Ubuntu Dialogue Corpus

Daniel Lee

November 23, 2017

The following analysis is inspired by Ubuntu Dialogue Corpus from Kaggle. 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/4` folder, which contains ~270,000 .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 top 10 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 top 10 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. Text Processing using tm

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

# obtain n - total number of conversations
n = as.numeric(system("ls /home/daniel/dialogs/4/ | 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/4/',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
```

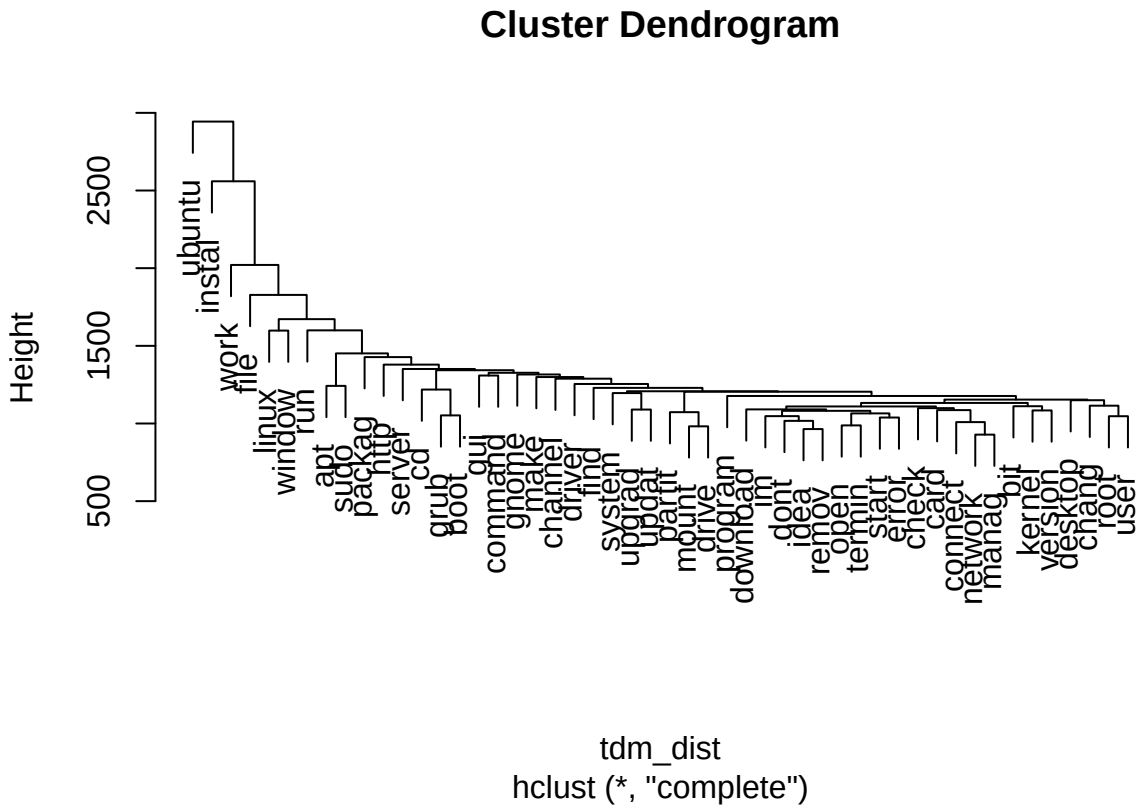
```
##             term   freq
## ubuntu     ubuntu 102996
## instal     instal  83570
## work         work  39675
## file         file  35541
## run           run  29661
## window     window  27078
## linux       linux  23954
## http         http  21451
## sudo         sudo  20840
## packag     packag  20010
## apt           apt  19196
## boot         boot  18782
## make         make  17839
## system     system  16989
```

```
## command   command  16462
## server     server  16205
## cd             cd  15763
## find         find  14994
## gui           gui  14790
## gnome       gnome  14421
## driver     driver  14348
## partit     partit  13618
## desktop   desktop  13315
## updat       updat  13074
## chang       chang  12218
## program   program  12202
## user         user  11983
## drive       drive  11816
## version   version  11772
## check       check  11764
## error       error  11622
## upgrad     upgrad  11579
## download download  11350
## start       start  11328
## im             im  10855
## connect   connect  10726
## termin     termin  10530
## card         card  10485
## open         open  10441
## grub         grub  10372
## bit           bit  10110
## mount       mount   9919
## root         root   9840
## manag       manag   9580
## dont         dont   9571
## kernel     kernel   9528
## remov       remov   9505
## network   network   9408
## channel   channel   9300
## idea         idea   9110
```

1c. Hierarchical Clustering Using the Top 50 Frequent Words

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

# select words that are mentioned more than 1000 times
tdm <- tdm[which(termFreq > 1000), ]

# 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], ]
```

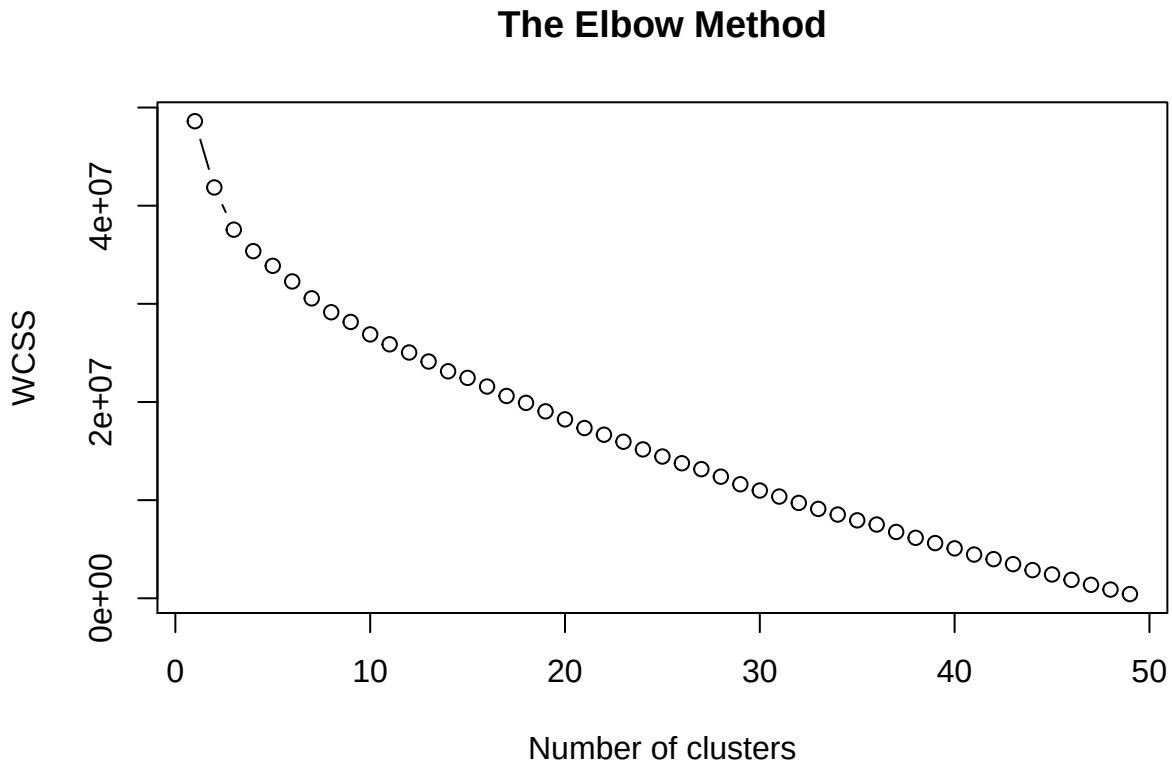
```r
# calculate euclidean distance
tdm_dist <- dist(tdm_50, method = "euclidean")

# 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)
```

```
##   instal    linux      run     work     file   window     http      gui
##        1        2        3        4        5        6        7        8
##  command  channel  network     card  connect   driver   server      apt
##        8        9        9        9        9        9        9       10
##     sudo   upgrad    updat   packag   ubuntu     grub       cd     boot
##       10       11       11       11       12       13       13       13
##     idea    remov   kernel     dont    manag     root    mount      bit
##       14       14       14       14       14       14       14       14
##     open   termin       im    start download    error    check  version
##       14       14       14       14       14       14       14       14
##    drive     user  program    chang  desktop   partit    gnome     find
##       14       14       14       14       14       14       14       14
```

```
##   system     make
##       14       14
```

Based on hierarchical clustering and the K-means clustering results, the following ten 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] "Anyone know?"
## [2] "Hey, I just installed ubuntu and the restricted driver for graphic card, I also rebooted and its
## [3] "system -> preferences -> screen resolution"
## [4] "what he said, ignore me"
##
## $Recommendations
## [1] "Graphics card error"     "Installation - Ubuntu"
## [3] "Installation - Packages" "General Ubuntu"
## [5] "Other"
```