Ubuntu Dialogue Corpus: Natural Language Processing

Daniel Lee

November 24, 2017

The following analysis is inspired by Ubuntu Dialogue Corpus from Kaggle website. The idea is to be able to create a chatbot based on the dataset. Dataset can be found in here. The following is a description of the dataset from Kaggle:

> The new Ubuntu Dialogue Corpus consists of almost one million two-person conversations extracted from the Ubuntu chat logs, used to receive technical support for various Ubuntu-related problems. The conversations have an average of 8 turns each, with a minimum of 3 turns. All conversations are carried out in text form (not audio).

From the Ubuntu dialogues dataset, I will focus on `dialogs/3` folder, which contains 346108 .tsv files, each representing one conversation by different participants.

From the dataset, I will focus on exploring the following two questions for potential use in creating a chatbot:

1. Identify the most popular topics
    - Text will be processed using the `textmineR` package. 80% of the data is used for training set.
    - I used two algorithms to find potential topics: hierarchical clustering and K-means clustering.
2. Given a random conversation, suggest topics
    - I used the `tm` package to process the text.
    - I developed recommendations using method inspired by J. Breen's approach to sentiment analysis.

The final analysis was done using Amazon AWS EC2 CentOS Instance for faster processing.

Problem 1: Identify the most popular topics

```
# Use packages tm and SnowballC for processing the text

library(tm)
library(SnowballC)
library(stringr)
library(foreach)
library(doParallel)
library(textmineR)
# Calculate the number of cores
no_cores <- detectCores() - 1
registerDoParallel(cores = no_cores)
```

1a. Natural Language Processing using textmineR Package

```
# set random seed for reproducibility
set.seed(123)

# obtain n - total number of conversations
n = as.numeric(system("ls /home/daniel/dialogs/3/ | wc -l", intern = TRUE))
```

```
# randomly sample 80% of the conversation indexes for training set
train_idx <- sort(sample(x = 1:n, size = ceiling(n*0.8), replace = FALSE))


dataset <- foreach(i = 1:ceiling(n*0.8)) %dopar% {

  paste(read.delim(file = paste0('./dialogs/3/',train_idx[i],'.tsv'),
                   quote = '',
                   stringsAsFactors = FALSE,
                   header = FALSE)[,4], collapse = " ")

}

dtm <- CreateDtm(dataset,
                 stem_lemma_function =
                   function(x) SnowballC::wordStem(x,"porter"),
                 stopword_vec = c(tm::stopwords("english"),
                                  tm::stopwords("SMART"),
                                  "anyon", "ask", "can", "good",
                                  "got", "hello", "hey", "inst",
                                  "ive", "just", "know", "like",
                                  "look", "may", "mean", "new",
                                  "now", "one", "problem", "question",
                                  "say", "see", "set", "someon",
                                  "someth", "still", "support", "sure",
                                  "tell", "thank", "thing", "think",
                                  "time", "tri", "use", "want",
                                  "way", "will"),
                 cpus = no_cores)
```

```
## Warning in CreateDtm(dataset, stem_lemma_function = function(x)
## SnowballC::wordStem(x, : No document names detected. Assigning
## 1:length(doc_vec) as names.
```

```
##
  |
  |=======                                                |  10%
  |
  |=============                                          |  20%
  |
  |====================                                   |  30%
  |
  |==========================                             |  40%
  |
  |================================                       |  50%
  |
  |======================================                 |  60%
  |
  |============================================           |  70%
  |
  |==================================================     |  80%
  |
  |========================================================|  90%
  |
```

```
  |======================================================================| 100%
##
##
  |
  |=======                                                               |  10%
  |
  |=============                                                         |  20%
  |
  |====================                                                  |  30%
  |
  |==========================                                            |  40%
  |
  |================================                                      |  50%
  |
  |======================================                                |  60%
  |
  |============================================                          |  70%
  |
  |==========================================================            |  80%
  |
  |================================================================      |  90%
  |
  |======================================================================| 100%
```

1b. Top 50 Most Frequently Mentioned Words

```
# insert row names to dtm matrix
rownames(dtm) <- train_idx

# obtain the fifty most frequently occuring words in the conversations
termFreq <- colSums(dtm)
tf <- data.frame(term = names(termFreq), freq = termFreq)
tf <- tf[order(-tf[,2]),]
tf_50 <- tf[1:50,]

tf_50
```

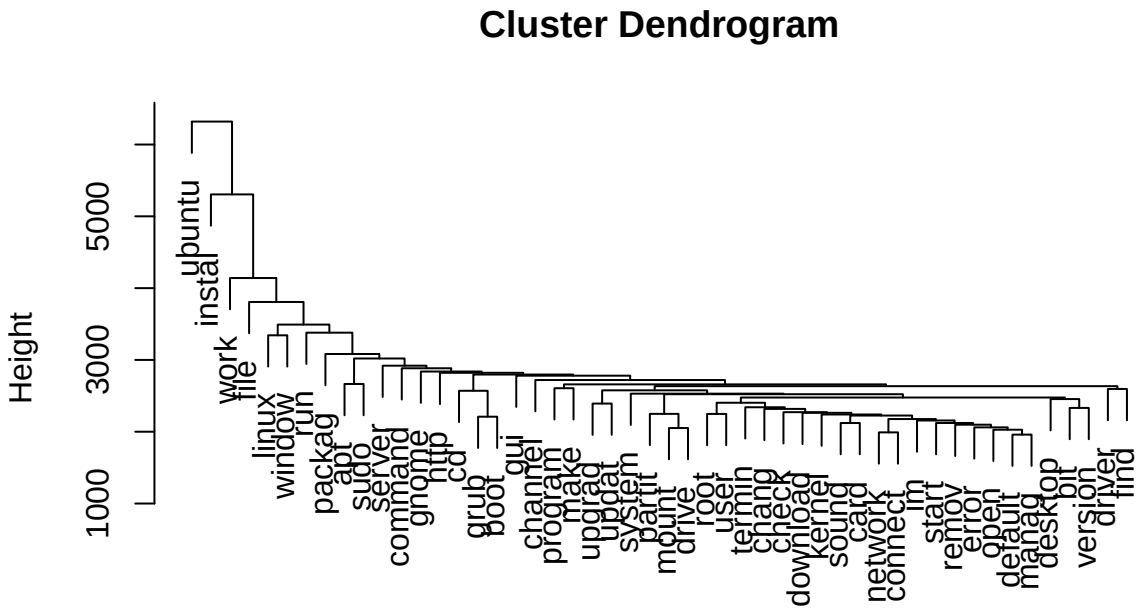
```
##              term   freq
## ubuntu     ubuntu 111227
## instal     instal  84372
## work         work  37972
## file         file  36509
## run           run  30377
## window     window  27419
## linux       linux  24513
## sudo         sudo  21183
## http         http  20249
## packag     packag  20212
## apt           apt  19637
## boot         boot  18546
## command   command  18053
## make         make  17589
```

```
## server     server  17090
## system     system  16983
## cd             cd  15982
## gui           gui  15797
## find         find  15487
## gnome       gnome  15075
## driver     driver  14080
## desktop   desktop  13783
## partit     partit  13677
## program   program  13549
## updat       updat  13491
## chang       chang  12384
## user         user  12349
## upgrad     upgrad  12321
## version   version  12136
## drive       drive  11907
## check       check  11739
## termin     termin  11477
## download download  11327
## start       start  11302
## error       error  10563
## card         card  10522
## connect   connect  10352
## channel   channel  10351
## open         open  10283
## im             im  10244
## root         root  10076
## grub         grub  10063
## bit           bit   9934
## mount       mount   9866
## manag       manag   9784
## network   network   9762
## remov       remov   9635
## kernel     kernel   9318
## sound       sound   9115
## default   default   8920
```

1c. Hierarchical Clustering Using the Top 50 Frequent Words

```
# transpose dtm for clustering
tdm = t(dtm)

# select words that are mentioned more than 100 times
tdm <- tdm[which(termFreq > 100), ]

# scale tdm
tdm_scale <- scale(tdm)
rownames(tdm_scale) <- rownames(tdm)

# obtain the top fifty most frequent words
tdm_50 <- tdm_scale[row.names(tdm_scale) %in% tf$term[1:50], ]
```

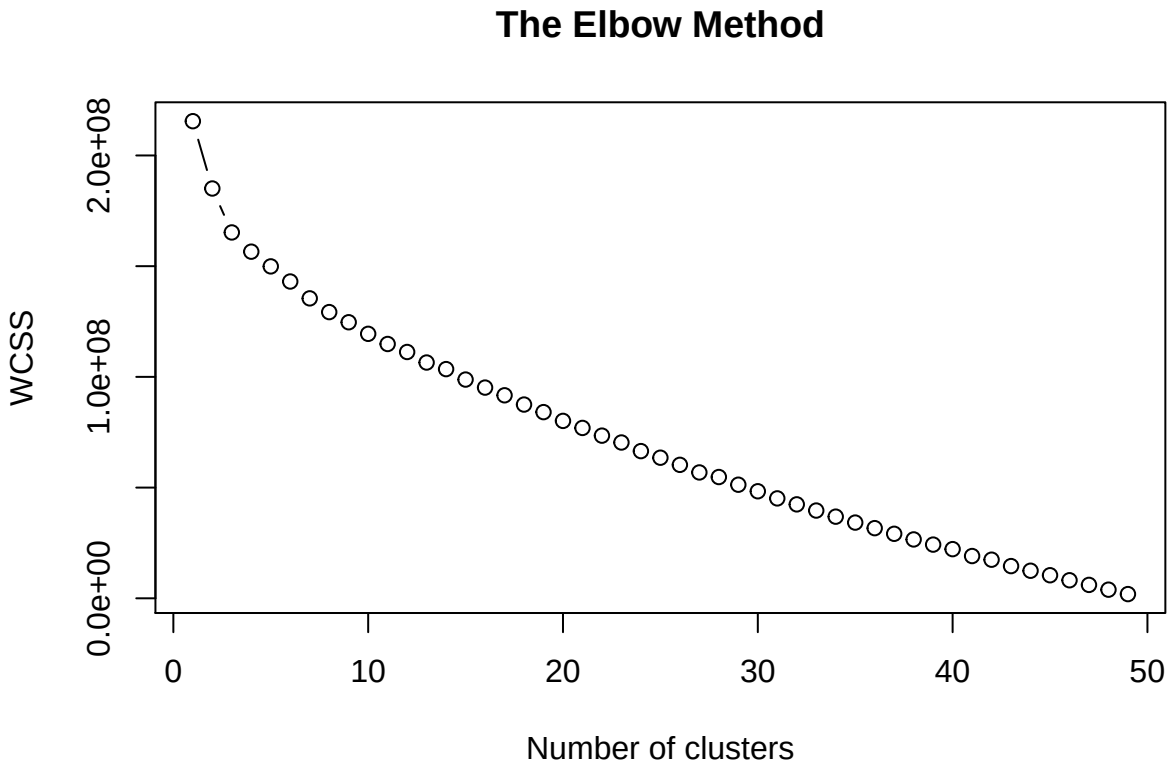
```r
# calculate euclidean distance
tdm_dist <- dist(tdm_50, method = "euclidean")

# hierarchical clustering
tdm_hclust <- hclust(tdm_dist)
plot(tdm_hclust)
```



1d. K-means Clustering Using the Top 50 Frequent Words

```r
# replace the na values in tdm_50 to zero
tdm_50[is.na(tdm_50)] <- 0

# for loop for calculating within cluster sum of squares using the top 50 mentioned words

wcss <- foreach(i = 1:(nrow(tf_50)-1),
                .combine = "c") %dopar% {

  sum(kmeans(x = tdm_50, centers = i, iter.max = 300)$withinss)

}

# plot the within cluster sum of squares for the clusters
plot(1:49,
```

```
    wcss,
    type = 'b',
    main = paste('The Elbow Method'),
    xlab = 'Number of clusters',
    ylab = 'WCSS')
```



Based on the within-cluster sum of squares plot, it seems as though the slope becomes less steep at around 14 clusters. As the number of clusters increase from 14, the curve becomes linear. I will choose k = 14.

```
# using the elbow method, choose center and run k means clustering method
kmeans = kmeans(x = tdm_50, centers = 14, iter.max = 300)

# display results from k-means clustering
sort(kmeans$cluster)
```

```
##    gnome   upgrad    updat  command     file  network  channel  connect
##        1        2        2        2        3        4        4        4
##     card   driver     find      gui     http   window     work   instal
##        4        4        4        4        4        5        6        7
##   ubuntu   packag    linux  default    sound   kernel    remov    manag
##        8        9       10       11       11       11       11       11
##    mount      bit     grub     root       im     open    error    start
##       11       11       11       11       11       11       11       11
## download   termin    check    drive  version     user    chang  program
##       11       11       11       11       11       11       11       11
##   partit  desktop       cd   system     make     boot   server      run
##       11       11       11       11       11       11       12       13
```

```
##      apt     sudo
##       14       14
```

Based on hierarchical clustering and the K-means clustering results, the following topics may be the most common support topics:

1. Installation - Ubuntu
2. Installation - packages (aptget)
3. General Ubuntu
4. Running file on windows and/or linux (compatibility)
5. Linux commands (sudo)
6. Upgrade or update (aptget)
7. Partitioning or mounting drive
8. Connection Issues
9. Gnome-Related
10. Graphics card error (nvidia)

Problem 2: Write a classifier (or topic detector) given a random conversation

```
# select random conversation from test set
x <- 1:n
random_text_idx <- sample(x = x[-train_idx], size = 1)

category <- c("Installation - Ubuntu",
              "Installation - Packages",
              "General Ubuntu",
              "Running file on windows and/or linux (compatibility)",
              "Linux commands (sudo)",
              "Upgrade or update (aptget)",
              "Partitioning or mounting drive",
              "Connection Issues",
              "Gnome-Related",
              "Graphics card error",
              "Other")

keyTerms <- c("instal ubuntu",
           "instal packag aptget",
           "ubuntu",
           "linux window",
           "command line sudo term",
           "upgrad updat aptget",
           "partit mount drive",
           "connect internet network wireless server",
           "gnome",
           "card graphic nvidia video sound driver",
           "")

Categories <- data.frame(Categories = category, Words = keyTerms)

# load random conversation
```

```
# input: conversation_idx - random index from test set
# input: categores - dataframe of top 10 most frequent categories
# output: recommendations - vector of recommendations
# output: convo_recommend - list containing the random conversation
#         and recommendations
# Given a random conversation, this function returns recommended categories for topic
topic_detector <- function(conversation_idx, categories) {

  conversation_vec =
    read.delim(file = paste0('./dialogs/4/',conversation_idx,'.tsv'),
                    quote = '',
                    stringsAsFactors = FALSE,
                    header = FALSE)[,4]

  conversation = paste(conversation_vec, collapse = " ")
  # empty vector that will contain match scores for categories
  scores <- vector()

  # empty vector that will return the recommendations
  recommendations <- vector()

  # empty list that will contain recommendations and random conversation
  convo_recommend <- list()

  # create a corpus of the conversations
  corpus = VCorpus(VectorSource(conversation))

  # make all text to lower case
  corpus = tm_map(corpus, content_transformer(tolower))

  # remove all numbers from text
  corpus = tm_map(corpus, removeNumbers)

  # remove all punctuations from text
  corpus = tm_map(corpus, removePunctuation)

  # remove common words using stopwords() from SnowballC package
  corpus = tm_map(corpus, removeWords, stopwords())

  # convert all text to stem words
  corpus = tm_map(corpus, stemDocument)

  # remove words not removed by stopwords() but not helpful for clustering
  corpus = tm_map(corpus, removeWords, c("anyon", "ask", "can", "good",
                                         "got", "hello", "hey", "inst",
                                         "ive", "just", "know", "like",
                                         "look", "may", "mean", "new",
                                         "now", "one", "problem", "question",
                                         "say", "see", "set", "someon",
                                         "someth", "still", "support", "sure",
                                         "tell", "thank", "thing", "think",
                                         "time", "tri", "use", "want",
                                         "way", "will"))
```

```
  # remove extra spaces
  corpus = tm_map(corpus, stripWhitespace)

  # convert to plain text document to create sparse matrix
  corpusPTD <- tm_map(corpus, PlainTextDocument)

  # sparse matrix dtm containing all the words and how many times the words
  # appear in a given conversation
  dtm = DocumentTermMatrix(corpusPTD)

  dtm.matrix <- as.matrix(dtm)

  sentence <- paste(colnames(dtm.matrix), collapse = " ")

  # split sentence into words with str_split from stringr package
  word.list = str_split(sentence, "\\s+")
  words = unlist(word.list)

  # compare words to the dictionaries of top ten topics
  for(i in 1:nrow(categories)-1) {

    categories.list = str_split(categories[i,2], "\\s+")
    categories_i = unlist(categories.list)
    scores[i] <- sum(match(words, categories_i), na.rm = TRUE)

  }

  # select the categories with high scores
  for(i in 1:length(scores)) {

    if(sum(scores) == 0) {

      recommendations[i] <- c("Other")
      break

    } else {

      recommendations[i] <- as.character(categories[which.max(scores), 1])
      scores[which.max(scores)] <- 0

    }

  }

  convo_recommend <- list("Conversation" = conversation_vec,
                          "Recommendations" = recommendations)

  return(convo_recommend)

}

# example run of function
topic_detector(conversation_idx = random_text_idx, categories = Categories)
```

```
## $Conversation
## [1] "How do you change the default application for a file extension in ubuntu?"
## [2] "what?"
## [3] "kdg ? gnome? other?"
## [4] "Gnome"
##
## $Recommendations
## [1] "Installation - Ubuntu" "General Ubuntu"        "Gnome-Related"
## [4] "Other"
```