Week 13: Text Mining and Natural Language Processing

Jessica Butts

Due 4/21/2020

Libraries

```r
library(tidyverse)   # For data manipulation
library(twitteR)     # For pulling in twitter data
library(tm)          # For preprocessing and creating corpus
library(qdap)        # For preprocessing corpus
library(textstem)    # For Lemmatization
library(RWeka)       # For creating n-grams
library(wordcloud)   # For creating word clouds
library(ldatuning)   # For topic modeling
library(topicmodels) # For topic modeling
library(tidytext)    # For topic modeling
library(ggwordcloud) # For comparing word clouds
library(caret)       # For machine learning
library(parallel)    # For parallelizing code
library(doParallel)  # For parallelizing code
```

Data Import and Cleaning

The first step in the data import is to authenticate the connection to Twitter in order to pull tweets. Then we search for 5,000 tweets that contain "#dog" in the tweet. We remove the retweets and convert this to a dataframe and then a tibble. We then change the encoding of the tweets from UTF-8 to ASCII. This dataset is then saved to the output folder as "tweets_original.csv."

The next step is to take the tibble and convert it to a corpus. After creating the corpus, we take several steps (listed below) to preprocess the tweets in order to distill the tweets down to the core words in the tweets.

Preprocessing Steps: 1. Remove all hashtags as these are not core words in the tweet 2. Remove all URLS as these also don't give us information about the content of the tweet 3. Replace abbreviations so that abbreviated words and their full versions are counted as the same word 4. Replace contractions so that we count these words as the same thing whether the user used a contraction or wrote out the whole thing 5. Remove punctuation so that words in the middle and end of sentences are counted as the same word 5. Remove numbers because they won't help understand the content of the tweet 6. Convert to lower case so that capitalization doesn't matter 7. Remove stopwords common in English 8. Remove extra white space 9. Create Lemmas so that different forms of the same word are counted as the same word.

Finally, we created a document term matrix with unigrams and bigrams and removed terms that had over 99.7% sparsity. This cutoff was chosen to retain 200 terms. We then convert this to a tibble for analysis.

```r
# # Authentication
# apikey <- "apikeygoeshere"
# apisecret <- "apisecretgoeshere"
# token <- "tokengoeshere"
# secrettoken <- "secrettokengoeshere"

# setup_twitter_oauth(apikey, apisecret, token, secrettoken)

# Pull the tweets from twitter
# imported_tbl <- searchTwitteR("#dog", n = 5000) %>%
#                    strip_retweets() %>%
#                    twListToDF() %>%
#                    as_tibble()
#
# # Modify raw text in imported_tbl
# imported_tbl$text <- imported_tbl$text %>%
#                          iconv("UTF-8", "ASCII", sub = "")

# Save the data set of tweets
#write_csv(imported_tbl, "../output/tweets_original")

# read in saved tbl
imported_tbl <- read_csv("../output/tweets_original")
```

```
## Parsed with column specification:
## cols(
##   text = col_character(),
##   favorited = col_logical(),
##   favoriteCount = col_double(),
##   replyToSN = col_character(),
##   created = col_datetime(format = ""),
##   truncated = col_logical(),
##   replyToSID = col_double(),
##   id = col_double(),
##   replyToUID = col_double(),
##   statusSource = col_character(),
##   screenName = col_character(),
##   retweetCount = col_double(),
##   isRetweet = col_logical(),
##   retweeted = col_logical(),
##   longitude = col_double(),
##   latitude = col_double()
## )
```

```r
# Create corpus and preprocess the tweets
twitter_cp <- imported_tbl %>%
                add_column(doc_id = 1:nrow(imported_tbl), .before = 1) %>%
                DataframeSource() %>%
                VCorpus() %>%
                # remove hashtags
                tm_map(content_transformer(function(x) str_remove_all(x, pattern = "#\\w*"))) %>%
                # remove urls
                tm_map(content_transformer(function(x) str_remove_all(x, "https://[\\w|.|/]*"))) %>%
                tm_map(content_transformer(replace_abbreviation)) %>%
                tm_map(content_transformer(replace_contraction)) %>%
                tm_map(removePunctuation) %>%
                tm_map(removeNumbers) %>%
                tm_map(content_transformer(str_to_lower)) %>%
                tm_map(removeWords,stopwords("en")) %>%
                tm_map(stripWhitespace) %>%
                tm_map(content_transformer(lemmatize_strings))
```

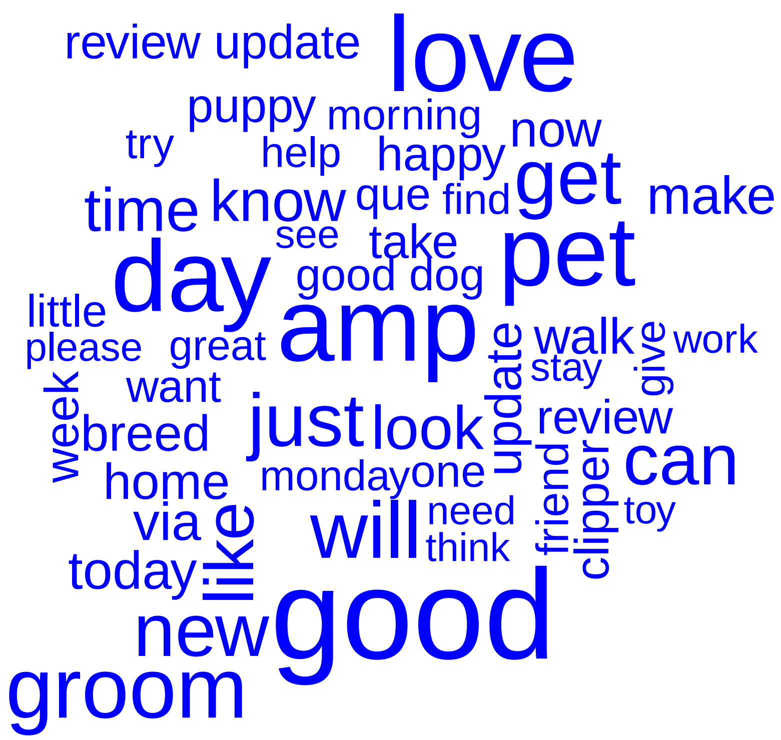
```
## Registered S3 methods overwritten by 'textclean':
##   method           from
##   print.check_text qdap
##   print.sub_holder qdap
```

```
# Function to create unigrams and bigrams from data
tokenizer <- function(x) {
  NGramTokenizer(x, Weka_control(min = 1, max = 2))
}

# Tokenize and remove sparse terms
twitter_dtm <- DocumentTermMatrix(twitter_cp,
                                  control = list(tokenize = tokenizer)) %>%
                  removeSparseTerms(.997)

# convert DTM to a tibble for analysis and remove the search term "dog"
twitter_tbl <- as_tibble(as.matrix(twitter_dtm))
twitter_tbl <- twitter_tbl[,-which(colnames(twitter_tbl) == "dog")]

# Remove rows with no remaining terms
dropped_tbl <- imported_tbl[rowSums(twitter_tbl) > 0,]
```

Visualization

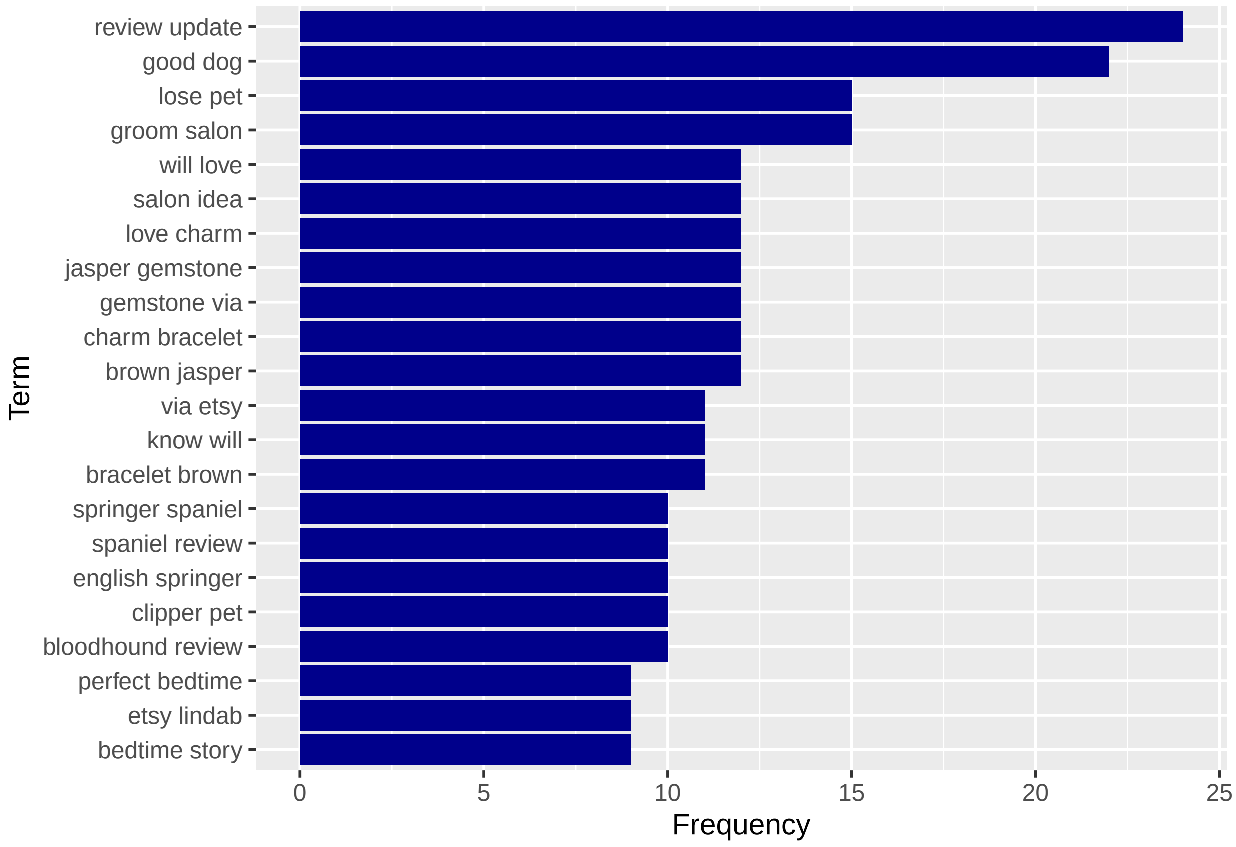
We pass the column names and the column sums (word frequency) from twitter_tbl to the wordcloud function and keep the top 50 most common words. We then create a tibble where the first column is the tokens that remain in the analysis, and the second column is the frequency of that token to make the bar chart easier to create. Then we create the bar chart of the most common bigrams. To do this, we subset the summary tibble to keep only tokens that contain a space. We then pass the top 20 most frequent words into ggplot to create the plot.

```
# Create wordcloud with 50 terms
wordcloud(colnames(twitter_tbl), colSums(twitter_tbl), max.words = 50, colors = "blue")
```



```r
# Create horizontal bar chart of bigrams
twitter_sum <- tibble(terms = colnames(twitter_tbl),
                      freq = colSums(twitter_tbl))
top_n(twitter_sum[str_detect(twitter_sum$terms, " "),], 20) %>%
  ggplot(aes(x = reorder(terms, freq), y = freq)) +
  geom_col(fill = "darkblue") +
  coord_flip() +
  labs(y = "Frequency", x = "Term")
```

```
## Selecting by freq
```



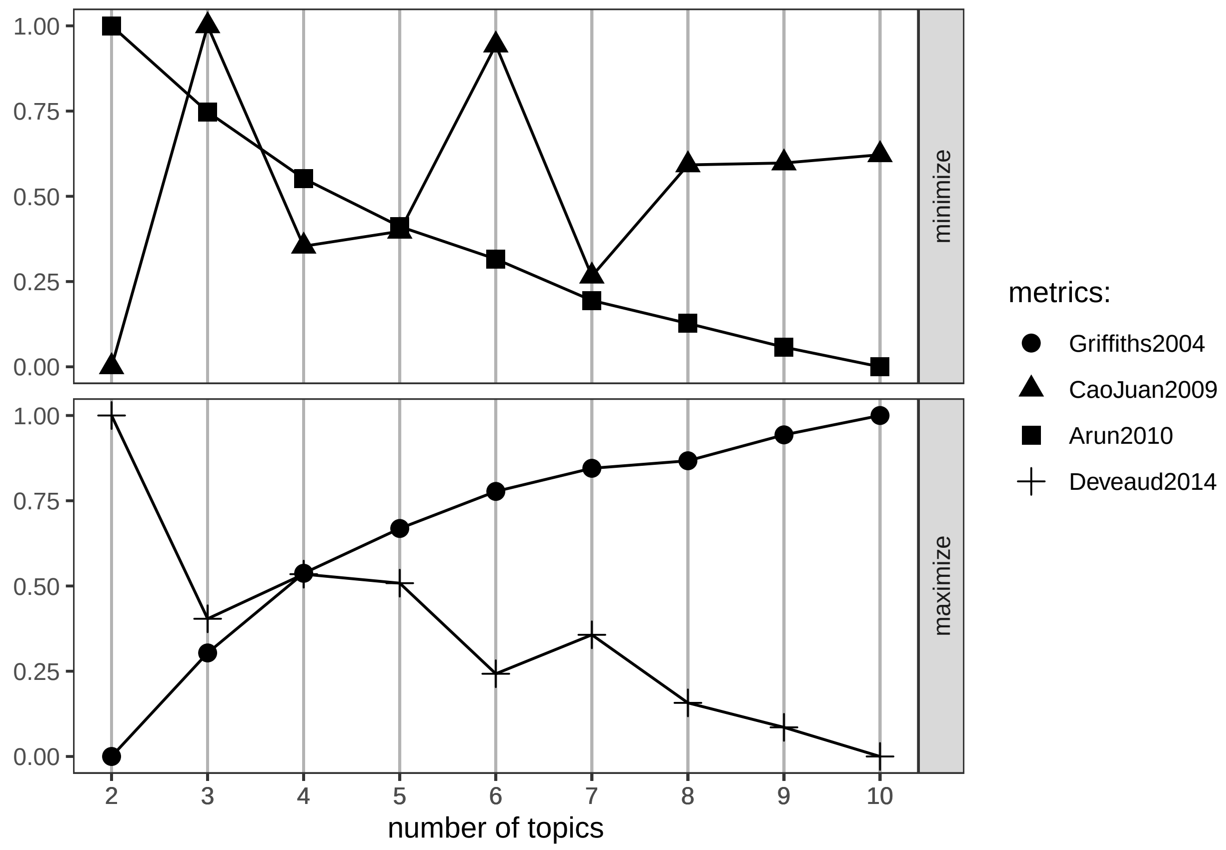
Analysis

Topic Modeling

For the topic modeling, we can't have any rows with all zeros, so we remove the rows that sum to 0 (indicates that none of the remaining tokens appear in that document). Then we create a plot to investigate the number of topics in the data set displaying results for 2 to 10 topics. Based on the plot, Arun2010 and Griffiths2004 are uninformative since they are approximately straight lines. Deveaud2014 has a maximum at 2 topics, and CaoJuan2009 has a minimum at 2 topics. Thus we select 2 topics.

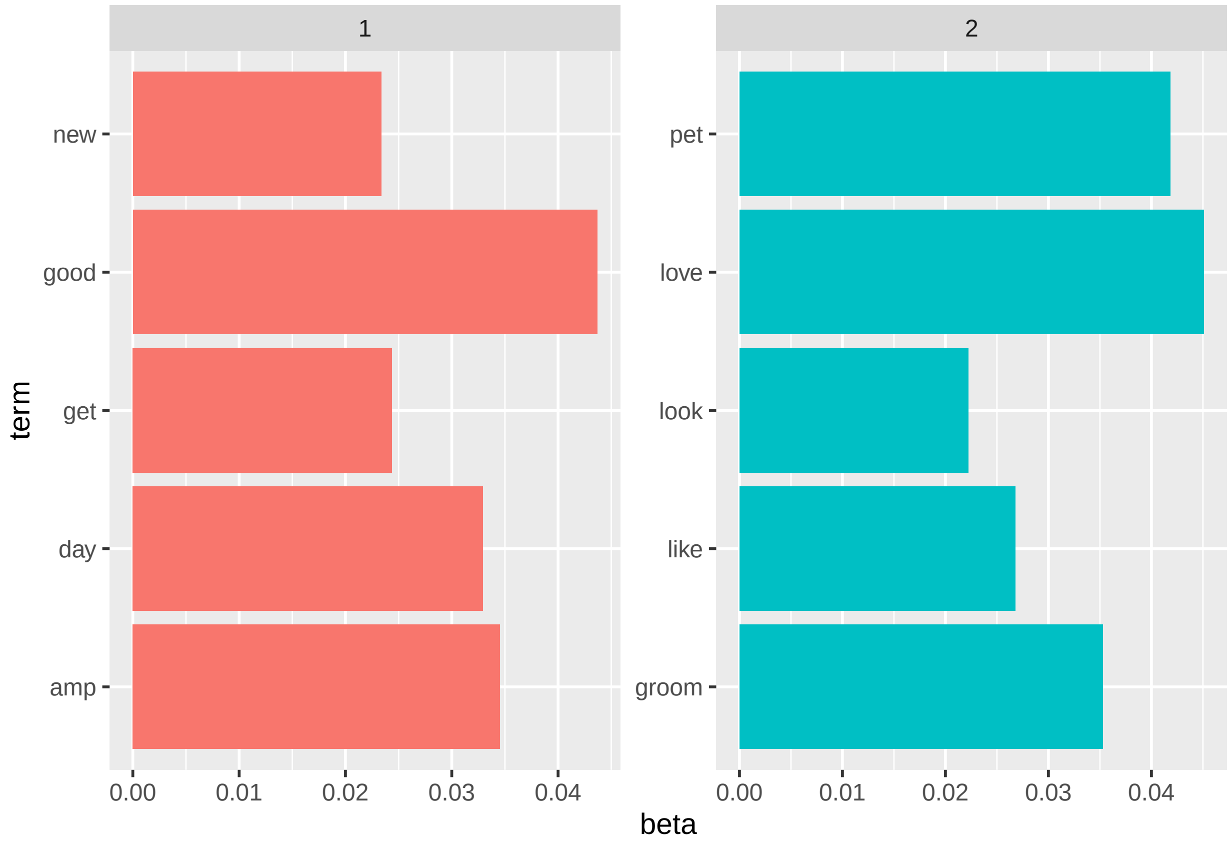
We then created two visuals. The first is a horizontal bar graph that plots the top 5 most likely words to be in each topic. The second is a wordcloud for each topic where the size of the words indicates frequency.

```r
# dtm with blank rows removed
twitter_tbl <- twitter_tbl[rowSums(twitter_tbl) > 0,]

# Create plots of metrics to determine the number of topics
tuning <- FindTopicsNumber(twitter_tbl,
                             topics = 1:10,
                             metrics = c("Griffiths2004",
                                         "CaoJuan2009",
                                         "Arun2010",
                                         "Deveaud2014"),
                            control = list(seed = 37))
FindTopicsNumber_plot(tuning)
```

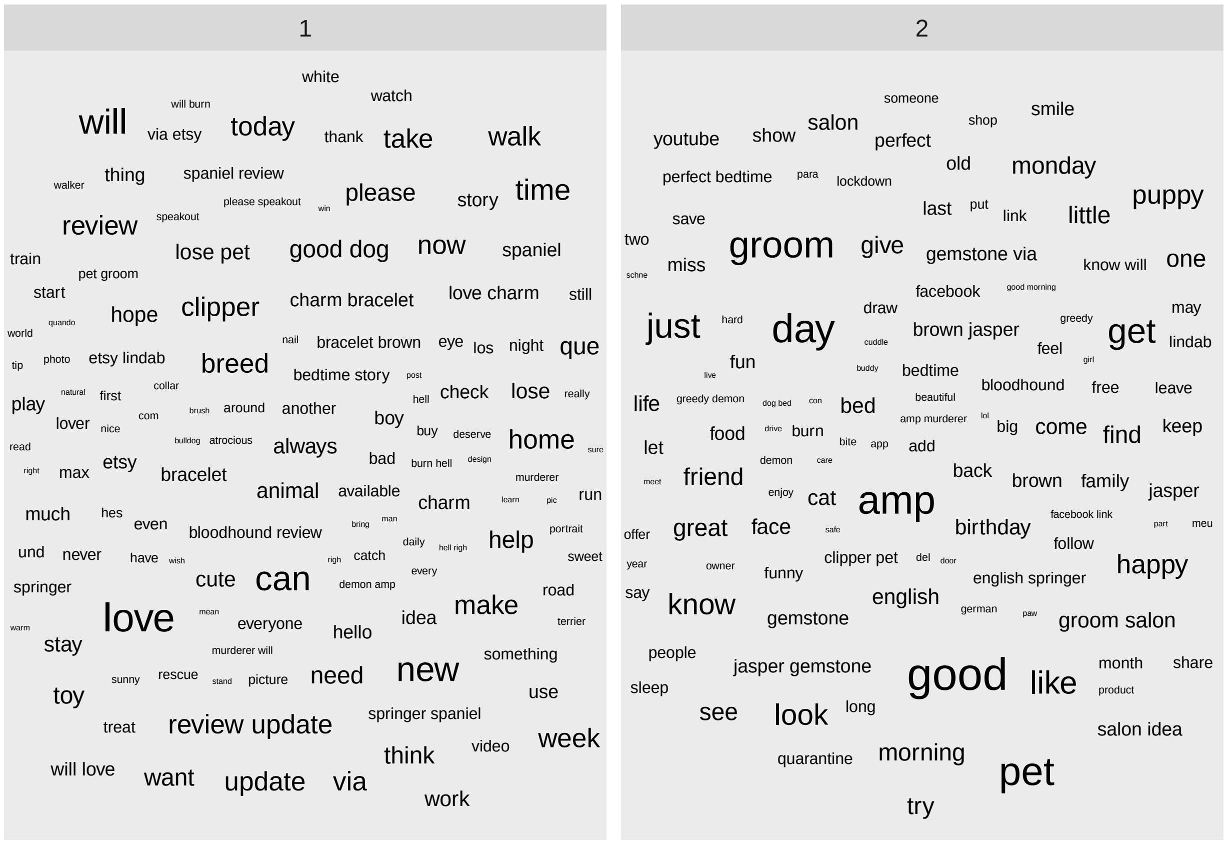


```
# Create model using  3 topics
lda_results <- LDA(twitter_tbl, 2, control = list(seed = 37))
# Posterior probabilities of being in each topic
lda_betas <- tidy(lda_results, matrix = "beta")
# Posterior probabilities of documents containing topics
lda_gammas <- tidy(lda_results, matrix = "gamma")

# Decide which topic each term belongs to
term_topic <- lda_betas %>%
                group_by(term) %>%
                arrange(desc(beta)) %>%
                top_n(1, beta)

# Add topic column to twitter_tbl
twitter_sum <- bind_cols(twitter_sum, topic = term_topic$topic)

# 5 most likely words to be in each topic
lda_betas %>%
  group_by(topic) %>%
  top_n(5, beta) %>%
  arrange(topic, -beta) %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()
```

```
## Warning in mutate_impl(.data, dots, caller_env()): Unequal factor levels:
## coercing to character

## Warning in mutate_impl(.data, dots, caller_env()): binding character and factor
## vector, coercing into character vector

## Warning in mutate_impl(.data, dots, caller_env()): binding character and factor
## vector, coercing into character vector
```



```
# Wordcloud by topic
ggplot(twitter_sum, aes(label = terms, size = freq)) +
  geom_text_wordcloud() +
  facet_wrap(~topic)
```



Based on the bar graphs and wordclouds, the first topic seems to have positive emotional words. The bar graph has the words “new” and “good,” and the wordcloud has words like “love”, “home”, and “hope”. The second topic seems to have more verbs. The bar graph has the words “groom” and “look,” and the wordcloud has words like “give”, “like”, and “try”. The bar graphs and word clouds don’t match entirely though since “love” is in topic 2 in the bar graph but topic 1 in the word clouds. The two aren’t exactly the same since the most likely words to be in each topic is not the same as the the most likely topic for a word; the first conditions on the topic, and the second conditions on the word.

```r
# Decide which topic each document belongs to
doc_topic <- lda_gammas %>%
                group_by(document) %>%
                top_n(1, gamma) %>%
                slice(1) %>%
                ungroup %>%
                mutate(document = as.numeric(document)) %>%
                arrange(document)

# Tabular summary of most likely topic per tweet
 table(doc_topic$topic)
```

```
##
##   1   2
## 588 439
```

It looks like there are 588 tweets in Topic 1 and 439 in Topic 2.

Machine Learning

Now, we are interested in investigation whether the popularity of a tweet measured by the number of favorites can be predicted by the words used in the tweet. We create two different models. The reduced model predicts popularity based on the words used in the tweet alone. The full model predicts popularity based on the words used in the tweet and the topic of the tweet. To include topic as a predictor, we must create a dummy variable. The first topic is the reference group. The variable "dummy_topic_2" is 1 if the tweet belongs to topic 2 and 0 otherwise. We also parallelize this code to use the number of cores available - 1 to speed up processing.

```
# Create dummy variables for topic.
# 0 = Topic 1; 1 = Topic 2
dummy2 <- ifelse(doc_topic$topic == 2, 1, 0)

# Data sets for the different models
data_red_model <- cbind(twitter_tbl,
                        favoriteCount = dropped_tbl$favoriteCount)
data_full_model <- cbind(twitter_tbl,
                         favoriteCount = dropped_tbl$favoriteCount,
                         topic2 = dummy2)

# Create cores to use in parallelization
local_cluster <- makeCluster(detectCores()-1)
registerDoParallel(local_cluster)

# Create folds so that models are comparable
set.seed(7)
folds <- createFolds(data_red_model[,"favoriteCount"], 10)

# Run support vector regression and compute 10-fold cv statistics with only tokens as predictors
svm_model_red <- train(favoriteCount ~ .,
                       data = data_red_model,
                       method = "svmLinear2",
                       tuneLength = 5,
                       trControl = trainControl(method = "cv",
                                                indexOut = folds,
                                                verboseIter = T)
)
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting cost = 2 on full training set
```

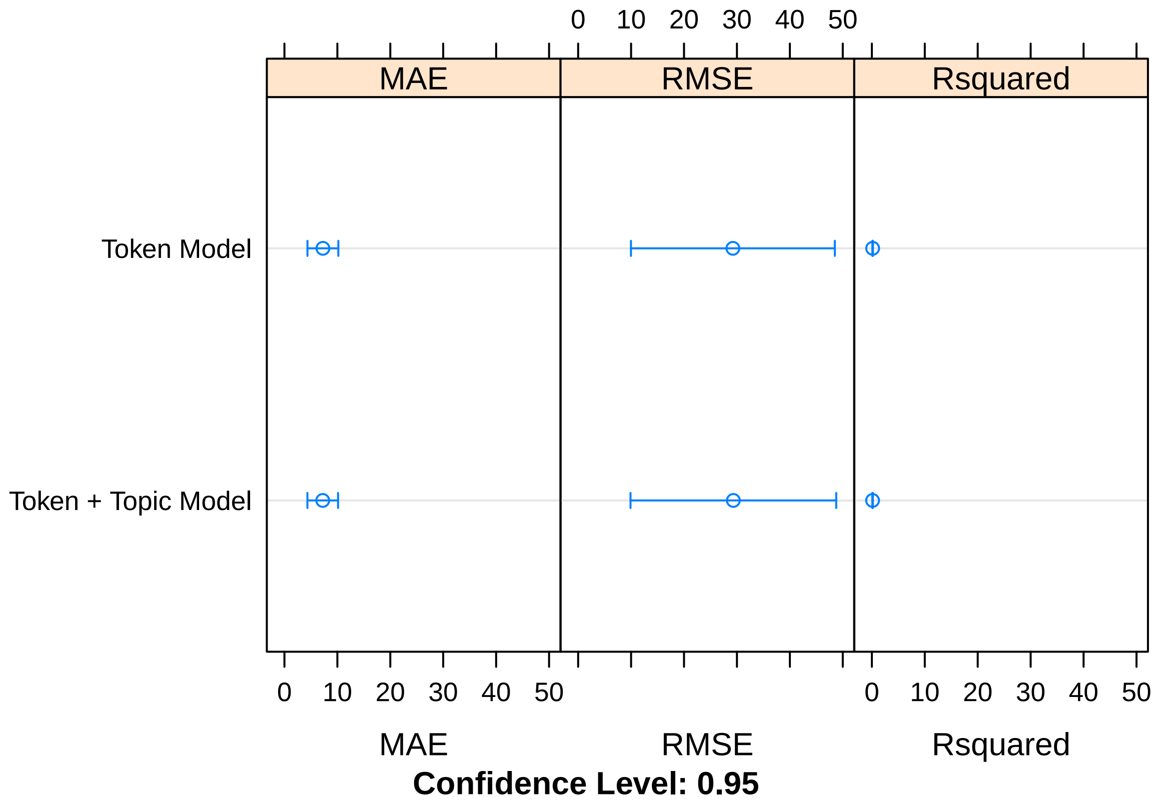
```
# Run support vector regression and compute 10-fold cv statistics adding topic as predictor
svm_model_full <- train(favoriteCount ~ .,
                        data = data_full_model,
                        method = "svmLinear2",
                        tuneLength = 5,
                        trControl = trainControl(method = "cv",
                                                 indexOut = folds,
                                                 verboseIter = T)
)
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting cost = 2 on full training set
```

```r
# Stop cluster
stopCluster(local_cluster)
registerDoSEQ()

# Compare results
summary(resamples(list("Token Model" = svm_model_red,
                       "Token + Topic Model" = svm_model_full)))
```

```
##
## Call:
## summary.resamples(object = resamples(list(`Token Model` = svm_model_red,
##  `Token + Topic Model` = svm_model_full)))
##
## Models: Token Model, Token + Topic Model
## Number of resamples: 10
##
## MAE
##                         Min.  1st Qu.   Median    Mean  3rd Qu.     Max. NA's
## Token Model         2.98577 4.496491 6.140538 7.26822 8.526483 16.39658    0
## Token + Topic Model 3.20343 4.531293 6.200643 7.23690 8.488927 16.61203    0
##
## RMSE
##                         Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## Token Model         4.239769 7.648979 21.73130 29.23740 40.26108 88.20195    0
## Token + Topic Model 4.532030 7.775583 21.56853 29.32717 40.37212 88.66777    0
##
## Rsquared
##                            Min.    1st Qu.    Median      Mean   3rd Qu.
## Token Model         0.068742855 0.09383888 0.1170427 0.1491963 0.2016550
## Token + Topic Model 0.008392448 0.09686533 0.1276720 0.1343181 0.1996236
##                          Max. NA's
## Token Model         0.2780112    0
## Token + Topic Model 0.2389207    0
```

```r
# Plot comparisons
dotplot(resamples(list("Token Model" = svm_model_red,
                       "Token + Topic Model" = svm_model_full)))
```



Based on the above results, it does not appear that adding the topic variable improves prediction of popularity of tweets as measured by favorite counts.

Final Interpretation

It doesn't seem like there is a strong difference in emotion associated with the two topics; both tend to be positive based on the words in the wordclouds. The first topic tends to be more descriptive where the second topic tends to have more actions. Since the topics both tend to be positive, it doesn't seem likely that the topics or emotions are are important in predicting the popularity of the tweets. It does not seem that the topic adds any additional information over what the words themselves provide in the machine learning models. The RMSE, MAE, and R^2 values are very similar between the full and reduced models, so we would likely choose the reduced model to predict popularity. It seems that the words themselves explain about 12% of the variation in the popularity of tweets, so that isn't much evidence of a strong linear relationship.

I am not sure that this design is the best design to answer this question though. I think that running a sentiment analysis to identify the dominant sentiment in a tweet and then running an ANOVA to see if tweet popularity is related to any particular sentiment would more directly answer the question. It's also possible that adding the number of favorites and number of retweets would be a better index of tweet popularity than number of favorites alone.

Another potential limitation is the choice of number of topics to use; the results are not deterministic, so they would change with a different seed. The topic modeling itself is random and could change even with the same number of topics. It also only looks for words that appear together across documents, so there is no guarantee that the topic is based on any type of sentiment.