Text Mining

Lecture 07

TF = 특정문서
IDF =문서전체

Text Quantification

Keungoui Kim
awekim@handong.edu

Text Quantification

Frequency

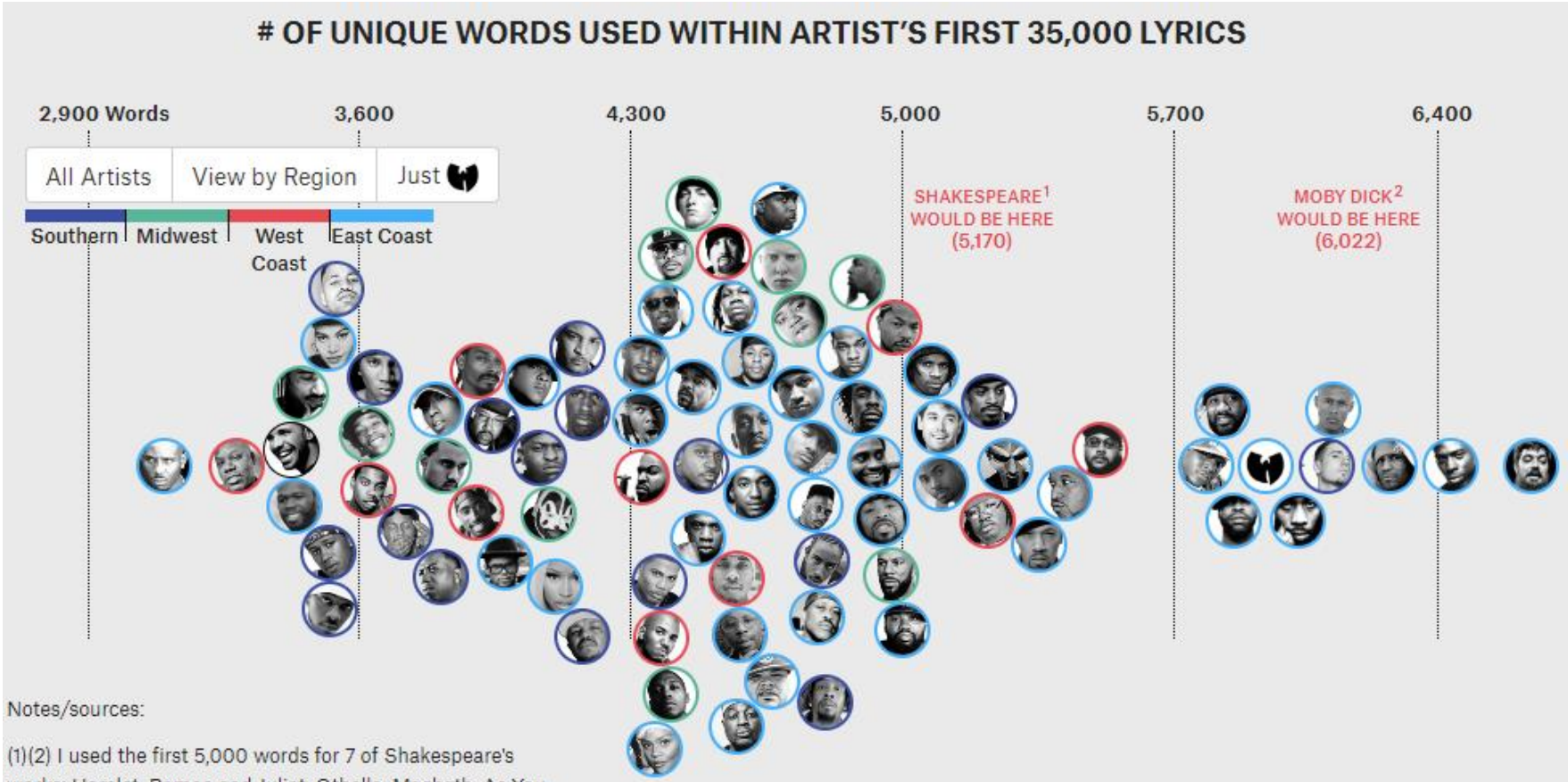
- A simple way of finding the “type” of student
  - Check what is in the bag… (only if agreed)
  - Student A: 5 textbooks, 5 notes, laptop, batteries, etc.
  - Student B: 5 comic books, 3 comic figures, a laptop, batteries, etc.

- Frequency
  - the rate at which something occurs or is repeated over a particular period of time or in a given sample [Oxford]

- Text frequency
  - Frequency can be used to extract words and sentences to represent a document.
  - Text frequency is widely used to determine important (or representative) text

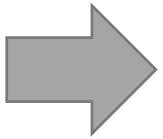
https://pudding.cool/2017/02/vocabulary/

- Ex) The largest vocabulary in Hip Hop



- Bag-of-Words (BoW)
  - A text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity [Wikipedia].

The university was founded in December 1994. The founder, Song Tae-Hun, was an owner of a medium-sized company, and donated land and funds to establish a Christian university. He invited Dr. Kim Young-Gil to be the first president. Dr. Kim was a Christian leader, and had formerly worked as a research scientist at NASA and also as a professor at Korea Advanced Institute of Science and Technology (KAIST).
The university faced many challenges in the beginning. The founder's company went bankrupt while President Kim was recruiting the first professors and students. The university also met local opposition from Pohang citizens who expected the university to serve primarily Pohang residents. When it became known that the university would be a Christian university, recruiting students from all over Korea, many Pohang residents opposed the establishment of the university. This led to lawsuits, and in one of these suits, President Kim was accused of having used government subsidies for purposes which had not been officially approved. He was acquitted and freed after spending 56 days in prison.



Bag of Words

University
founded
December
owner
…
Medium-sized
NASA

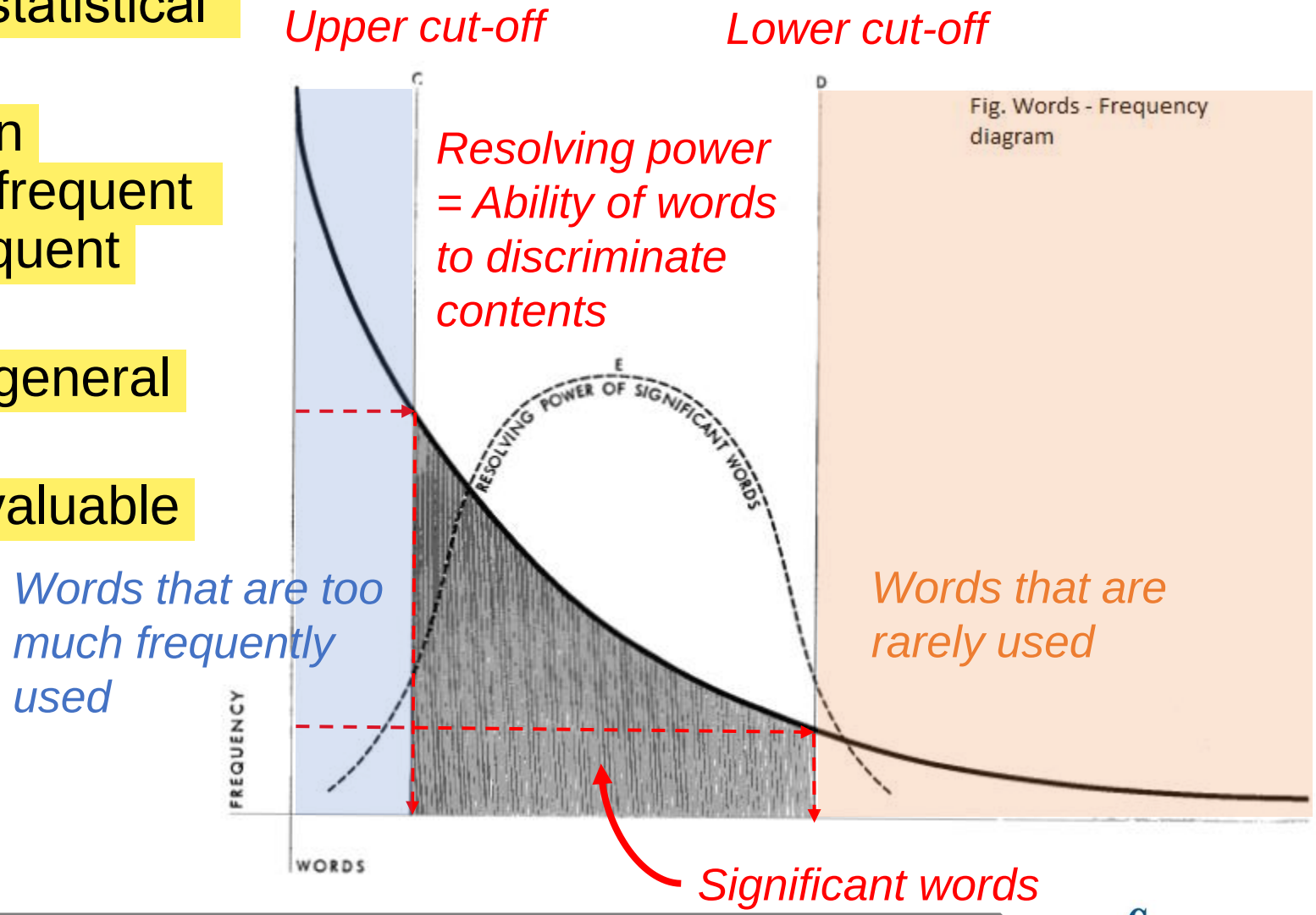
- Zipf's law
  - An empirical law formulated using mathematical statistics that refers to the fact that for many types of data studied in the physical and social sciences, the rank-frequency distribution is an inverse relation [Wikipedia]

$$rank = \frac{1}{frequency}$$

  - Similar patterns are observed in various social science fields that are irrelevant to the text analysis (ex. Zipf's distribution of U.S. Firm Sizes, https://www.science.org/doi/10.1126/science.1062081)

Luhn's Theory

- Luhn (1957)'s idea
  - Analyzing text data with "statistical" features of text frequency
  - Classify words by using an arbitrary "cut-off" → High-frequent mid-frequent, and low-frequent words
  - High-frequent words: too general words
  - Low-frequent words: not valuable words



TF-IDF

Term Frequency, TF

- In text mining & NLP, finding what a document “quantifies” is an important issue.

- Term Frequency, TF
  - Frequency of terms being used in a document
  - Simple and intuitive measure
  - t: term
  - d: document

Inverse Document Frequency, IDF

- Document frequency, DF
  - Number of documents where the term *t* is used

- Inverse document frequency, IDF
  - A measure of whether a term is common or rare in a given document corpus
  - $|D|$: number of documents
  - Logarithm helps reduce the gap between frequently used words and rarely used words
  - Assumes that words that appear frequently in the entire document are of low importance; words that occur only in specific documents are of high importance.

- Term frequency – inverse document frequency (TF-IDF)

Conglomerate
South Korea
Semiconductor

Conglomerate
South Korea
Car

- Words that do not appear in other documents, but do appear in certain documents

→ Words that play an important role in the document

→ Representative text

- TF-IDF is a measure that allows us to capture which word represents a particular document.

- Term frequency – inverse document frequency (TF-IDF)
  - Numerical statistics to reflect how important a term is to a document in a collection or corpus
  - TF-IDF value is proportional to the importance of the term ➔ Greater TF-IDF indicates greater importance of term

Number of occurrences of a term in a document

Number of documents in which the term appears

$$TF - IDF(t,d) = TF(t,d) * IDF(t,d)$$

$$= f_d(t) * log\left(\frac{|D|}{|\{d \epsilon D : t \epsilon d|}\right)$$

Number of documents

** TF-IDF smooth adds 1 to the denominator to avoid denominator equalling 0*

Either 10 or 2

Number of documents that the term appears in

- High TF
  - The term appears many times in the document
  - Important term

- High IDF (low DF)
  - There are few documents in which the term appears
  - A word used less frequently in many documents
  - Like a penalty term

- High TF-IDF
  - Frequently used term in a document, but not much used in other documents

- The key advantage of using TF-IDF is that it allows us to capture the "relatively" important words
  - This simple calculation helps us distinguish between important and unimportant texts
  - Remember that documents can be multiple documents, a document, or a paragraph

  - Multiple documents → how texts are used differently among documents
  - Multiple speakers → how texts are used differently among people

내가 term frequency 해서 따올수있지만 이슈는 문서가 여러개인데 여러개중에서 중요한거 플러스 얘한테 중요한거 대표적인것을 찾고싶은데 IDF라는 패널티 텀을 주는것이다 많이 나온다 하더라도 모든문서에 많이 나오면 나한테 중요한것이 아니다 그개념이 상대적이란것이 뭐냐면 내가 보는 문서 안에서 중요하다는거지 내가 보는 문서에 뭘 더하면 IDF가 바뀐다 내가 보는대상이 무엇인지를 반드시 알고있어야하고
이 안에서상대적으로 중요한녀석이다 아니다라는것을 이야기해야한다.

Text Matrix

- Document-Term Matrix (DTM)
  - Presents frequency of the term by using documents as rows, and terms as columns
  - Document → sample; Term → variable

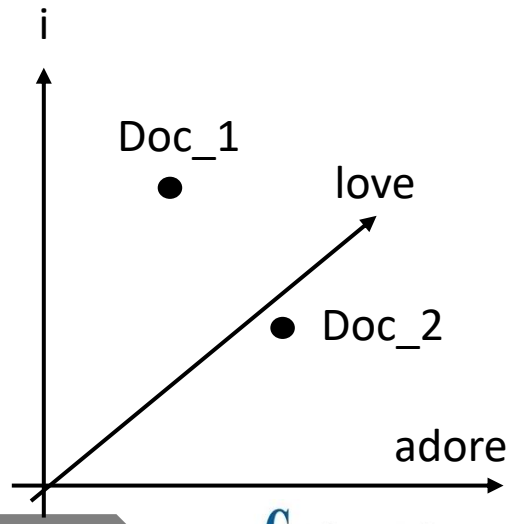
Terms as columns

	Term 1	Term 2	Term 3	…
Document 1	#			
Document 2				
Document 3				
…				

Documents as row

Doc_1: “I love you”
Doc_2: “I adore you you”

	adore	i	love	you
Doc_1	0	1	1	1
Doc_2	1	1	0	2



- Sparsity
  - Sparsity of matrix (or sparse matrix): a matrix where most of elements are zero
  - Sparsity = 1 → all zero elements
  - Sparsity = 0 → no zero elements

	adore	i	love	you
Doc_1	0	1	1	1
Doc_2	1	1	0	2

2/8 = 0.25
➔ 25% sparsity

- Text Sparsity
  - Text sparsity = 1 → terms appear in all cases
  - Text sparsity = 0 → terms do not appear in all cases

$$sparsity = \frac{Number\ of\ terms\ with\ 0\ frequency}{Total\ number\ of\ terms}$$

Document-Term Matrix in R

- Create DTM using tm::DocumentTermMatrix()
  - By default, it creates DTM with terms greater than length 3
  - Returns DTM object
  - (If the source is data.frame) Create DataframeSource → Create corpus → create DTM object

```
> library(tm)
> ex.df <-
+   data.frame(
+     doc_id=c("Doc_1","Doc_2"),
+     text=c("I love you",
+            "I adore you you"))
> ex.df
  doc_id             text
1  Doc_1       I love you
2  Doc_2 I adore you you
> ex.df.dtm <-
+   ex.df %>%
+   DataframeSource %>%
+   Corpus %>%
+   DocumentTermMatrix(control =
+                        list(wordLengths=c(1, Inf)))
> ex.df.dtm %>%
+   inspect
<<DocumentTermMatrix (documents: 2, terms: 4)>>
Non-/sparse entries: 6/2
Sparsity           : 25%
Maximal term length: 5
Weighting          : term frequency (tf)
Sample             :
       Terms
Docs    adore i love you
  Doc_1     0 1    1   1
  Doc_2     1 1    0   2
```

Allows us to create a DTM including character less than length 2

25% of terms have 0 frequency

Weight of matrix is measured with tf

Document-Term Matrix in R

- Measuring TF-IDF with DTM
  - Either level or normalized version
  - *weightTfIdf* uses log_2

	adore	i	love	you
Doc_1	0	1	1	1
Doc_2	1	1	0	2

$$TfIdf(t,d) = f_d(t) * log_2\left(\frac{|D|}{|\{d \epsilon D : t \epsilon d|}\right)$$

	adore	i	love	You
Doc_1	0 * log2(2/1) = 0	1 * log2(2/2) = 0	1 * log2(2/1) = 1	1 * log2(2/2) = 0
Doc_2	1 * log2(2/1) = 1	1 * log2(2/2) = 0	0 * log2(2/1) = 0	2 * log2(2/2) = 0

$$weightTfIdf(t,d,normalized) = \frac{f_d(t)}{\sum_k f_d(t_k)} * log_2\left(\frac{|D|}{|\{d \epsilon D : t \epsilon d|}\right)$$

k = total number of terms in document d

	adore	I	love	You
Doc_1	0 * log2(2/1) = 0	1 * log2(2/2) = 0	1/3 * log2(2/1) = 0.33	1 * log2(2/2) = 0
Doc_2	1/4 * log2(2/1) = 0.25	1 * log2(2/2) = 0	0 * log2(2/1) = 0	2 * log2(2/2) = 0

Document-Term Matrix in R

- Measuring weighted TF-IDF with DTM
  - Either level or normalized version

```
> ex.df %>%
+   DataframeSource %>%
+   Corpus %>%
+   DocumentTermMatrix(
+     control = list(wordLengths=c(1, Inf),
+                    weighting=function(x)
+                      weightTfIdf(x, normalize = FALSE))) %>%
+   inspect
<<DocumentTermMatrix (documents: 2, terms: 4)>>
Non-/sparse entries: 2/6
Sparsity           : 75%
Maximal term length: 5
Weighting          : term frequency - inverse document frequency (tf-idf)
Sample             :
       Terms
Docs    adore i love you
  Doc_1     0 0    1   0
  Doc_2     1 0    0   0
```

Document-Term Matrix in R

- Measuring weighted TF-IDF with DTM
  - Either level or normalized version

```
> ex.df %>%
+   DataframeSource %>%
+   Corpus %>%
+   DocumentTermMatrix(
+     control = list(wordLengths=c(1, Inf),
+                    weighting=function(x)
+                      weightTfIdf(x, normalize = TRUE))) %>%
+   inspect
<<DocumentTermMatrix (documents: 2, terms: 4)>>
Non-/sparse entries: 2/6
Sparsity           : 75%
Maximal term length: 5
Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
Sample             :
       Terms
Docs    adore i      love you
  Doc_1  0.00 0 0.3333333   0
  Doc_2  0.25 0 0.0000000   0
```

Document-Term Matrix in R

- Comparison of TF and TF-IDF

```
> # Comparison of TF and TF-IDF
> comp.table <- data.frame(
+   doc = rep(rownames(ex.df.dtm), dim(ex.df.dtm)[2]),
+   term = rep(colnames(ex.df.dtm) %>% sort(decreasing=FALSE),
+              each=dim(ex.df.dtm)[1]),
+   TF = c(0,1,1,1,1,0,1,2),
+   TF_IDF = as.vector(ex.df.dtm.tfidf),
+   W_TF_IDF = as.vector(ex.df.dtm.tfidf.w) %>% round(3))
> comp.table
    doc  term TF TF_IDF W_TF_IDF
1 Doc_1 adore  0      0    0.000
2 Doc_2 adore  1      1    0.250
3 Doc_1     i  1      0    0.000
4 Doc_2     i  1      0    0.000
5 Doc_1  love  1      1    0.333
6 Doc_2  love  0      0    0.000
7 Doc_1   you  1      0    0.000
8 Doc_2   you  2      0    0.000
```

High TF does not guarantee high TF-IDF

→ Correlation between TF and TF-IDF is not high

- Term-Document Matrix (TDM)
  - Presents frequency of the term by using terms as rows, and documents as columns
  - Term → sample; Document → variable

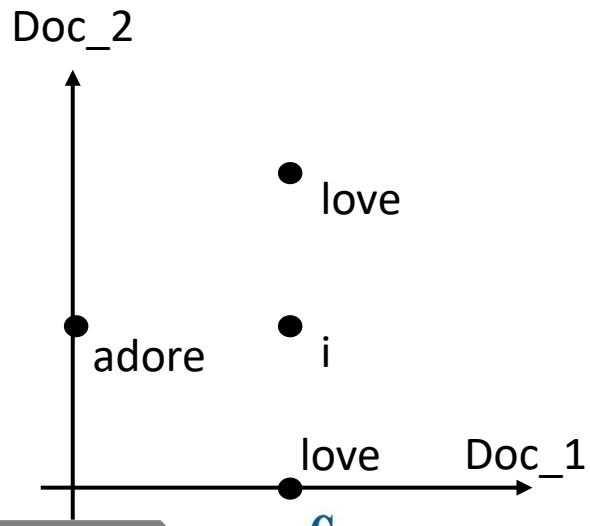
Documents as columns

Terms as row

	Document 1	Document 2	Document 3	…
Term 1	#			
Term 2				
Term 3				
…				

Doc_1: “I love you”
Doc_2: “I adore you you”

	Doc_1	Doc_2
adore	0	1
i	1	1
love	1	0
you	1	2



Term-Document Matrix in R

- Create DTM using tm::TermDocumentMatrix()
  - By default, it creates TDM with terms greater than length of 3
  - TDM object
- tm package's Function logic for generating DTM with data.frame
  - Create DataframeSource → Create corpus → create DTM object

```
> ex.df.tdm <-
+   ex.df %>%
+   DataframeSource %>%
+   Corpus %>%
+   TermDocumentMatrix(control =
+                        list(wordLengths=c(1, Inf)))
> ex.df.tdm %>%
+   inspect
<<TermDocumentMatrix (terms: 4, documents: 2)>>
Non-/sparse entries: 6/2
Sparsity           : 25%
Maximal term length: 5
Weighting          : term frequency (tf)
Sample             :
        Docs
Terms    Doc_1 Doc_2
  adore      0     1
  i          1     1
  love       1     0
  you        1     2
```

DTM ? TDM?

- Which matrix should I use? DTM? TDM?
  - Whether to select DTM or TDM for analysis is determined by the data analyst's goal
  - Analyzing with DTM → Using **terms** as analysis features

    → More interested in finding a relationship between **documents**
  - Analyzing with TDM → Using **documents** as analysis features

    → More interested in finding a relationship between **terms**

- Spare representation
  - Many zeros in the matrix → not a good sign for calculation
  - In the case of DTM, we assume that columns are all the terms used in the multiple documents. It's obvious that most of the listed terms may not occur in all cases
  - We cannot prevent this issue, but we can try to minimize the problem by text-preprocessing

Doc_1: "I love you."
Doc_2: "I adore you YOU"

	adore	I	love	you.	you	YOU
Doc_1	0	1	1	1	0	0
Doc_2	1	1	0	0	1	1

Reduce dimension & sparsity

Text
pre-processing

	adore	i	love	you
Doc_1	0	1	1	1
Doc_2	1	1	0	2

Text Matrix with n-gram

Text Matrix with n-gram

- DTM (or TDM) + n-gram tokenizer
  - (Step 1) Create an n-gram tokenizer function
  - With RWeka::NGramTokenizer(), we can create an n-gram tokenizer with more than 2 ns (ex. bigram + trigram)

Name of function

Argument

```
bigramTokenizer <- function( x ){
    RWeka::NGramTokenizer(x, Rweka::Weka_control(min=2, max=2))
}
```

n-gram setting

By changing min and max value, we can have bigram and trigram at the same time (min=2, max=3)

Text Matrix with n-gram

- DTM (or TDM) + n-gram tokenizer
  - (Step 2) Create a DTM with tm::DocumentTermMatrix()
  - Using the tokenize option, we can create an n-gram DTM

```
corpus %>%
      DocumentTermMatrix( control = list ( tokenize = bigramTokenizer ) )
```

Function made from the previous slide

```
corpus %>%
      TermDocumentMatrix( control = list ( tokenize = bigramTokenizer ) )
```

Text Matrix with n-gram

- Import and create an n-gram tokenizer function
  - Import .text file with readLines()
  - To create a corpus with vector, use tm::VectorSource()

```
> mlk <-
+   readLines("R file/R file_LEC07/mlk_speech.txt")
> mlk <- mlk[mlk != " "]
> mlk <- mlk[mlk != ""]
> mlk.corpus <-
+   mlk %>%
+   VectorSource %>%
+   VCorpus
> bigramTokenizer <- function(x) {
+   RWeka::NGramTokenizer(x,
+                         RWeka::Weka_control(min=2, max=2))}
```

Import text data

Create a vector source

Create n-gram tokenizer function

Text Matrix with n-gram

- n-gram with DTM / TDM

```
> mlk.corpus %>%
+   DocumentTermMatrix(control=
+                        list(tokenize=bigramTokenizer)) %>%
+   inspect
<<DocumentTermMatrix (documents: 30, terms: 406)>>
Non-/sparse entries: 530/11650
Sparsity           : 96%
Maximal term length: 23
Weighting          : term frequency (tf)
Sample             :
    Terms
Docs a dream able to be able freedom ring have a i have let fr
  1         2         0       0             0      1      0
  10        1         0       0             0      1      1
  12        0         3       3             0      0      0
  13        0         1       1             0      0      0
  2         1         0       0             0      1      1
  28        0         2       2             1      0      0
  4         1         1       1             0      1      1
  5         1         0       0             0      1      1
  6         1         0       0             0      1      1
  8         1         1       1             0      1      1
```

```
> mlk.corpus %>%
+   TermDocumentMatrix(control=
+                        list(tokenize=bigramTokenizer)) %>%
+   inspect
<<TermDocumentMatrix (terms: 406, documents: 30)>>
Non-/sparse entries: 530/11650
Sparsity           : 96%
Maximal term length: 23
Weighting          : term frequency (tf)
Sample             :
             Docs
Terms         1 10 12 13 2 28 4 5 6 8
 a dream      2  1  0  0 1  0 1 1 1 1
 able to      0  0  3  1 0  2 1 0 0 1
 be able      0  0  3  1 0  2 1 0 0 1
 freedom ring 0  0  0  0 0  1 0 0 0 0
 have a       1  1  0  0 1  0 1 1 1 1
 i have       0  1  0  0 1  0 1 1 1 1
 let freedom  0  0  0  0 0  0 0 0 0 0
 one day      0  1  1  0 1  0 1 1 1 2
 ring from    0  0  0  0 0  1 0 0 0 0
 will be      0  2  4  3 0  2 1 1 0 1
```

Text Mining DTM

Import Data

- crude data
  - Includes 20 news articles with additional meta information from the Reuters-21578 data set. All documents belong to the topic ”crude,” which deals with crude oil

```
> data("crude")
> crude %>% length
[1] 20
> crude %>% summary
    Length Class             Mode
127 2      PlainTextDocument list
144 2      PlainTextDocument list
191 2      PlainTextDocument list
194 2      PlainTextDocument list
211 2      PlainTextDocument list
```

```
> crude[[1]]$content
[1] "Diamond Shamrock Corp said that\neffective today it had cut its contract prices for crude o
il by\n1.50 dlrs a barrel.\n    The reduction brings its posted price for West Texas\nIntermedia
te to 16.00 dlrs a barrel, the copany said.\n    \"The price reduction today was made in the lig
ht of falling\noil product prices and a weak crude oil market,\" a company\nspokeswoman said.\n
   Diamond is the latest in a line of U.S. oil companies that\nhave cut its contract, or posted,
 prices over the last two days\nciting weak oil markets.\n Reuter"
```

- Simple text pre-processing

```
> library(textstem)
> crude.cleaned <- crude %>%
+   tm_map(removePunctuation) %>%
+   tm_map(removeNumbers) %>%
+   tm_map(removeWords, stopwords('en')) %>%
+   tm_map(stripWhitespace) %>%
+   tm_map(content_transformer(lemmatize_strings)) %>%
+   tm_map(content_transformer(tolower))
> crude.cleaned[[1]]$content
[1] "diamond shamrock corp say effective today cut contract price crude oil dlrs barrel the redu
ction bring post price west texas intermediate dlrs barrel copany say the price reduction today
 make light fall oil product price weak crude oil market company spokeswoman say diamond late li
ne us oil company cut contract post price last two day cite weak oil market reuter"
```

- Create DTM

```
> crude.dtm <-
+   crude.cleaned %>%
+   DocumentTermMatrix()
> crude.dtm %>%
+   inspect
<<DocumentTermMatrix (documents: 20, terms: 820)>>
Non-/sparse entries: 1619/14781
Sparsity           : 90%
Maximal term length: 16
Weighting          : term frequency (tf)
Sample             :
     Terms
Docs  barrel bpd last market mln oil opec price say the
  144      0   4    1      5   4  12   13     6  11   2
  236      4   7    4      3   4   7    6     8  11   0
  237      0   0    3      0   1   3    1     1   9   1
  242      0   0    0      2   0   3    2     2   3   1
  246      1   0    2      0   0   5    1     2   6   3
  248      3   2    1     10   3   9    6    10   8   1
  273      3   8    7      1   9   5    5     5   8   4
  489      3   0    0      0   3   4    0     3   2   1
  502      3   0    0      0   3   5    0     3   3   2
  704      0   0    0      3   0   3    0     3   4   4
```

```
> crude.dtm$dimnames$Terms
  [1] "abdulaziz"        "ability"          "able"
  [5] "accept"           "accord"           "across"
  [9] "add"              "address"          "adhere"
 [13] "advantage"        "adviser"          "after"
 [17] "agree"            "agreement"        "agricultural"
 [21] "aground"          "ali"              "alkhalifa"
```

```
> crude.dtm$dimnames$Docs
 [1] "127" "144" "191" "194" "211" "236" "237" "242" "246"
[15] "368" "489" "502" "543" "704" "708"
> crude.dtm %>%
+   as.matrix
     Terms
Docs  abdulaziz ability able abroad accept accord across a
  127         0       0    0      0      0      0      0
     Terms
Docs  adherence advantage adviser after agency agree agree
  127         0         0       0     0      0     0
```

Fix Sparse Representation Issue

- tm::removeSparseTerms(*DTM or TDM*, *sparse*)
  - Remove sparce terms
  - sparse: threshold of relative document frequency (proportion) for a term. Terms above the threshold are removed (0 < sparse < 1)

$$sparsity = \frac{Number\ of\ terms\ with\ 0\ frequency}{Total\ number\ of\ terms}$$

```
> crude.dtm %>%
+   removeSparseTerms(0.9) %>%
+   inspect
<<DocumentTermMatrix (documents: 20, terms: 301)>>
Non-/sparse entries: 1100/4920
Sparsity           : 82%
Maximal term length: 13
Weighting          : term frequency (tf)
Sample             :
     Terms
Docs  barrel bpd last market mln oil opec price say the
  144      0   4    1      5   4  12   13     6  11   2
  236      4   7    4      3   4   7    6     8  11   0
  237      0   0    3      0   1   3    1     1   9   1
  246      1   0    2      0   0   5    1     2   6   3
  248      3   2    1     10   3   9    6    10   8   1
  273      3   8    7      1   9   5    5     5   8   4
  352      1   0    1      2   0   5    2     5   2   0
  489      3   0    0      0   3   4    0     3   2   1
  502      3   0    0      0   3   5    0     3   3   2
  704      0   0    0      3   0   3    0     3   4   4
```

Remove terms that are more sparse than 0.9

```
> crude.dtm %>%
+   removeSparseTerms(0.1) %>%
+   inspect
<<DocumentTermMatrix (documents: 20, terms: 3)>>
Non-/sparse entries: 60/0
Sparsity           : 0%
Maximal term length: 6
Weighting          : term frequency (tf)
Sample             :
     Terms
Docs  oil reuter say
  127   5      1   3
  144  12      1  11
  236   7      1  11
  237   3      1   9
  246   5      1   6
  248   9      1   8
  273   5      1   8
  352   5      1   2
  502   5      1   3
  543   3      1   4
```

Remove terms that are more sparse than 0.1

Create Bag-of-Words

- Create BoW with DTM
  1) Covert to matrix
  2) Measure tf
  3) Convert to data.frame

```
> crude.dtm.mat <-
+   crude.dtm %>% as.matrix
> crude.dtm.mat[1,1:8]
abdulaziz   ability      able    abroad         a
        0         0         0         0
> dim(crude.dtm.mat)
[1]  20 820
> crude.dtm.mat %<>% colSums %>%
+   sort(decreasing=TRUE)
> crude.dtm.mat[1:10]
   say    oil  price   opec    mln market ba
    86     85     63     42     31     30
> crude.dtm.mat.df <-
+   data.frame(word=names(crude.dtm.mat),
+              freq=crude.dtm.mat)
> crude.dtm.mat.df %>%
+   head()
          word freq
say        say   86
oil        oil   85
price    price   63
opec      opec   42
mln        mln   31
market  market   30
```

Convert to matrix

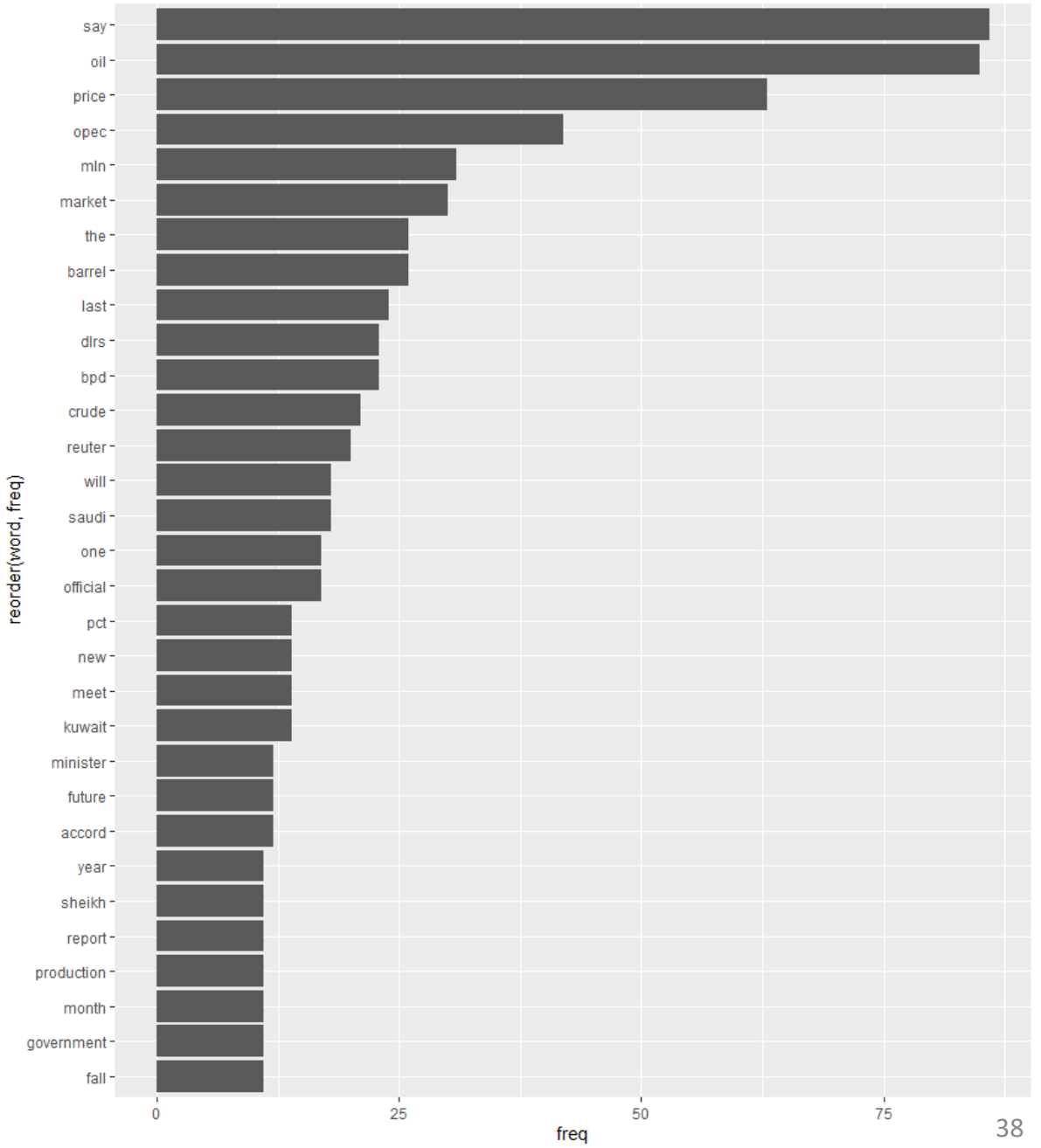
Measure tf

Convert to data.frame

Visualization

- Check the distribution of terms that appeared more than 10 times

```
library(ggplot2)
crude.dtm.mat.df %>%
  filter(freq>10) %>%
  ggplot(aes(x=reorder(word,freq),y=freq)) +
  geom_bar(stat='identity') + coord_flip()
```



Compare TF & TF-IDF

- Create doc-word-tf table

```
> crude.dtm.mat <-
+   crude.cleaned %>%
+   DocumentTermMatrix %>%
+   as.matrix
> crude.dtm.mat.df <- 0
> for(i in 1:nrow(crude.dtm.mat)){
+   temp <- crude.dtm.mat[i,]
+   temp <- data.frame(
+     doc=rownames(crude.dtm.mat)[i],
+     word=names(temp),
+     tf=temp,
+     row.names = NULL)
+   crude.dtm.mat.df <-
+     rbind(crude.dtm.mat.df, temp)
+   rm(temp)
+ }
> crude.dtm.mat.df <-
+   crude.dtm.mat.df[2:nrow(crude.dtm.mat.df),]
```

```
> crude.dtm.mat.df %>%
+   group_by(doc) %>%
+   arrange(desc(tf)) %>%
+   slice(1)
# A tibble: 20 x 3
# Groups:   doc [20]
   doc   word          tf
   <chr> <chr>      <dbl>
 1 127   oil            5
 2 144   opec          13
 3 191   canada         2
 4 194   crude          3
 5 211   estimate       3
 6 236   say           11
 7 237   say            9
 8 242   yesterday      4
 9 246   billion        7
10 248   market        10
11 273   mln            9
12 349   oil            4
13 352   oil            5
14 353   oil            4
15 368   power          4
16 489   oil            4
17 502   oil            5
18 543   dlrs           5
19 704   future         9
20 708   january        4
```

Compare TF & TF-IDF

- Create doc-word-tfidf table

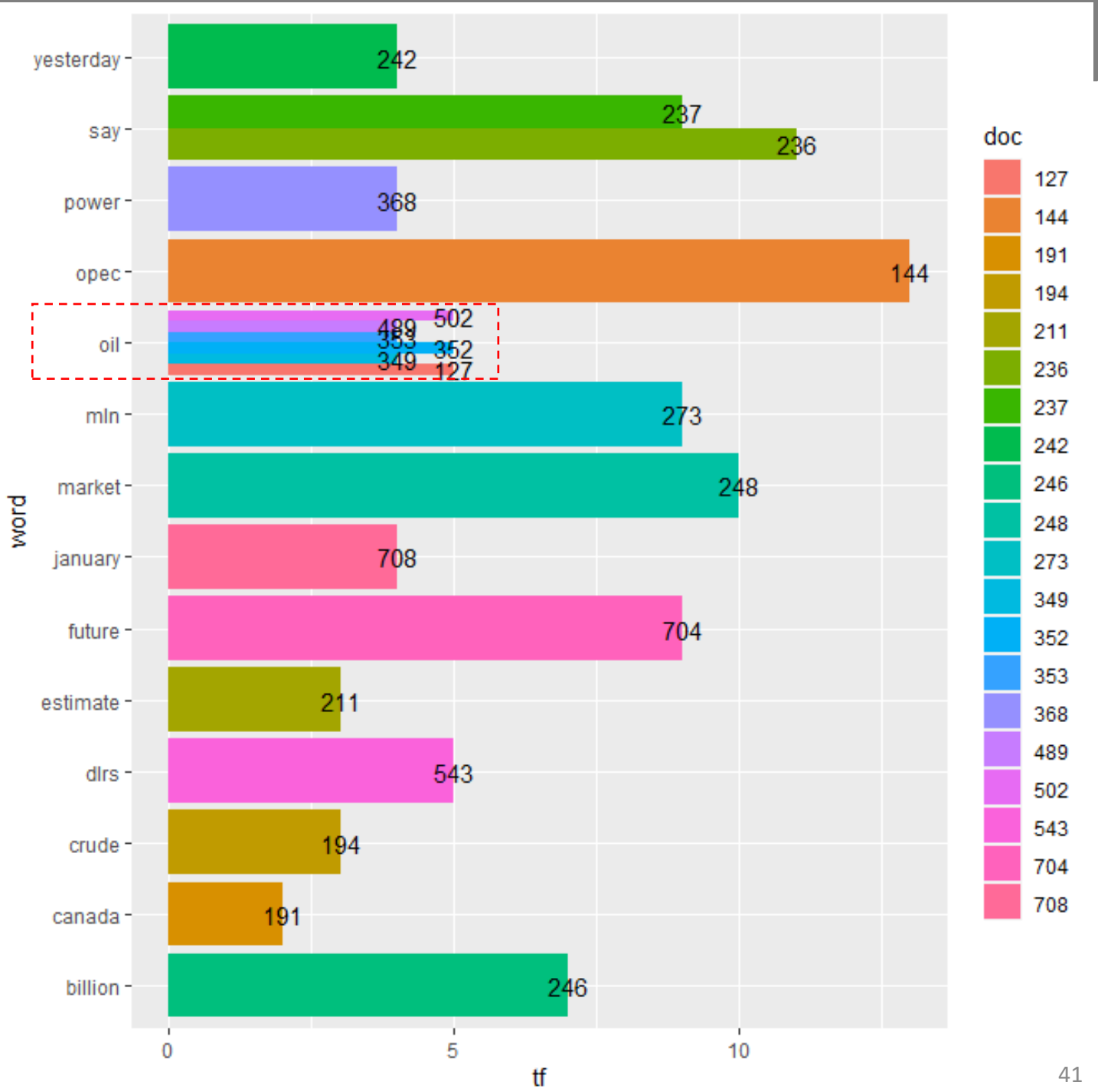
```
> crude.dtm.tfidf <-
+   crude.cleaned %>%
+   DocumentTermMatrix(
+     control=list(weighting=function(x)
+       weightTfIdf(x, normalize=FALSE))) %>%
+   as.matrix
> crude.dtm.tfidf.mat.df <- 0
> for(i in 1:nrow(crude.dtm.tfidf.mat)){
+   temp <- crude.dtm.tfidf.mat[i,]
+   temp <- data.frame(
+                doc=rownames(crude.dtm.tfidf.mat)[i],
+                word=names(temp),
+                tfidf=temp,
+                row.names = NULL)
+   crude.dtm.tfidf.mat.df <-
+     rbind(crude.dtm.tfidf.mat.df, temp)
+   rm(temp)
+ }
> crude.dtm.tfidf.mat.df <-
+   crude.dtm.tfidf.mat.df[2:nrow(crude.dtm.tfidf.mat.df),]
```

```
> crude.dtm.tfidf.mat.df %>%
+   group_by(doc) %>%
+   arrange(desc(tfidf)) %>%
+   slice(1)
# A tibble: 20 x 3
# Groups:   doc [20]
   doc   word      tfidf
   <chr> <chr>     <dbl>
 1 127   diamond    8.64
 2 144   problem   25.9
 3 191   canada     8.64
 4 194   marathon   8.64
 5 211   trust     13.0
 6 236   kuwait    18
 7 237   growth    21.6
 8 242   yesterday 13.3
 9 246   budget    25.9
10 248   accord    11.6
11 273   average   17.3
12 349   discuss    8.64
13 352   saudi      8
14 353   pump       6.64
15 368   power     17.3
16 489   increase   8.21
17 502   benefit    8.64
18 543   union     13.0
19 704   nymex     30.3
20 708   january   13.3
```

Visualization

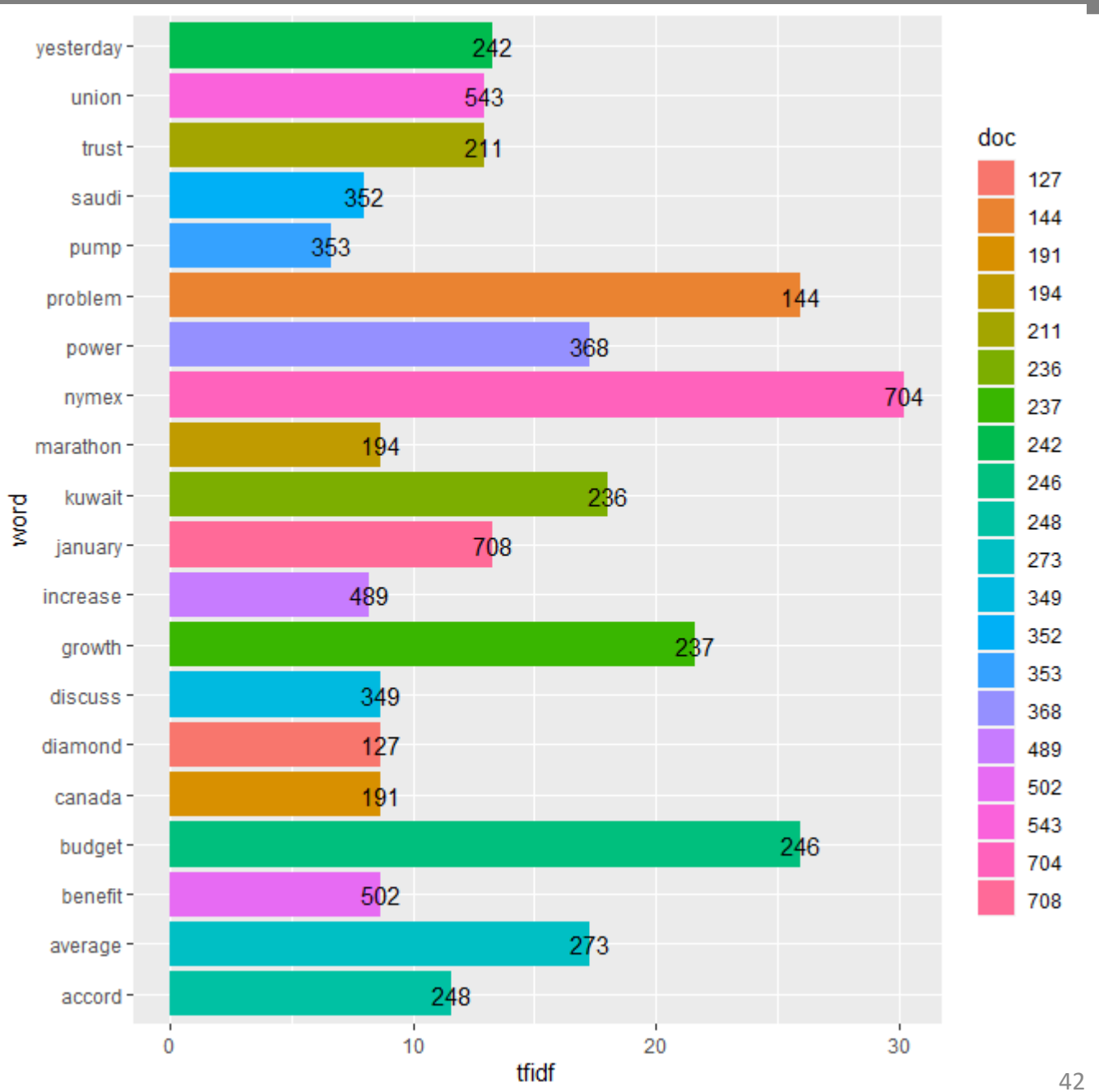
- Check the top tf distribution

```
crude.dtm.mat.df %>%
  group_by(doc) %>%
  arrange(desc(tf)) %>%
  slice(1) %>%
  ggplot(aes(x=word,y=tf, fill=doc))+
  geom_bar(stat='identity',
           position='dodge') +
  geom_text(aes(x=word, y=tf, group=doc, label=doc),
                position=position_dodge(0.9)) +
  coord_flip()
```



- Check the top tf-idf distribution

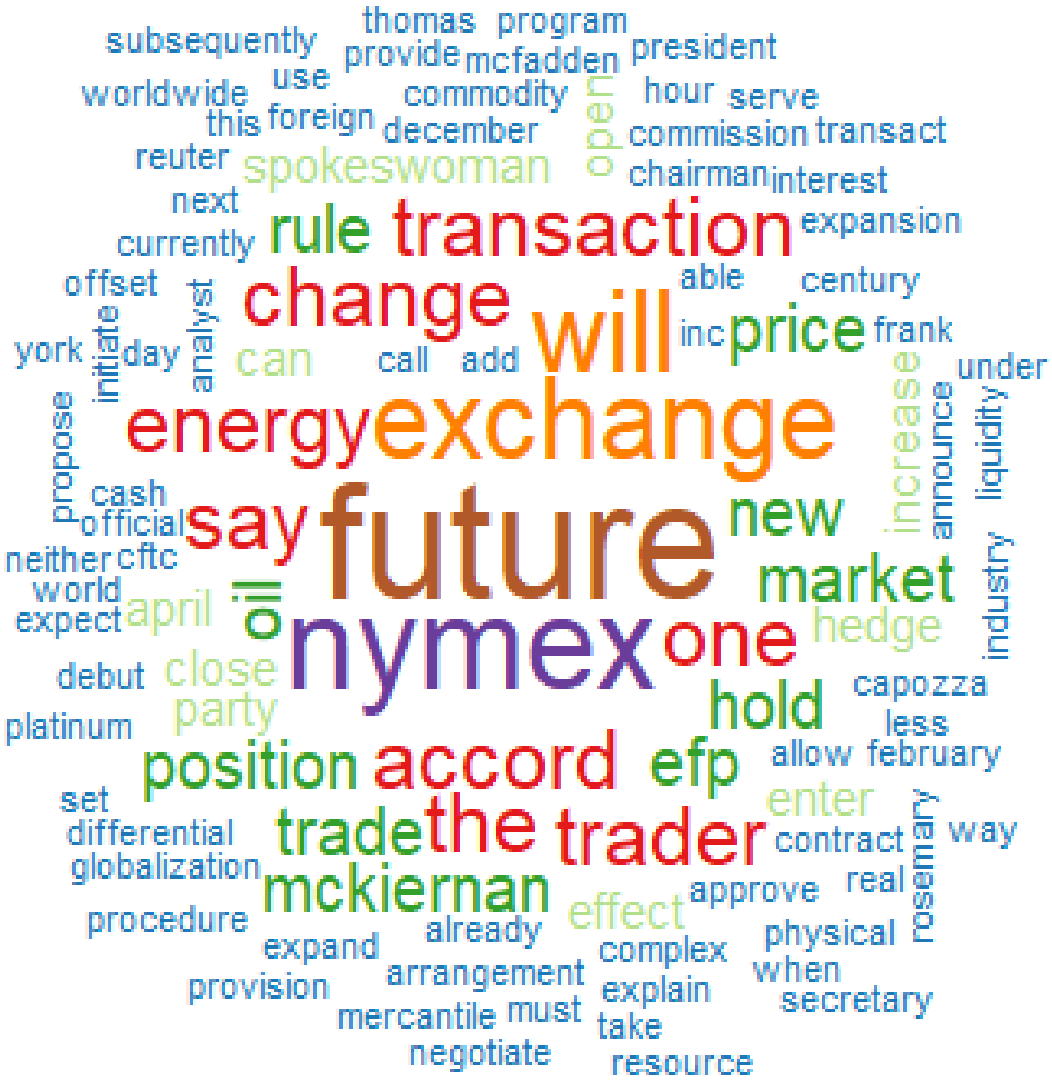
```
crude.dtm.tfidf.mat.df %>%
  group_by(doc) %>%
  arrange(desc(tfidf)) %>%
  slice(1) %>%
  ggplot(aes(x=word,y=tfidf, fill=doc))+
  geom_bar(stat='identity') +
  geom_text(aes(label=doc)) +
  coord_flip()
```



Word Cloud

Word Cloud

- Word cloud (also known as a tag cloud)
  - a visual representation of text data
  - Simple but very intuitively visualized work
  - Size indicates the frequency of words (the bigger the word, the higher the frequency)
  - Color can indicate either the frequency level or text type (obtained from the lexical resource)

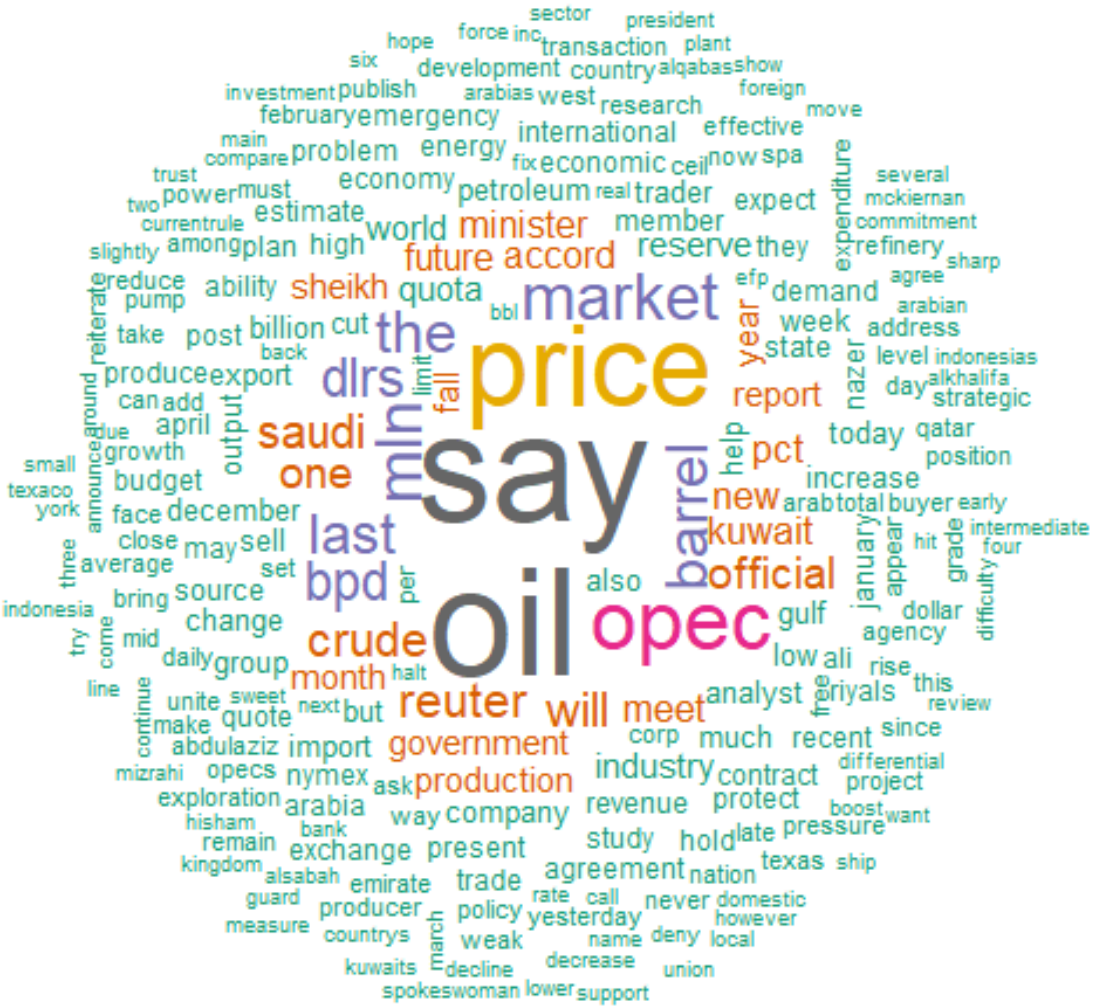


Word Cloud in R

- wordcloud::wordcloud(*words*, *freq*)
  - Returns word cloud graphics
  - *words* & *freq* are vectors
- Creating a wordcloud with DTM
  - Create data.frame containing *words* and *freq* vector

```
> crude.dtm.mat <-
+   crude.dtm %>% as.matrix
> crude.dtm.mat %<>% colSums %>%
+   sort(decreasing=TRUE)
> crude.dtm.mat.df <-
+   data.frame(word=names(crude.dtm.mat),
+              freq=crude.dtm.mat)
> head(crude.dtm.mat.df)
         word freq
say       say   86
oil       oil   85
price   price   63
opec     opec   42
mln       mln   31
market market   30
```

```
> set.seed(1004)
> library(wordcloud)
> wordcloud(
+   words=crude.dtm.mat.df$word,
+   freq=crude.dtm.mat.df$freq,
+   random.order=FALSE,
+   colors=brewer.pal(8, "Dark2")
+ )
```



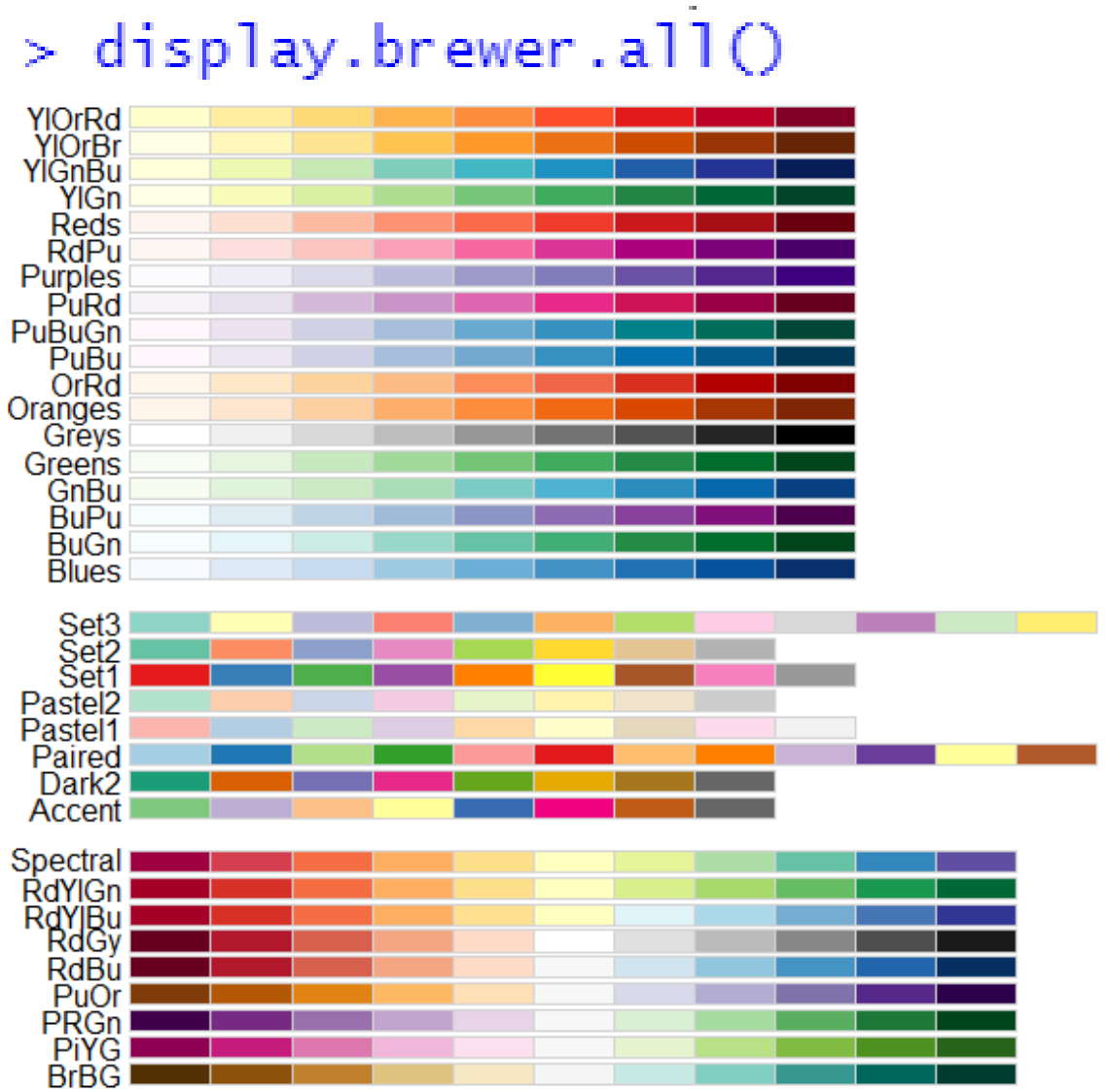
Word Cloud in R

- wordcloud2:: wordcloud2(*data*)
  - Returns word cloud html widget

```
> set.seed(1004)
> library('wordcloud2')
> wordcloud2(
+   data=crude.dtm.mat.df,
+   size=2.0,
+   color='random-dark'
+ )
```

- brewer.pal(*n*, *name*)
  - n: number of colors in palette
  - name: name of palette

Accent	8
Dark2	8
Paired	12
Pastel1	9
Pastel2	8
Set1	9
Set2	8
Set3	12



Word Cloud for Multiple Documents

- Create wordcloud figures for multiple documents

```
for(i in 1:nrow(crude.dtm)){
  temp <-
    crude.dtm[i,] %>%
    as.matrix
  temp %<>% colSums %>%
    sort(decreasing=TRUE)
  temp <-
    data.frame(word=names(temp),
               freq=temp)
  png(file=paste0("R file/R file_LEC07/wordcloud/wordcloud_",i,".png"),
      width=600, height=350)
  wordcloud(
      words=temp$word,
      freq=temp$freq,
      min.freq=1,
      random.order=FALSE,
      colors=brewer.pal(12, "Paired"))
  dev.off()
  rm(temp)
}
```

Create word-frequency data.frame

Save wordcloud .png file

Word Cloud for Multiple Documents

- Create wordcloud2 figures for multiple documents

```
library(htmlwidgets)
library(webshot)
webshot::install_phantomjs()
set.seed(1004)
wordcloud2.ls <- list()
for(i in 1:nrow(crude.dtm)){
  temp <-
    crude.dtm[i,] %>%
    as.matrix
  temp %<>% colSums %>%
    sort(decreasing=TRUE)
  temp <-
    data.frame(word=names(temp),
               freq=temp)
  wordcloud2.ls[[i]] <-
    wordcloud2(
      data=temp,
      size=2.0,
      color='random-dark'
    )
  saveWidget(wordcloud2.ls[[i]],
             paste0("R file/R file_LEC07/wordcloud2/wordcloud2_",i,".html"),
             selfcontained = F)
  webshot(url=paste0("R file/R file_LEC07/wordcloud2/wordcloud2_",i,".html"),
          file=paste0("R file/R file_LEC07/wordcloud2/wordcloud2_",i,".png"),
          delay = 10, vwidth = 2000, vheight = 2000)
  rm(temp)
}
```

Create word-frequency data.frame

Save html widget

Save the screenshot of the html

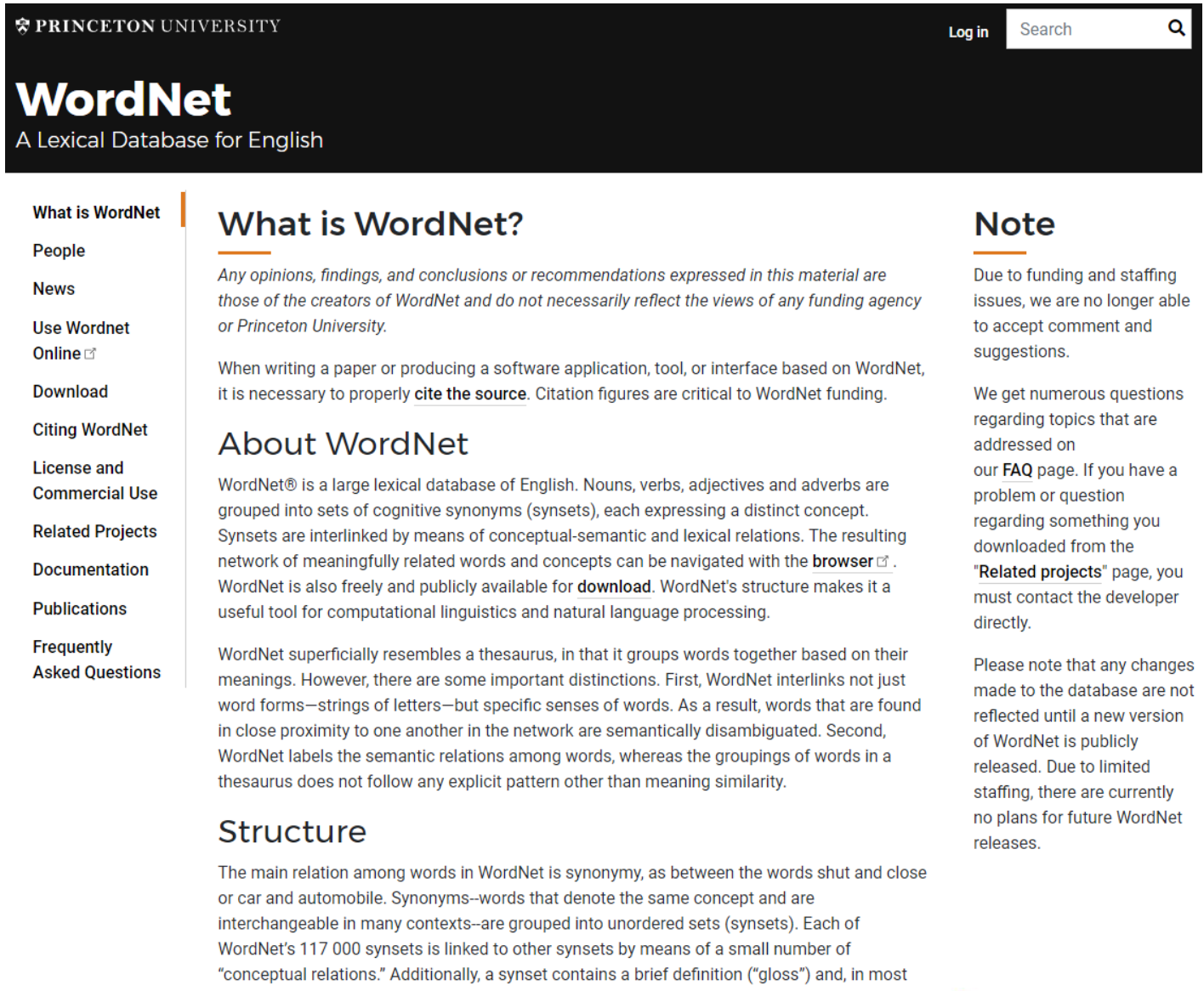
Lexical Resource: WordNet

Lexical Resource

- Lexical resource
  - A collection of lexical items with additional linguistic information
  - Such as dictionary, thesaurus, semantic classes

- In discrete representation, lexical resources are often used to add "meanings"
  - Text classification
  - Text relatedness
  - Sentiment analysis

- WordNet
  - Developed by a research team led by George Miller (https://wordnet.princeton.edu/)
  - Covers majority of nouns, verbs, adjectives, adverbs
- Versions
  - Wordnet 2.1: the most recent window version
  - Wordnet 3.0: available for Unix/Linux/Solaris
  - Wordnet 3.1: only available online.



PRINCETON UNIVERSITY

Log in Search

WordNet

A Lexical Database for English

What is WordNet
People
News
Use Wordnet Online
Download
Citing WordNet
License and Commercial Use
Related Projects
Documentation
Publications
Frequently Asked Questions

What is WordNet?

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the creators of WordNet and do not necessarily reflect the views of any funding agency or Princeton University.

When writing a paper or producing a software application, tool, or interface based on WordNet, it is necessary to properly **cite the source**. Citation figures are critical to WordNet funding.

About WordNet

WordNet® is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the **browser**. WordNet is also freely and publicly available for **download**. WordNet's structure makes it a useful tool for computational linguistics and natural language processing.

WordNet superficially resembles a thesaurus, in that it groups words together based on their meanings. However, there are some important distinctions. First, WordNet interlinks not just word forms—strings of letters—but specific senses of words. As a result, words that are found in close proximity to one another in the network are semantically disambiguated. Second, WordNet labels the semantic relations among words, whereas the groupings of words in a thesaurus does not follow any explicit pattern other than meaning similarity.

Structure

The main relation among words in WordNet is synonymy, as between the words shut and close or car and automobile. Synonyms--words that denote the same concept and are interchangeable in many contexts--are grouped into unordered sets (synsets). Each of WordNet's 117 000 synsets is linked to other synsets by means of a small number of "conceptual relations." Additionally, a synset contains a brief definition ("gloss") and, in most

Note

Due to funding and staffing issues, we are no longer able to accept comment and suggestions.

We get numerous questions regarding topics that are addressed on our **FAQ** page. If you have a problem or question regarding something you downloaded from the "**Related projects**" page, you must contact the developer directly.

Please note that any changes made to the database are not reflected until a new version of WordNet is publicly released. Due to limited staffing, there are currently no plans for future WordNet releases.

- Semantic Relations in WordNet

"A is XX of B"

Relation	Description	Example
Hypernym (nouns, verbs)	B is an A	Canine ~ dog
Hyponyms (noun, verbs)	A is an B	Dalmatian ~ dog
Holonyms (nouns)	B is part of A	Tree ~ trunk 나무줄기
Meronyms (nouns)	A is part of B	Bark 나무껍질 ~ trunk
Coordinates (nouns, verbs)	A and B have a common hypernym	Dalmatian ~ poodle (hypernym- dog)
Troponym (verbs)	Doing A is a manner of doing B	To march ~ to walk
Entailment (verbs)	Doing A implies also doing B	Snore 코를 골다 ~ Sleep
Related nouns (adjectives)	A was derived from B	Studious 신중한 ~ Study
Antonym (adjectives, adverbs)	A and B have opposite meanings	Beautiful ~ ugly
Similar to (adjectives)	A and B have similar meanings	Beautiful ~ lovely

- WordNet Online version
  - http://wordnetweb.princeton.edu/perl/webwn

WordNet Search - 3.1
- WordNet home page - Glossary - Help

Word to search for: car Search WordNet

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **car**, auto, automobile, machine, motorcar (a motor vehicle with four wheels; usually propelled by an internal combustion engine) *"he needs a car to get to work"*
- S: (n) **car**, railcar, railway car, railroad car (a wheeled vehicle adapted to the rails of railroad) *"three cars had jumped the rails"*
- S: (n) **car**, gondola (the compartment that is suspended from an airship and that carries personnel and the cargo and the power plant)
- S: (n) **car**, elevator car (where passengers ride up and down) *"the car was on the top floor"*
- S: (n) cable car, **car** (a conveyance for passengers or freight on a cable railway) *"they took a cable car to the top of the mountain"*

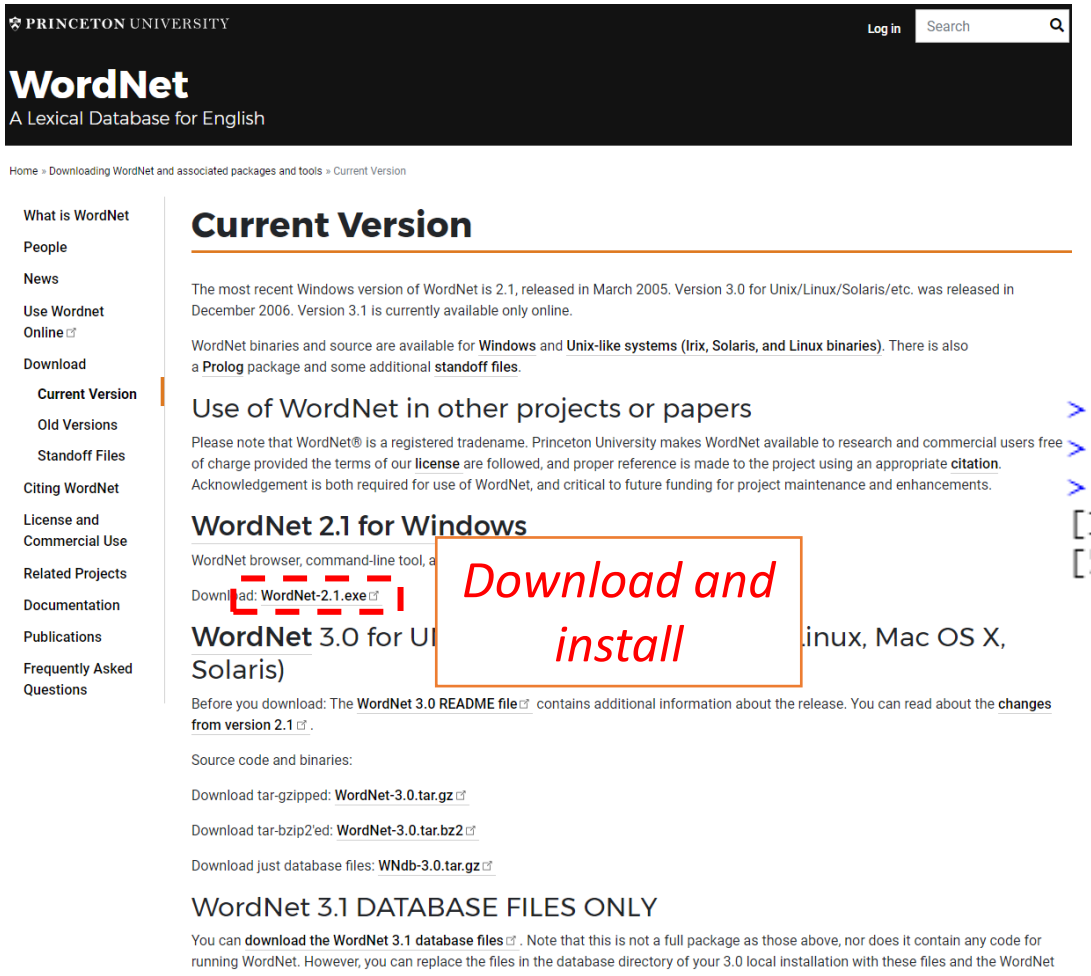
WordNet in R

https://wordnet.princeton.edu/download/current-version

- Three stages
  - Stage 1) download and install wordnet.exe
  - Stage 2) install.packages('wordnet')
  - Stage 3) load package

> library(wordnet)

> setDict("C:/Program Files (x86)/WordNet/2.1/dict")




```
> library(wordnet)
> setDict("C:/Program Files (x86)/WordNet/2.1/dict")
> getFilterTypes()
[1] "ContainsFilter"   "EndsWithFilter"   "ExactMatchFilter" "RegexFilter"
[5] "SoundFilter"      "StartsWithFilter" "WildcardFilter"
```

- Find synonyms

```
> word.filter <-
+   getTermFilter("ExactMatchFilter",
+                 "worship",
+                 ignoreCase=TRUE)
> word.filter
[1] "Java-Object{com.nexagis.jawbone.filter.ExactMatchFilter@433c675d}"
> word.terms <-
+   getIndexTerms("VERB",
+                 maxLimit = -1,
+                 word.filter)
> word.terms
[[1]]
[1] "Java-Object{Lemma: worship  POS: verb  Tag-Sense-Count: 3\nList of
 Synsets (3)\n  #1: 1761013\n  #2: 1761986\n  #3: 2588169\nList of Point
ers (4)\n  #1: @ (Hypernym)\n  #2: ~ (Hyponym)\n  #3: + (Derivationally
 related form)\n  #4: ; ([Unknown])}"

> word.terms[[1]] %>%
+   getSynonyms()
[1] "hero-worship" "idolise"      "idolize"      "revere"
[5] "worship"
```

"ContainsFilter", "EndsWithFilter", "ExactMatchFilter", "RegexFilter", "SoundFilter", "StartsWithFilter", "WildcardFilter"

"ADJECTIVE", "ADVERB", "NOUN", "VERB"

- Get synsets
  - Synsets: sets of cognitive synonyms

```
> word.synsets <- getSynsets(word.terms[[1]])
> sapply(
+   getRelatedSynsets(word.synsets[[1]],
+                     pointerSymbol="@"), getWord)
[1] "adore"
> sapply(
+   getRelatedSynsets(word.synsets[[2]],
+                     pointerSymbol="@"), getWord)
     [,1]
[1,] "reverence"
[2,] "fear"
[3,] "revere"
[4,] "venerate"
> sapply(
+   getRelatedSynsets(word.synsets[[3]],
+                     pointerSymbol="@"), getWord)
     [,1]
[1,] "attend"
[2,] "go to"
```

love unquestioningly and uncritically or to excess

show devotion to (a deity)

attend religious services

ptr_type	Value	Pointer Symbol	Search
ANTPTR	1	!	Antonyms
HYPERPTR	2	@	Hypernyms
HYPOPTR	3		Hyponyms
ENTAILPTR	4	*	Entailment
SIMPTR	5	&	Similar
ISMEMBERPTR	6	#m	Member meronym
ISSTUFFPTR	7	#s	Substance meronym
ISPARTPTR	8	#p	Part meronym
HASMEMBERPTR	9	%m	Member holonym
HASSTUFFPTR	10	%s	Substance holonym
HASPARTPTR	11	%p	Part holonym
MERONYM	12	%	All meronyms
HOLONYM	13	#	All holonyms
CAUSETO	14	>	Cause
PPLPTR	15	<	Participle of verb

https://wordnet.princeton.edu/documentation/wnsearch3wn

Lexical Resource: Part-of-Speech

Text as a Language

- Text
  - Computational symbols based on 0
  - a written form of language
  - English 한국어 Bahasa 中國語

- Text is a written form of language
  - As a form of language, text can present two features of language: meaning & function
  - Even if we used the same set of words, both features may be changed depending on how they are used.
  - When analysing text with the BoW approach, such changes cannot be considered.
  - Ex) “God loves you.” : (BoW) “god”, “love”, “you”
    “You love God.” : (BoW) “you”, “love”, “god”
    → the same words but with different meanings

Text as a Language

- Consideration of functional features of text
  - We can extract the actual meaning of the text
  - While the BoW approach only considers the frequency of text, this can be used as additional information for the text analysis
  - But, as expected, this requires a higher cost…

Part-of-Speech

POS	Definition	POS	Definition
CC	Coordinating conjunction	PRP$	Possessive pronoun (mine, his)
CD	Cardinal number	RB	Adverb (enough, not)
DT	Determiner	RBR	Comparative adverb (better)
EX	Existential there	RBS	Superlative adverb (best)
FW	Foreign word	RP	Particule
IN	preposition or subordinating conjunction	SYM	Symbol
JJ	Adjective	TO	To (to)
JJR	Comparative adjective	UH	Interjection (oh, uhm)
JJS	Superlative adjective	VB	Base form verb (love, eat)
LS	List item marker	VBD	Past tense verb (loved, ate)
MD	Modal	VBG	gerund or present particle (loving, eating)
NN	Singular noun	VBN	Past particle (loved, eaten)
NNS	Plural noun	VBP	Non-3rd person singular present (love, eat)
NNP	Proper noun	VBZ	3rd person singular present (loves, eats)
NNPs	Prural proper noun	WDT	Wh-determiner (which, that)
PDT	Predeterminer	WH	Wh-pronoun (what, who, whom)
POS	Possessive ending	WP$	Possessive wh-pronoun (whose, who)
PRP	Personal pronoun	WRB	Wh-adverb (how, where, why)

Part-of-Speech Analysis

- Part-of-Speech (PoS)
  - Often used in computational linguistics
  - PoS analysis, PoS tagging, PoS annotation

  - Identify the type of word
  - Technically, PoS is all about “adding” PoS information to our text data.

- NLP::annotate(*string*, *annotator*, *annotation object*)
  - Sentence object → word object → POS tag object

```
> sent.ant <-
+   annotate('God loves you. You love God.',
+            Maxent_Sent_Token_Annotator())
> sent.ant
 id type     start end features
  1 sentence     1  14
  2 sentence    16  28
```

```
> word.ant <-
+   annotate('God loves you. You love God.',
+            Maxent_Word_Token_Annotator(),
+            sent.ant)
> word.ant
 id type     start end features
  1 sentence     1  14 constituents=<<integer,4>>
  2 sentence    16  28 constituents=<<integer,4>>
  3 word         1   3
  4 word         5   9
  5 word        11  13
  6 word        14  14
  7 word        16  18
  8 word        20  23
  9 word        25  27
 10 word        28  28
```

```
> pos.ant <-
+   annotate('God hates you. You love God.',
+            Maxent_POS_Tag_Annotator(),
+            word.ant)
> pos.ant
 id type     start end features
  1 sentence     1  14 constituents=<<integer,4>>
  2 sentence    16  28 constituents=<<integer,4>>
  3 word         1   3 POS=NNP
  4 word         5   9 POS=VBZ
  5 word        11  13 POS=PRP
  6 word        14  14 POS=.
  7 word        16  18 POS=PRP
  8 word        20  23 POS=VBP
  9 word        25  27 POS=NNP
 10 word        28  28 POS=.
```

Proper noun

3rd person singular present

Personal pronoun

PoS Tagging

- Any recognizable differences?
  - We can consider the functional feature of text

```
> msg <- 'I love you. The love is all you need.'
> sent.1.ant <-
+   annotate(msg,
+            Maxent_Sent_Token_Annotator())
> word.1.ant <-
+   annotate(msg,
+            Maxent_Word_Token_Annotator(),
+            sent.1.ant)
> pos.1.ant <-
+   annotate(msg,
+            Maxent_POS_Tag_Annotator(),
+            word.1.ant)
```

```
> pos.1.ant
 id type     start end features
  1 sentence     1  11 constituents=<<integer,4>>
  2 sentence    13  37 constituents=<<integer,7>>
  3 word         1   1 POS=PRP
  4 word         3   6 POS=VBP
  5 word         8  10 POS=PRP
  6 word        11  11 POS=.
  7 word        13  15 POS=DT
  8 word        17  20 POS=NN
  9 word        22  23 POS=VBZ
 10 word        25  27 POS=DT
 11 word        29  31 POS=PRP
 12 word        33  36 POS=VBP
 13 word        37  37 POS=.
```

Love as verb

Love as noun

- udpipe::udpipe(*string*, object='*language*')

The treebank-specific parts of speech tag of the token

The universal parts of speech tag of the token

```
> udpipe(msg, object='english')
   doc_id paragraph_id sentence_id                    sentence start end term_id token_id token lemma  upos xpos
1    doc1            1           1                 I love you.     1   1       1        1     I     I  PRON  PRP
2    doc1            1           1                 I love you.     3   6       2        2  love  love  VERB  VBP
3    doc1            1           1                 I love you.     8  10       3        3   you   you  PRON  PRP
4    doc1            1           1                 I love you.    11  11       4        4     .     .  PUNCT    .
5    doc1            1           2 The love is all you need.    13  15       5        1   The   the   DET   DT
6    doc1            1           2 The love is all you need.    17  20       6        2  love  love  NOUN   NN
7    doc1            1           2 The love is all you need.    22  23       7        3    is    be   AUX  VBZ
8    doc1            1           2 The love is all you need.    25  27       8        4   all   all   DET   DT
9    doc1            1           2 The love is all you need.    29  31       9        5   you   you  PRON  PRP
10   doc1            1           2 The love is all you need.    33  36      10        6  need  need  VERB  VBP
11   doc1            1           2 The love is all you need.    37  37      11        7     .     .  PUNCT    .

                                                 feats head_token_id   dep_rel deps            misc
                       Case=Nom|Number=Sing|Person=1|PronType=Prs             2      nsubj <NA>            <NA>
                                    Mood=Ind|Tense=Pres|VerbForm=Fin         0       root <NA>            <NA>
                                      Case=Acc|Person=2|PronType=Prs         2        obj <NA>   SpaceAfter=No
                                                                <NA>         2      punct <NA>            <NA>
                                         Definite=Def|PronType=Art         2        det <NA>            <NA>
                                                        Number=Sing         4      nsubj <NA>            <NA>
            Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin         4        cop <NA>            <NA>
                                                                <NA>         0       root <NA>            <NA>
                                      Case=Nom|Person=2|PronType=Prs         6      nsubj <NA>            <NA>
                                    Mood=Ind|Tense=Pres|VerbForm=Fin         4 acl:relcl <NA>   SpaceAfter=No
                                                                <NA>         4      punct <NA> SpacesAfter=\\n
```

- udpipe::udpipe(*string*, object='*language*')
  - Can easily compare the same words with the different functions

```
> msg.pos <-
+   udpipe(msg, object='english')
> msg.pos %>% filter(token=="love") %>%
+   select(doc_id, sentence, token, upos, xpos, feats)
  doc_id                    sentence token upos xpos                          feats
1   doc1                 I love you.  love VERB  VBP Mood=Ind|Tense=Pres|VerbForm=Fin
2   doc1 The love is all you need.  love NOUN   NN                      Number=Sing
```

Recommendations for PoS Analysis

- Should PoS always be prioritized over BoW?
  - Compared to the BoW approach, the PoS analysis can provide better results compared to the BoW approach because it can provide information. So should we?
  - Imagine you want to buy a new laptop… There are only a few people who can buy the best performing product…
  - Cost matters…

- Cost benefit analysis (CBA)
  - The process used to measure the benefits of a decision or taking action minus the costs associated with taking that action [Investopedia]
  - Conduct PoS analysis only if [Benefit of PoS > cost of PoS]