8. Text Mining

Statistical Modelling & Machine Learning

Jaejik Kim

Department of Statistics
Sungkyunkwan University

STA3036

Introduction

- Text mining: Tools to analyze a large size of text from human language.
- Goal of text mining: Extracting useful information from text as human learns from reading text.
- Examples of text mining:
  - Trend analysis for news articles.
  - Identification of real-time topics emerging on Twitter.
  - Classification of papers with similar topics, etc.

Terminology

- Corpus: A large and unstructured set of texts which are usually electronically stored and processed.
- Document: Article, story book chapter, papers, etc.
- Corpus includes several separate documents.
- Each document is considered as individual entities or records.
- Document term matrix:
  - A matrix representing frequencies of words in each document.
  - Generally, it is a large and sparse matrix (sparse matrix: Most of elements in the matrix are zero).
  - Row: Document; column: word.

Basic Analysis Process

1. Collection of texts (R: Plain text, PDF, MS Word, XML files).
2. Pre-processing: Transformation of documents.
   - Remove special characters, numbers, punctuation, white space, stop words (e.g., i, me, my, he, she,etc.).
   - convert to lower case.
   - Stemming: Remove common word endings (e.g., 's', 'es', 'ed', etc.).
   - Specific transformation (e,g., kor $\rightarrow$ korea).
   - Remove user defined words (unmeaningful words defined by user).
3. Computing the document term matrix.
4. Various analyses using the document term matrix (frequencies, correlation, word cloud, clustering, topic search, etc.).

R code: Preparation

```
> install.packages('quanteda'); install.packages('readtext')
> install.packages('tidyverse'); install.packages('tidytext')
> library(quanteda); library(readtext)
> library(tidyverse); library(tidytext)
>
> # Bar plot function
> facet_bar <- function(df, y, x, by, nrow = 2, ncol = 2,
+ scales = "free") {
+   mapping <- aes(y = reorder_within({{ y }}, {{ x }},
+                  {{ by }}), x={{ x }}, fill={{ by }})
+
+   facet <- facet_wrap(vars({{ by }}),
+            nrow = nrow, ncol = ncol, scales = scales)
+
+   ggplot(df, mapping = mapping) +
+     geom_col(show.legend = FALSE) +
+     scale_y_reordered() + facet + ylab("")}
```

R code: Read external text files

```
> dat <- readtext("txt/*")
> dat
readtext object consisting of 9 documents and 0 docvars.
# Description: df[,2] [9 x 2]
  doc_id   text
  <chr>    <chr>
1 doc1.txt "\"iPhone 13 \"..."
2 doc2.txt "\"As a priva\"..."
3 doc3.txt "\"iOS 14 rei\"..."
4 doc4.txt "\"iOS is alr\"..."
5 doc5.txt "\"It's 2021 \"..."
6 doc6.txt "\"Apple's iP\"..."
# ... with 3 more rows
```

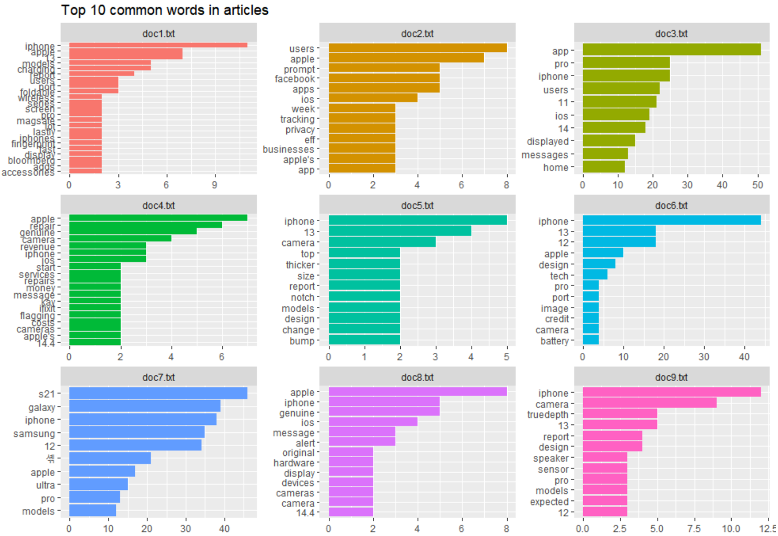
R code: Select top 10 words for each document

```
> doc_common <- dat %>%
+   unnest_tokens(word, text) %>%
+   anti_join(stop_words) %>%
+   count(doc_id, word) %>%
+   group_by(doc_id) %>%
+   top_n(10) %>%
+   ungroup()
Joining, by = "word"
Selecting by n
> # unnest_tokens: split input(text) into tokens(word).
> # anti_join: Filtering.
> # count: count the unique values of one or more variables.
> # group_by: convert an existing table into a grouped table.
> # top_n: select n highest values.
> # ungroup: remove grouping.
```

R code: Bar chart of top 10 words

```
> facet_bar(doc_common,
+           y = word,
+           x = n,
+           by = doc_id,
+           nrow = 3, ncol = 3)+
+   labs(title = "Top 10 common words in articles",x = "")
```

R code: Bar chart of top 10 words



Sentiment Analysis

- Sentiment analysis: A natural language process (NLP) detecting sentiments such as positive and negative.
- Examples of sentiment analysis:
  - Political sentiment analysis: Measuring the positive or negative level for a policy or party from texts such as twitter.
  - Brand reputation: Analysis for brand likeness through texts.
  - Customer feedback: Customer review text analysis.
- Lexicon: Dictionary that defines sentiment for words (rule-based approach).

Sentiment Analysis

- Types of lexicons:
  - `nrc`: A list of English words and their associations with eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive).
  - `bing`: Two sentiments (positive and negative).
  - `AFINN`: It gives a value between -5 and 5 to each word (-5: Strong negative setiment; 5: Strong positive sentiment).

R code: Sentiment Analysis

```
> # bing lexicon with positive words
> bing_pos <- get_sentiments('bing') %>%
+   filter(sentiment == 'positive')
> bing_pos
# A tibble: 2,005 x 2
   word         sentiment
   <chr>        <chr>
 1 abound       positive
 2 abounds      positive
 3 abundance    positive
 4 abundant     positive
 5 accessable   positive
 6 accessible   positive
 7 acclaim      positive
# ... with 1,995 more rows
```

R code: Sentiment Analysis

```
> # Tokenizing: unigram.
> doc <- dat %>%
+   ungroup() %>%
+   unnest_tokens(word, text)
> doc
readtext object consisting of 6316 documents and 0 docvars.
# Description: df[,3] [6,316 x 3]
  doc_id   word   text
  <chr>    <chr>  <chr>
1 doc1.txt iphone "\"\"..."
2 doc1.txt 13     "\"\"..."
3 doc1.txt series "\"\"..."
4 doc1.txt launch "\"\"..."
5 doc1.txt is     "\"\"..."
# ... with 6,310 more rows
```

R code: Sentiment Analysis

```
> # List and count poistive words in doc2.txt.
> doc %>%
+   filter(doc_id == 'doc2.txt') %>%
+   inner_join(bing_pos) %>%
+   count(word, sort = TRUE)
Joining, by = "word"
readtext object consisting of 7 documents and 0 docvars.
# Description: df[,3] [7 x 3]
  word              n text
  <chr>         <int> <chr>
1 prompt            5 "\"\"..."
2 personalized      2 "\"\"..."
3 right             2 "\"\"..."
4 better            1 "\"\"..."
# ... with 3 more rows
```

R code: Sentiment Analysis

```
> # Positive and negative word frequencies for each doc.
> doc_sentiment <- doc %>%
+   inner_join(get_sentiments('bing')) %>%
+   count(doc_id, sentiment) %>%
+   pivot_wider(names_from = sentiment, values_from = n,
+               values_fill = list(n = 0)) %>%
+   mutate(pos_diff = positive - negative)
Joining, by = "word"
> doc_sentiment
# A tibble: 9 x 4
  doc_id   negative positive pos_diff
  <chr>       <int>    <int>    <int>
1 doc1.txt        5        6        1
2 doc2.txt        8       13        5
# ...
```

R code: Word Cloud

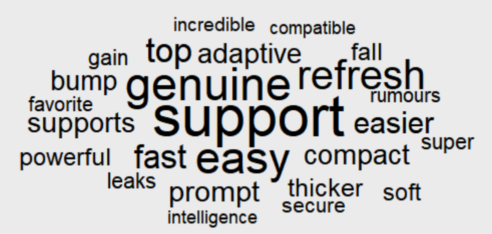
```
> # Word Cloud
> install.packages('ggwordcloud')
> library(ggwordcloud)
>
> # Word cloud for words with high frequencies.
> wc <- doc %>%
+   anti_join(stop_words) %>%
+   inner_join(get_sentiments('bing')) %>%
+   count(sentiment, word, sort = T) %>%
+   top_n(20)
Joining, by = "word"
Joining, by = "word"
Selecting by n
```

R code: Word Cloud

```
> print(as_tibble(wc), n = 9)
# A tibble: 25 x 3
   sentiment word              n
   <chr>     <chr>         <int>
 1 positive  support          13
 2 positive  genuine          10
 3 positive  easy              8
 4 positive  refresh           8
 5 positive  adaptive          5
 6 positive  compact           5
 7 positive  easier            5
 8 positive  fast              5
 9 positive  prompt            5
```

R code: Word Cloud

```
> wc %>%
+   ggplot() +
+   geom_text_wordcloud_area(aes(label = word, size = n)) +
+   scale_size_area(max_size = 15)
```



TF-IDF

TF-IDF (Term Frequency - Inverse Document Frequency):

- Words frequently appeared in documents cannot be important.
- Words that distinguish documents can be regarded as important words.
- TF: Proportion of a certain word for all words in a document.

$$TF(w_{ij}, d_i) = \log(F(w_{ij}, d_i) + 1),$$

  where w_{ij} is the jth word in the ith document, d_i is the ith document in corpus D, and $F(w_{ij}, d_i)$ is the frequency of w_{ij} in d_i.

TF-IDF

TF-IDF (Term Frequency - Inverse Document Frequency):

- IDF: Inverse proportion of the number of documents where a certain word appears for the total number of documents in the corpus (a word in few of documents has high IDF value).

$$IDF(w_{ij}, D) = \log \frac{|D|}{1 + |\{d_i \in D : w_{ij} \in d_i\}|},$$

  where $|\cdot|$ is the number of documents.

- TF-IDF:

$$TF - IDF(w_{ij}, d_i, D) = TF(w_{ij}, d_i) \cdot IDF(w_{ij}, D).$$

R code: TF-IDF

```
> doc_words <- dat %>%
+   unnest_tokens(word, text) %>%
+   add_count(doc_id, name = "total_words") %>%
+   group_by(doc_id, total_words) %>%
+   count(word, sort = TRUE) %>%
+   ungroup()
> doc_words
# A tibble: 2,517 x 4
   doc_id   total_words word      n
   <chr>          <int> <chr> <int>
 1 doc7.txt        1834 the     113
 2 doc3.txt        1867 and      78
 3 doc3.txt        1867 the      68
 4 doc3.txt        1867 to       63
# ... with 2,513 more rows
```

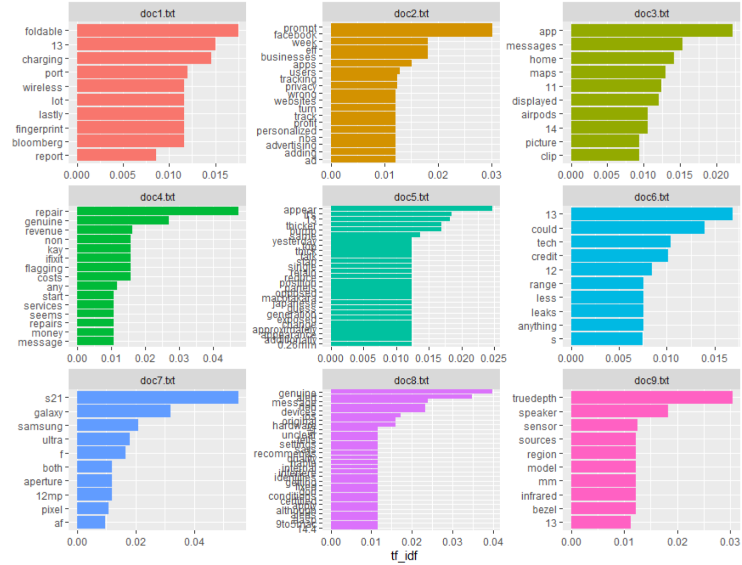
R code: TF-IDF

```
> # Calculate TF-IDF.
> doc_words <- doc_words %>%
+   select(-total_words) %>%
+   bind_tf_idf(term = word, document = doc_id, n = n)
> doc_words
# A tibble: 2,517 x 6
   doc_id   word       n     tf   idf tf_idf
   <chr>    <chr> <int>  <dbl> <dbl>  <dbl>
 1 doc7.txt the     113 0.0616 0      0
 2 doc3.txt and      78 0.0418 0      0
 3 doc3.txt the      68 0.0364 0      0
 4 doc3.txt to       63 0.0337 0      0
 5 doc6.txt the      57 0.0657 0      0
 6 doc7.txt a        57 0.0311 0      0
# ... with 2,511 more rows
```

R code: TF-IDF

```
# Plot for 10 words with highest TF-IDF values.
doc_words %>%
  group_by(doc_id) %>%
  top_n(10) %>%
  ungroup() %>%
  facet_bar(y = word,
            x = tf_idf,
            by = doc_id,
            nrow = 3, ncol = 3)
```

R code: Top 10 TF-IDF Words



R code: Tokenizing by n gram

```
> doc_bigrams <- dat %>%
+   unnest_tokens(bigram, text, token = "ngrams", n = 2)
> doc_bigrams %>%
+   count(bigram, sort = TRUE)
readtext object consisting of 4556 documents and 0 docvars.
# Description: df[,3] [4,556 x 3]
  bigram          n text
  <chr>       <int> <chr>
1 iphone 12      55 "\"\"..."
2 the iphone     53 "\"\"..."
3 iphone 13      34 "\"\"..."
4 galaxy s21     32 "\"\"..."
5 in the         32 "\"\"..."
6 of the         27 "\"\"..."
# ... with 4,550 more rows
```

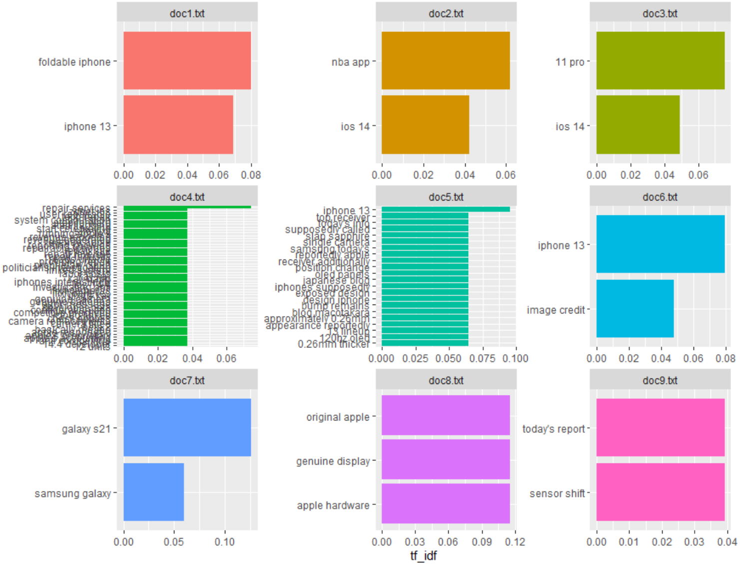
R code: Remove bigrams with stop words

```
> # If there is a stop word in a bigram token, remove it.
> doc_separated <- doc_bigrams %>%
+   separate(bigram, into = c("word1", "word2"), sep = " ")
> doc_united <- doc_separated %>%
+   filter(!word1 %in% stop_words$word,
+          !word2 %in% stop_words$word) %>%
+   unite(bigram, c(word1, word2), sep = " ")
> doc_united %>% count(bigram, sort = TRUE)
readtext object consisting of 1155 documents and 0 docvars.
# Description: df[,3] [1,155 x 3]
  bigram              n text
  <chr>           <int> <chr>
1 iphone 12          55 "\"\"..."
2 iphone 13          34 "\"\"..."
# ... with 1,153 more rows
```

R code: Bar chart for top 2 TF-IDF bigrams

```
> # Plot for top 2 bigram words
> doc_united %>%
+   count(doc_id, bigram, sort = TRUE) %>%
+   bind_tf_idf(term = bigram, document = doc_id, n = n) %>%
+   group_by(doc_id) %>%
+   top_n(2) %>%
+   ungroup() %>%
+   facet_bar(y = bigram, x=tf_idf, by=doc_id, nrow=3, ncol=3)
Selecting by tf_idf
```

R code: Bar chart for top 2 TF-IDF bigrams



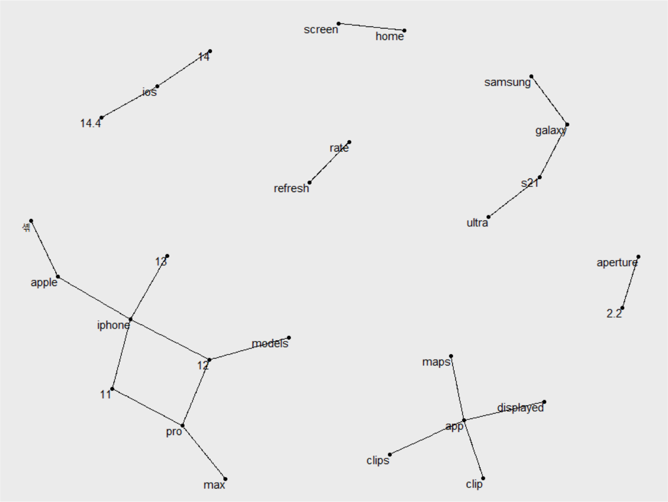
R code: Network of bigrams

```
> install.packages('tidygraph'); install.packages('ggraph')
> library(tidygraph); library(ggraph)
>
> bigram_counts <- doc_separated %>%
+   filter(!word1 %in% stop_words$word,
+          !word2 %in% stop_words$word) %>%
+   count(word1, word2, sort = TRUE)
> bigram_counts
readtext object consisting of 1155 documents and 1 docvar.
# Description: df[,4] [1,155 x 4]
  word1   word2      n text
  <chr>   <chr>  <int> <chr>
1 iphone  12        55 "\"\"..."
2 iphone  13        34 "\"\"..."
# ... with 1,153 more rows
```

R code: Network of bigrams

```
> bigram_graph <- bigram_counts %>%
+   filter(n > 5) %>%
+   as_tbl_graph()
>
> ggraph(bigram_graph, layout = "fr") +
+   geom_edge_link() +
+   geom_node_point() +
+   geom_node_text(aes(label = name), vjust = 1, hjust = 1)
```

R code: Network of bigrams



R code: Document Term Matrix

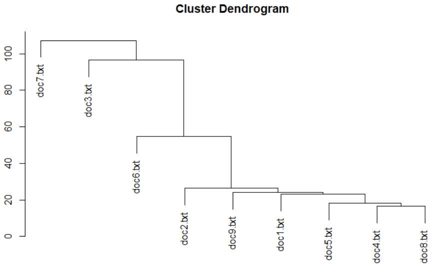
```
> cop <- corpus(dat)   # Construct a corpus
> # Document term matrix (Document feature matrix)
> dfmat <- cop %>%
+   tokens(remove_punct=T, remove_symbols=T) %>%
+   tokens_select(pattern = stopwords("en"),
+                 selection = "remove") %>% dfm()
> dfmat
Document-feature matrix of: 9 documents,
1,508 features (84.7% sparse).
          features
docs        iphone 13 series launch still months away can now
  doc1.txt      11  7      2      1     1      1    1   2   1
  doc2.txt       1  0      0      0     1      0    0   0   0
[ reached max_ndoc ... 7 more documents, reached max_nfeat
... 1,499 more features ]
```

R code: TF-IDF Matrix

```
> tf_idfmat <- dfm_tfidf(dfmat)
> tf_idfmat
Document-feature matrix of: 9 documents,
1,508 features (84.7% sparse).
          features
docs         iphone        13   series    launch     still
  doc1.txt        0 2.465278 1.306425 0.9542425 0.4771213
  doc2.txt        0 0        0        0         0.4771213
  doc3.txt        0 0        0        0         0
  doc4.txt        0 0        0        0         0
  doc5.txt        0 1.408730 0        0         0
  doc6.txt        0 6.339285 0        0         0
[ reached max_ndoc ... 3 more documents, reached max_nfeat
 ... 1,503 more features ]
```

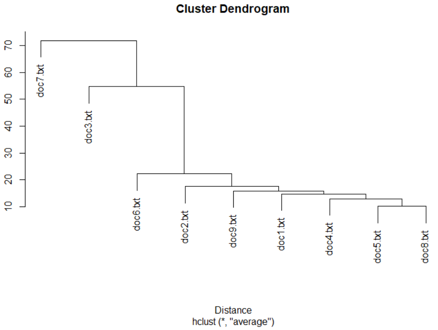
R code: Clustering documents using DTM

```
> install.packages('quanteda.textstats')
> library(quanteda.textstats)
>
> # Clustering documents using dtm
> text_dist <- as.dist(textstat_dist(dfmat))
> clust <- hclust(text_dist, method = 'average')
> plot(clust, xlab = "Distance", ylab = "")
```



R code: Clustering documents using TF-IDF

```
> # Clustering documents using tf-idf
> text_dist <- as.dist(textstat_dist(tf_idfmat))
> clust <- hclust(text_dist, method = 'average')
> plot(clust, xlab = "Distance", ylab = "")
```



Topic model

- A machine learning technique that automatically analyzes text data to determine cluster words for a set of documents.
- Unsupervised learning problem.
- An example of topic model:
  - Identification of the topics of a set of customer reviews by detecting patterns and recurring words.
- Latent Dirichlet Allocation (LDA):
  - One of the representative topic models.
  - Statistical model to detect hidden meaning structure in documents.

Assumptions of LDA

- Document: A random mixture of latent topics.
  - E.g., Two-topic model.
  - Document1 consists of 80% of topic1 and 20% of topic2.
  - Document2 consists of 40% of topic1 and 60% of topic2.

- Topic: A random mixture of words.
  - E.g., news articles with topics ‘politics’ and ‘media’.
  - ‘Politics’ topic has words ‘president’, ‘congress’, ‘government’, etc.
  - ‘Media’ topic has words ‘television’, ‘radio’, ‘news’, etc.
  - Each word in a topic probabilistically appears in a document.

Generative process for documents in LDA

- LDA assumes the following generative process for each document d in a corpus D.
  1. The # of words in a document $N \sim Poisson(\xi)$.
  2. Parameter of distribution of topics for d: $\boldsymbol{\theta} \sim Dirichlet(\boldsymbol{\alpha})$.
  3. For each of the N words w_n,
     4-1. A topic $z_n \sim Multinomial(\boldsymbol{\theta})$.
     4-2. A word $w_n \sim p(w_n|z_n, \boldsymbol{\beta})$, a multinomial probability conditioned on the topic z_n.

  - The number of topics, K, is assumed known and fixed (The dimension of $\boldsymbol{\theta}$).

  - Word probabilities are parameterized by a $K \times V$ matrix $\boldsymbol{\beta}$.
    - V: the number of vocabularies.
    - Element of $\boldsymbol{\beta}$: $\beta_{ij} = p(w^j|z^i) \Rightarrow$ Prob. that the word w^j being generated from the topic z^i.

LDA

- Word-topic probabilities: $\beta_{ij} = p(w^j | z^i)$.
- Document-topic probabilities: $\boldsymbol{\theta} \sim Dirichlet(\boldsymbol{\alpha})$.

$$p(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \frac{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \cdots \theta_K^{\alpha_K - 1}, \ \theta_i \geq 0,$$

  - $\alpha_i > 0$, $\sum_{i=1}^{K} \theta_i = 1$.
- Joint distribution of a topic mixture $\boldsymbol{\theta}$:

$$p(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{w} | \boldsymbol{\alpha}, \boldsymbol{\beta}) = p(\boldsymbol{\theta}|\boldsymbol{\alpha}) \prod_{n=1}^{N} p(z_n|\boldsymbol{\theta}) p(w_n | z_n, \boldsymbol{\beta}).$$

- Estimation of $(\boldsymbol{\alpha}, \boldsymbol{\beta})$: Gibbs sampling, Variational EM, etc.

R code: Preprocessing

```
# Preprocessing
>
> library(tm)
> library(topicmodels)
>
> docs <- Corpus(DirSource('./txt/'))
>
> docs <- docs %>%
+     tm_map(content_transformer(tolower)) %>%
+     tm_map(removeNumbers) %>%
+     tm_map(removePunctuation) %>%
+     tm_map(removeWords, stopwords("english")) %>%
+     tm_map(stripWhitespace) %>%
+     tm_map(stemDocument)
```

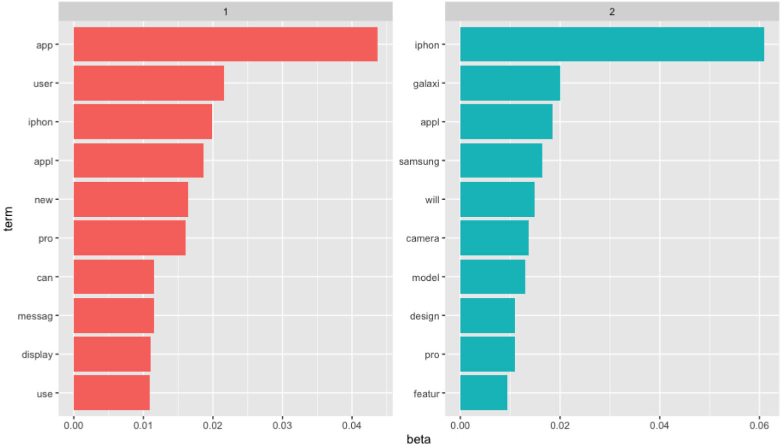
R code: LDA & Word-topic prob.

```
> # Document term matrix.
> dtm = DocumentTermMatrix(docs)
>
> # LDA
> doc_lda <- LDA(dtm, k = 2, control = list(seed = 1234))
>
> # Word-topic probability
> doc_topics <- tidy(doc_lda, matrix = "beta")
> doc_topics
> doc_topics
# A tibble: 2,260 × 3
   topic term          beta
   <int> <chr>        <dbl>
 1     1 accessori 1.28e- 3
 2     2 accessori 2.00e- 3
 3     1 add       2.57e- 3
 4     2 add       9.99e- 4
# ... with 2,256 more rows
```

R code: Top ten words by topic

```
> # Top ten words by topic
> library(ggplot2)
> library(dplyr)
> top_terms <- doc_topics %>%
+   group_by(topic) %>%
+   slice_max(beta, n = 10) %>%
+   ungroup() %>%
+   arrange(topic, -beta)
>
> top_terms %>%
+   mutate(term = reorder_within(term, beta, topic)) %>%
+   ggplot(aes(beta, term, fill = factor(topic))) +
+   geom_col(show.legend = FALSE) +
+   facet_wrap(~ topic, scales = "free") +
+   scale_y_reordered()
```

R code: Top ten words by topic



R code: Document-topic prob.

```
> # Document-topic probability
> doc_documents <- tidy(doc_lda, matrix = "gamma")
> doc_documents
# A tibble: 18 × 3
   document topic     gamma
   <chr>    <int>     <dbl>
 1 doc1.txt     1 0.000228
 2 doc2.txt     1 1.00
 3 doc3.txt     1 1.00
 ..........
 9 doc9.txt     1 0.000223
10 doc1.txt     2 1.00
11 doc2.txt     2 0.000238
12 doc3.txt     2 0.0000436
 ..........
18 doc9.txt     2 1.00
```

R code: High freq. words for each document

```
> tidy(dtm) %>%
+   filter(document == 'doc6.txt') %>%
+   arrange(desc(count))
# A tibble: 259 × 3
   document term    count
   <chr>    <chr>   <dbl>
 1 doc6.txt iphon      46
 2 doc6.txt appl       12
 3 doc6.txt design      8
 4 doc6.txt batteri     6
 5 doc6.txt new         6
 6 doc6.txt tech        6
 7 doc6.txt best        5
 8 doc6.txt featur      5
# ... with 251 more rows
> ## Topic 1: Hardware of iphone
```

R code: High freq. words for each document

```
> # Top terms for each document
> tidy(dtm) %>%
+   filter(document == 'doc3.txt') %>%
+   arrange(desc(count))
# A tibble: 439 × 3
   document term    count
   <chr>    <chr>   <dbl>
 1 doc3.txt app        59
 2 doc3.txt iphon      25
 3 doc3.txt pro        25
 4 doc3.txt new        24
 5 doc3.txt user       23
 6 doc3.txt can        16
 7 doc3.txt clip       15
 8 doc3.txt display    15
# ... with 431 more rows
> ## Topic 2: Apps in iphone
```