Statistical Language Models

Week 7.2

Textmining in R, chapter 8

Perplexity

- Training dataset used to build LDA model
- Validation dataset used to asses the (model selection) number of ~~clusters~~ topics

From hold out data

- Great resource: Section 2 of
- vignette("topicmodels")
- Geometric mean per word likelihood:
- $Perplexity(Doc_{test}) = exp\left\{ -\frac{\sum_{doc} log(p(w \mid \theta, \phi))}{\sum_{doc} \{\text{\#words is in doc}\}} \right\}$
- lower perplexity score indicates better generalization performance

Perplexity

- Measure is how likely new data is given the LDA model that was learned.

- Assumptions: standard to training /testing

- Perplexity is not well correlated with human perception of quality by interpretability of topics

JavaScript Object Notation (JSON) formats

- JSON data comes in key/value pairs
- Think of it as named data frame columns or named vectors.
- Our course schedule might be have several key/value pairs:

"startTime":"13:05", “days”:"Tu/Th",

“endTime":"14:25",

“roomNumber":"3224","building":"Richcraft Hall"

JavaScript Object Notation (JSON) formats

- Data can be
  - Numbers: 1, 2, 3.1415
  - Strings: “Text in double quotes”
  - Boolean: TRUE
  - Array [“ordered”, “comma separated”, “enclosed in square brackets”,”any data type inside”]
- Object {unordered, comma separated, collection of key: value pairs in curly brackets, any data types}

JavaScript Object Notation (JSON) formats

- Our course schedule might be have several “key”:value pairs:

"startTime":"13:05", “days”:"Tue/Thu",

"endTime":"2:25",

“roomNumber":"3224","building":"Richcraft Hall"

JavaScript Object Notation (JSON) formats

- Our course info could be split into a hierarchical data structure with the top levels:
- CourseSchedule
- Course Instructor
- Grading
- Course Info
- Text

JavaScript Object Notation (JSON) formats

- Our course info could be split into a hierarchical data structure with the top levels:
- CourseSchedule [start time, end time, day, room, …]
- Course Instructor [name, office, email, phone,…]
- Grading [assignments, midterms, final,…]
- Course Info [pre-req, delivery method, title,description,…
- Text [required, recommended,…]

Some course info in JSON:

“courseSchedule”:[{"startTime":"13:05",
"startDate":"Tue Jan 07 2020", "roomNumber":"7618",
"days":"Mo","endDate":"Thu Apr 07
2020”,"endTime":"14:20", **"isExam":false**,
“roomNumber":"3224","building":"Richcraft Hall"}],

Big JSON file

- Can we find datasets that are related? Are there clusters of similar datasets? What does NASA research?

- https://data.nasa.gov/data.json

Metadata from NASA datasets from their research

- library(jsonlite)
- metadata = fromJSON("https://data.nasa.gov/data.json") # big file
- class(metadata)
- names(metadata$dataset)
-

- class(metadata$dataset$title)
- length(metadata$dataset$title)
- class(metadata$dataset$keyword)
- class(metadata$dataset$description)

Tidying up the title data

- #Set up separate nibbles for title, description, and keyword
- library(dplyr)
- nasa_title = tibble(id = metadata$dataset$`identifier`,
-                                 title = metadata$dataset$title)
- nasa_title
-

Data set descriptions

- nasa_desc = tibble(id = metadata$dataset$`identifier`,
-                             desc = metadata$dataset$description)

- nasa_desc %>%
-   select(desc) %>%
-   sample_n(5)
-

The data keywords (one row for each keyword)

- library(tidyr)

- nasa_keyword = tibble(id = metadata$dataset$`identifier`,
-                               keyword = metadata$dataset$keyword) %>%
-   unnest(keyword)

- nasa_keyword

Tokenize title and description

- library(tidytext)
- nasa_title = nasa_title %>%
-   unnest_tokens(word, title) %>%
-   anti_join(stop_words)
- nasa_title
- ############# stop_words?????
  #########

- nasa_desc = nasa_desc %>%
-   unnest_tokens(word, desc)
  %>%
-   anti_join(stop_words)
- nasa_desc

Basic Explorations

- # Find the most common words in the dataset
- nasa_title %>%
-   count(word, sort = TRUE)
- nasa_desc %>%
-   count(word, sort = TRUE)

Custom stop words #arguably include words like "data"

- my_stopwords = tibble(word = c(as.character(1:10),
- "v1", "v1.0","v03", "l2", "l3", "l4", "v5.2.0",
- "v003", "v004", "v005", "v006", "v7"))
- nasa_title = nasa_title %>%
- anti_join(my_stopwords)
- nasa_desc = nasa_desc %>%
- anti_join(my_stopwords)

Common keywords

- nasa_keyword %>%
- group_by(keyword) %>%
- count(sort = TRUE)

Networks of Description and Title Words

- #Count the number of pairwise occurrences of words in a title or description
- library(widyr)
- title_word_pairs = nasa_title %>%
-   pairwise_count(word, id, sort = TRUE, upper = FALSE)
- title_word_pairs

Common co-occuring description words

- desc_word_pairs = nasa_desc %>%
-    pairwise_count(word, id, sort = TRUE, upper = FALSE)
- desc_word_pairs

Networks of co-occuring words in titles

- library(ggplot2)
- library(igraph)
- library(ggraph)

- title_word_pairs %>%
- filter(n >= 250) %>%
- graph_from_data_frame() %>%

- **ggraph**(layout = "fr") +
- geom_edge_link(aes(edge_alpha = n, edge_width = n), edge_colour = "cyan4") +
- geom_node_point(size = 5) +
- geom_node_text(aes(label = name), repel = TRUE,
- point.padding = unit(0.2, "lines")) +
- theme_void()

Networks of Co-occurences in descritions

- desc_word_pairs %>%
- filter(n >= 1000) %>%
- graph_from_data_frame() %>%

- ggraph(layout = "fr") +
- geom_edge_link(aes(edge_alpha = n, edge_width = n), edge_colour = "darkred") +
- geom_node_point(size = 5) +
- geom_node_text(aes(label = name), repel = TRUE,
- point.padding = unit(0.2, "lines")) +
- theme_void()

Networks of co-occurences in keywords

- keyword_pairs = nasa_keyword %>%
- pairwise_count(keyword, id, sort = TRUE, upper = FALSE)
- keyword_pairs

- keyword_pairs %>%
- filter(n >= 500) %>%
- graph_from_data_frame() %>%
- ggraph(layout = "fr") +
- geom_edge_link(aes(edge_alpha = n, edge_width = n), edge_colour = "royalblue") +
- geom_node_point(size = 5) +
- geom_node_text(aes(label = name), repel = TRUE,
- point.padding = unit(0.2, "lines")) +
- theme_void()

- #Note the clusters; these seem to define redundancy, but counts won’t tell us for sure. Co-occurrence isn’t always enough
- keyword_cors = nasa_keyword %>%
-   group_by(keyword) %>%
-   filter(n() >= 50) %>%
-   pairwise_cor(keyword, id, sort = TRUE, upper = FALSE)
- keyword_cors

Correlation options

- Pearson Correlation is classic linear relationship for two continuous variables
- Kendall rank correlation measures the strength of dependence for ordered data
- $\tau = \frac{N_{concordant} - N_{discordant}}{.5N(N-1)} = \frac{1}{.5N(N-1)} \sum_{i<j} sgn(x_i - x_j) sgn(y_i - y_j)$
- Concordant = consistent
- Spearman rank correlation for ordinal or continuous variables
- $\rho = 1 - \frac{6 \sum d_i^2}{N(N^2-1)}$, for difference between ranks d_i and N observations

Visualize networks of words via correlations

- keyword_cors %>%
- filter(correlation > .6) %>%
- graph_from_data_frame() %>%
- ggraph(layout = "fr") +
- geom_edge_link(aes(edge_alpha = correlation, edge_width = correlation), edge_colour = "royalblue") +
- geom_node_point(size = 5) +
- geom_node_text(aes(label = name), repel = TRUE,
- point.padding = unit(0.2, "lines")) +
- theme_void()

tf_idf from descriptions

- desc_tf_idf = nasa_desc %>%
-   count(id, word, sort = TRUE) %>%
-   ungroup() %>%
-   bind_tf_idf(word, id, n)

- desc_tf_idf = full_join(desc_tf_idf, nasa_keyword, by = "id")
- desc_tf_idf %>%
-   arrange(-tf_idf)
- #many boring words

Look for high tf_idf terms within keywords

- desc_tf_idf %>%
- filter(!near(tf, 1)) %>%
- filter(keyword %in% c("solar activity", "clouds" , "seismology" ,"astrophysics", "human health" , "budget" ,"climate"   )) %>%
- arrange(desc(tf_idf)) %>%
- group_by(keyword) %>%
- distinct(word, keyword, .keep_all = TRUE) %>%
- top_n(15, tf_idf) %>%
- ungroup() %>%
- mutate(word = factor(word, levels = rev(unique(word)))) %>%
- ggplot(aes(word, tf_idf, fill = keyword)) +
- geom_col(show.legend = FALSE) +
- facet_wrap(~keyword, ncol = 3, scales = "free") +
- coord_flip() +
- labs(title = "Highest tf-idf words in NASA metadata description fields",
- caption = "NASA metadata from https://data.nasa.gov/data.json",
- x = NULL, y = "tf-idf")

LDA

- my_stop_words = bind_rows(stop_words, tibble(word = c("nbsp", "amp", "gt", "lt", "timesnewromanpsmt", "font", "td", "li", "br", "tr", "quot", "st", "img", "src", "strong", "http", "file", "files", as.character(1:12)), lexicon = rep("custom", 30)))
- word_counts = nasa_desc %>%
- anti_join(my_stop_words) %>%
- count(id, word, sort = TRUE) %>%
- ungroup()
- word_counts

To DTM

- desc_dtm = word_counts %>%
-    cast_dtm(id, word, n)

- desc_dtm
- ### desc_dtm = removeSparseTerms(desc_dtm,.95)

LDA

- library(topicmodels)

- # be aware that running this model is time intensive
- desc_lda <- LDA(desc_dtm, k = 24, control = list(seed = 1234))
- desc_lda

- tidy_lda <- tidy(desc_lda)

- tidy_lda

Their β = probability of a term given a topic

- top_terms <- tidy_lda %>%
-   group_by(topic) %>%
-   top_n(10, beta) %>%
-   ungroup() %>%
-   arrange(topic, -beta)

- top_terms

In plots

- top_terms %>%
- mutate(term = reorder_within(term, beta, topic)) %>%
- group_by(topic, term) %>%
- arrange(desc(beta)) %>%
- ungroup() %>%
-
-
- ggplot(aes(term, beta, fill = as.factor(topic))) +
- geom_col(show.legend = FALSE) +
- coord_flip() +
- scale_x_reordered() +
- labs(title = "Top 10 terms in each NASA topic",
- x = NULL, y = expression(beta)) +
- facet_wrap(~ topic, ncol = 4, scales = "free")

Their γ = probability of topic within document Connecting topic modeling with keywords

- lda_gamma = tidy(desc_lda, matrix = "gamma")

- lda_gamma

- lda_gamma = full_join(lda_gamma, nasa_keyword, by = c("document" = "id"))

- lda_gamma

In plots

- top_keywords <- lda_gamma %>%
-   filter(gamma > 0.9) %>%
-   count(topic, keyword, sort = TRUE)
- top_keywords

- top_keywords %>%
-   group_by(topic) %>%
-   top_n(5, n) %>%
-   ungroup %>%
-   mutate(keyword = reorder_within(keyword, n, topic)) %>%
-   ggplot(aes(keyword, n, fill = as.factor(topic))) +
-   geom_col(show.legend = FALSE) +
-   labs(title = "Top keywords for each topic",
-        x = NULL, y = "Number of documents") +
-   coord_flip() +
-   scale_x_reordered() +
-   facet_wrap(~ topic, ncol = 4, scales = "free")

Supervised LDA

- https://papers.neurips.cc/paper/3328-supervised-topic-models.pdf

- Fit LDA so as to best estimate Y