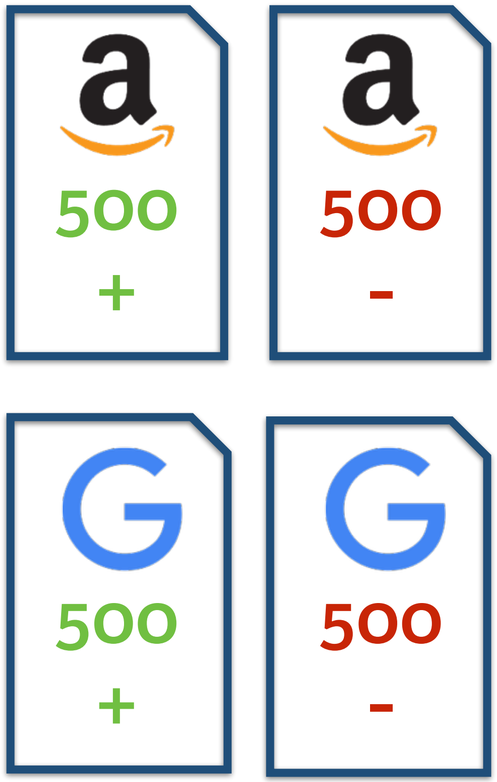
6 - Reach an insight, recommendation or output

A case study in HR analytics

1. **Problem definition**

Which company has better work life balance? Which has better perceived pay according to online reviews?

2. **Unorganized state**



3. Organization
4. Feature Extraction
5. Analysis

6. **Organized state**

Insight, recommendation, analytical output

INTRO TO TEXT MINING: BAG OF WORDS

Let’s practice!

INTRO TO TEXT MINING: BAG OF WORDS

Text organization

Text organization with qdap

```
# qdap cleaning function
qdap_clean <- function(x) {
  x <- replace_abbreviation(x)
  x <- replace_contraction(x)
  x <- replace_number(x)
  x <- replace_ordinal(x)
  x <- replace_symbol(x)
  x <- tolower(x)
  return(x)
}
```

Text organization with tm

```
# tm cleaning function
tm_clean <- function(corpus) {
  tm_clean <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, stripWhitespace)
  corpus <- tm_map(corpus, removeWords,
                   c(stopwords("en"), "Google", "Amazon", "company"))
  return(corpus)
}
```

Cleaning your corpora

INTRO TO TEXT MINING: BAG OF WORDS

Let’s practice!

INTRO TO TEXT MINING: BAG OF WORDS

Feature extraction and analysis

Feature extraction

```
> # Create bigram TDM
> amzn_p_tdm <- TermDocumentMatrix(
    amzn_pros_corp,
    control = list(tokenize = tokenizer)
  )
```

	Review 1	Review 2	...	Review N
Bigram 1	0	0	0	0
Bigram 2	1	1	0	0
Bigram 3	1	0	0	0
...	0	0	1	1
Bigram M	0	0	1	0

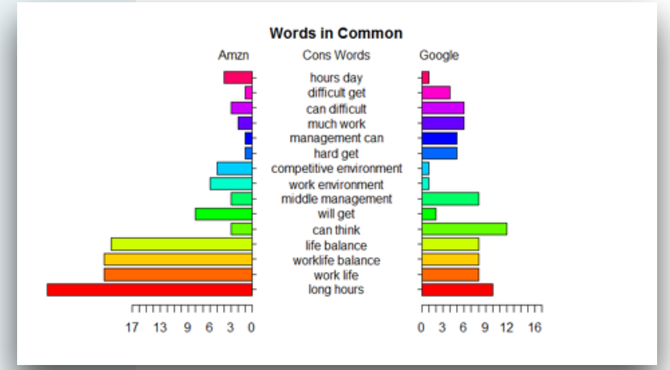
Term Document Matrix (TDM)

Get term frequencies

```
> # Convert TDM to matrix
> amzn_p_m <- as.matrix(amzn_p_tdm)

> # Compute term frequencies
> amzn_p_freq <- rowSums(amzn_p_m)

> # Sort in decreasing order of frequency
> term_frequency <- sort(amzn_p_freq, decreasing = TRUE)

> View the top 5 most frequent bigrams
> term_frequency[1:5]
      good pay great benefits   smart people
            25              24              20
   place work     fast paced
            17              16
```

Create visuals with plotrix

```
> # Find common words
> common_words <- subset(all_tdm_m,
                         all_tdm_m[, 1] > 0 & all_tdm_m[, 2] > 0)
> difference <- abs(common_words[, 1] - common_words[, 2])
> common_words <- cbind(common_words, difference)
> common_words <- common_words[order(common_words[, 3],
                                     decreasing = TRUE), ]

> # Create data frame: top 15 words
> top15_df <- data.frame(x = common_words[1:15, 1],
                         y = common_words[1:15, 2],
                         labels = rownames(common_words[1:15, ]))

> # Make pyramid plot
> pyramid.plot(top15_df$x, top15_df$y, labels = top15_df$labels,
               gap = 12, main = "Words in Common", unit = NULL,
               top.labels = c("Amzn", "Cons Words", "Google"))
```



INTRO TO TEXT MINING: BAG OF WORDS

Let’s practice!

INTRO TO TEXT MINING: BAG OF WORDS

Reach a conclusion

Time to reach a conclusion!

Problem definition

Which company has better work life balance? Which has better perceived pay according to online reviews?

Unorganized state

Organized state

Insight, recommendation, analytical output

INTRO TO TEXT MINING: BAG OF WORDS

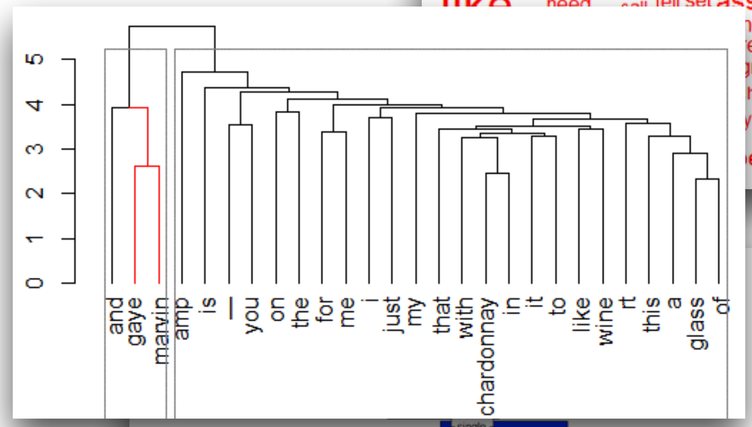
Let’s practice!

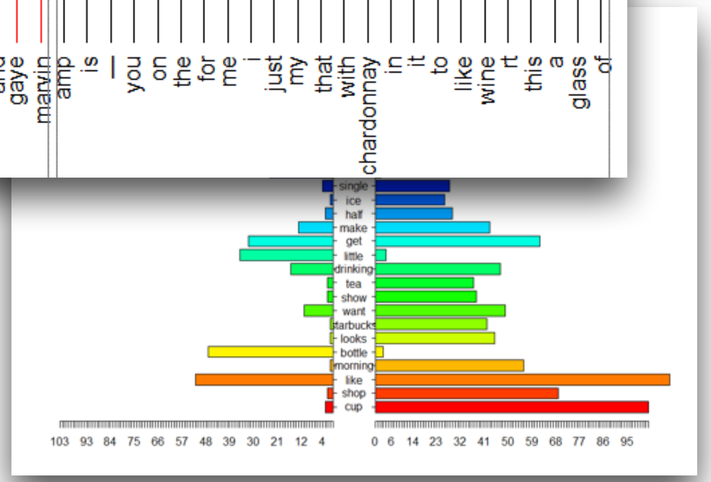
INTRO TO TEXT MINING: BAG OF WORDS

Congratulations!

In this course, you learned how to...

- Organize and clean text data
- Tokenize into unigrams & bigrams
- Build TDMs & DTMs
- Extract features
  - Top terms
  - Word associations
- Visualize text data





Get to work!