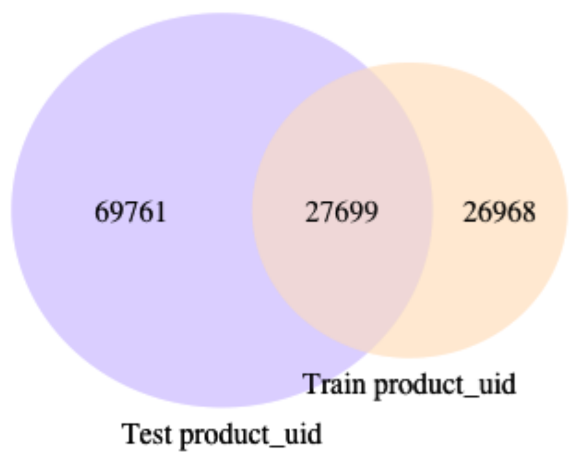
Data Analytics Final Project

Home Depot Competition

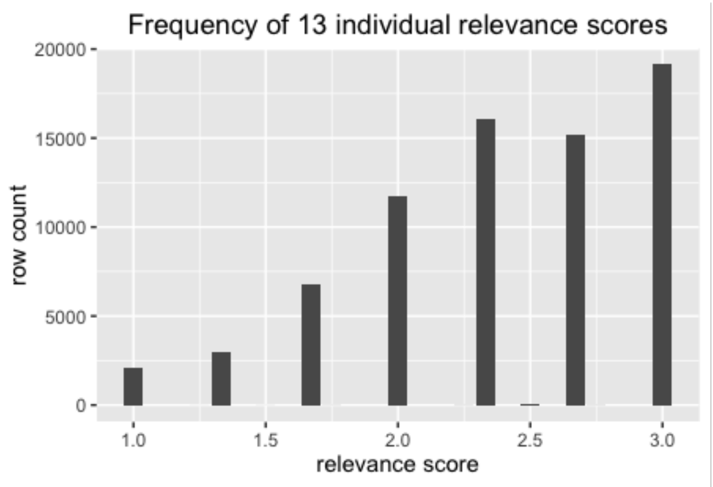
Group Members: Tong Wei (TOW6), Fengxi Liu (FEL34), Jinrong Liu (JIL181) Zhenyu Peng(ZHP6)

1. Data Exploration

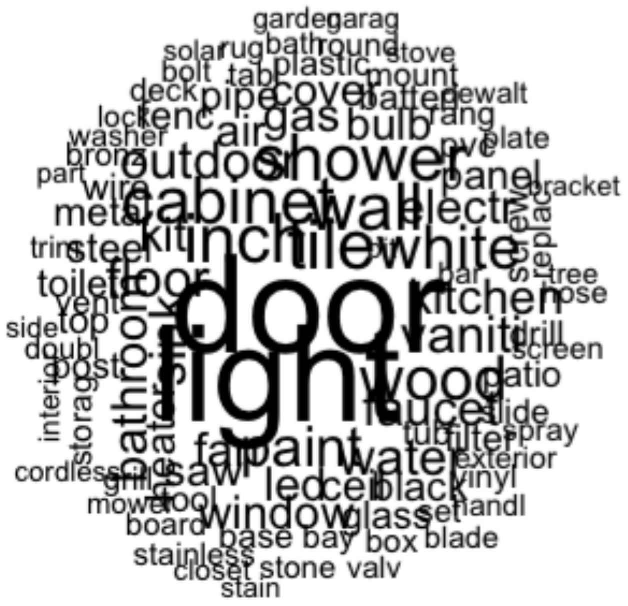
a. Venn Diagram



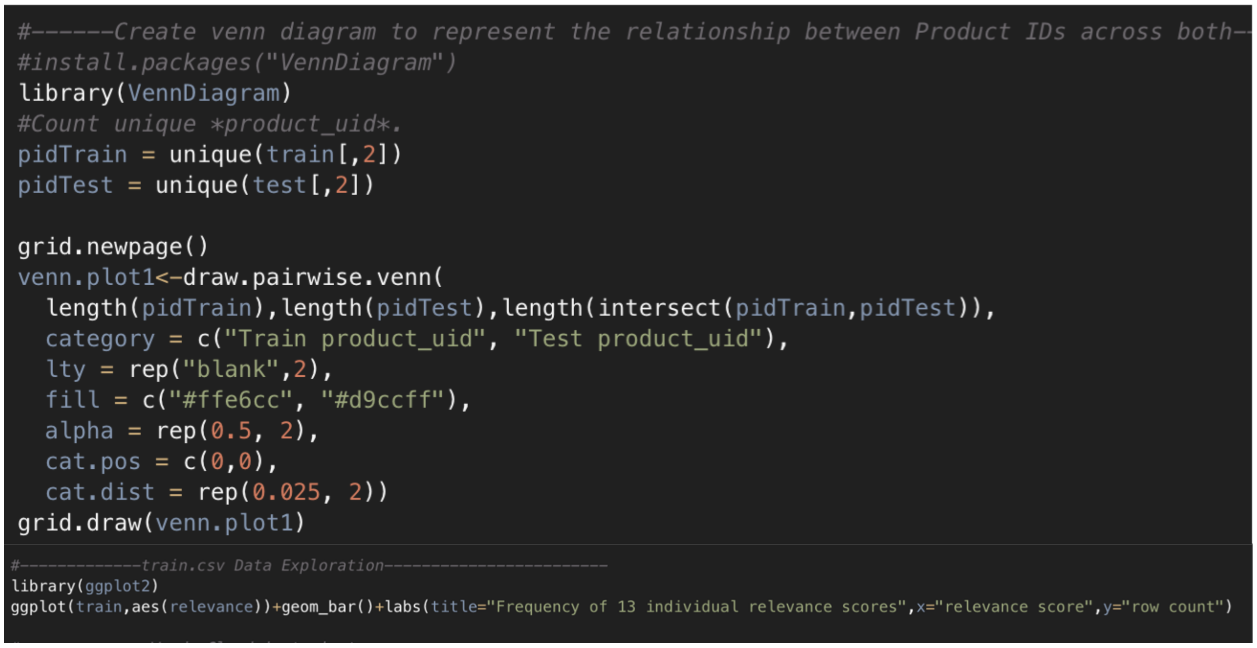
b. Histogram



c. WordsCloud

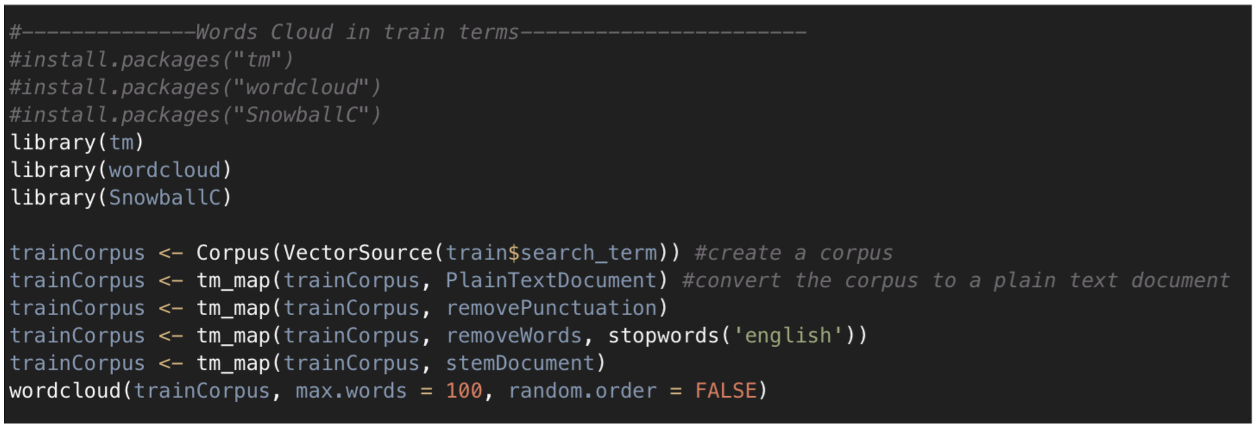


d. Codes



```
#------Create venn diagram to represent the relationship between Product IDs across both-
#install.packages("VennDiagram")
library(VennDiagram)
#Count unique *product_uid*.
pidTrain = unique(train[,2])
pidTest = unique(test[,2])

grid.newpage()
venn.plot1<-draw.pairwise.venn(
  length(pidTrain),length(pidTest),length(intersect(pidTrain,pidTest)),
  category = c("Train product_uid", "Test product_uid"),
  lty = rep("blank",2),
  fill = c("#ffe6cc", "#d9ccff"),
  alpha = rep(0.5, 2),
  cat.pos = c(0,0),
  cat.dist = rep(0.025, 2))
grid.draw(venn.plot1)
```

```
#------------train.csv Data Exploration-----------------------
library(ggplot2)
ggplot(train,aes(relevance))+geom_bar()+labs(title="Frequency of 13 individual relevance scores",x="relevance score",y="row count")
```

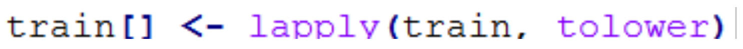


```r
#--------------Words Cloud in train terms-----------------------
#install.packages("tm")
#install.packages("wordcloud")
#install.packages("SnowballC")
library(tm)
library(wordcloud)
library(SnowballC)

trainCorpus <- Corpus(VectorSource(train$search_term)) #create a corpus
trainCorpus <- tm_map(trainCorpus, PlainTextDocument) #convert the corpus to a plain text document
trainCorpus <- tm_map(trainCorpus, removePunctuation)
trainCorpus <- tm_map(trainCorpus, removeWords, stopwords('english'))
trainCorpus <- tm_map(trainCorpus, stemDocument)
wordcloud(trainCorpus, max.words = 100, random.order = FALSE)
```

2. Preparing Data

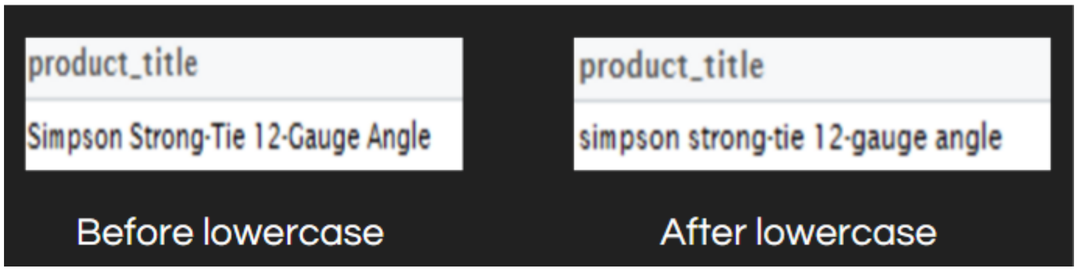
We do the data preparing following the steps on the guidance.

Step one: Lowercase. Tolower function is used here.



```r
train[] <- lapply(train, tolower)
```

Result:



product_title
Simpson Strong-Tie 12-Gauge Angle

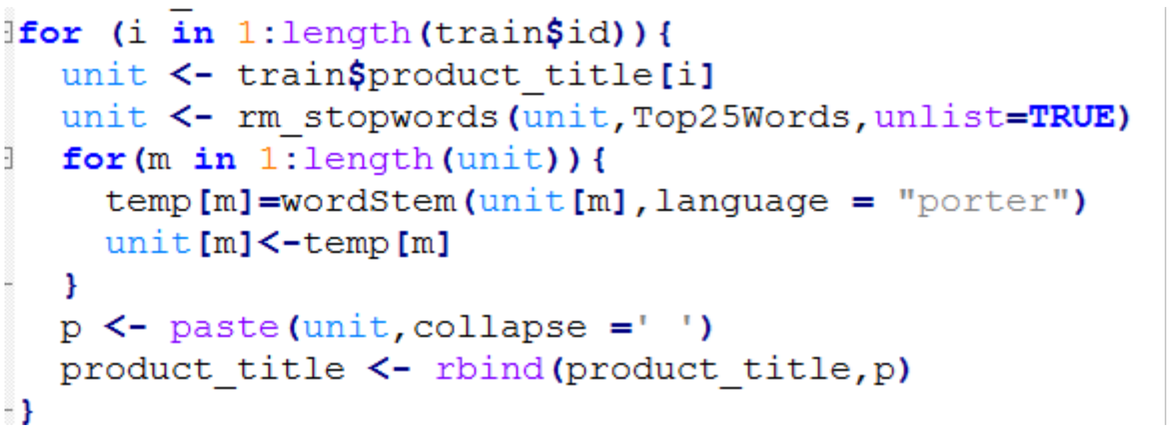
Before lowercase

product_title
simpson strong-tie 12-gauge angle

After lowercase

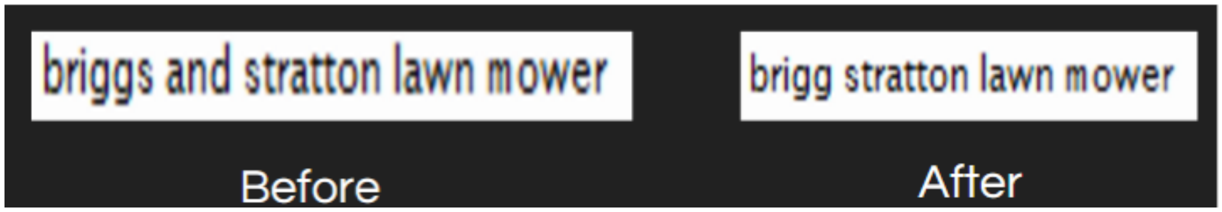
Step two: Remove stopwords and stem data:

We used the functions in QDAP package to achieve removing stopwords in a for loop. At the same time, we found we can do the stem in the loop. We choose porter stemmer in the SnowballC package, and stemming data in the second for loop inside the first loop.



```
for (i in 1:length(train$id)){
  unit <- train$product_title[i]
  unit <- rm_stopwords(unit,Top25Words,unlist=TRUE)
  for(m in 1:length(unit)){
    temp[m]=wordStem(unit[m],language = "porter")
    unit[m]<-temp[m]
  }
  p <- paste(unit,collapse =' ')
  product_title <- rbind(product_title,p)
}
```

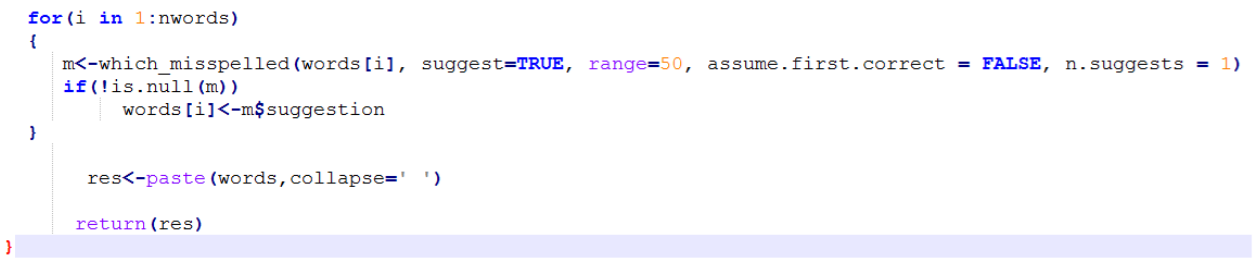
Results:



```
briggs and stratton lawn mower      brigg stratton lawn mower
           Before                             After
```

Step three: Correct misspelled words

This step is important because this will greatly affect our final result. We use which_misspelled function in QDAP package and look up into the QDAP dictionary. We can achieve all the correcting in a loop.

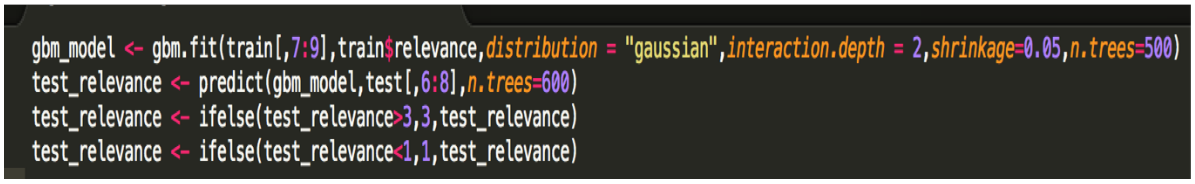


```
  for(i in 1:nwords)
  {
    m<-which_misspelled(words[i], suggest=TRUE, range=50, assume.first.correct = FALSE, n.suggests = 1)
    if(!is.null(m))
        words[i]<-m$suggestion
  }

     res<-paste(words,collapse=' ')

    return(res)
}
```

3. Model Selection and Model Evaluation

Model choosing: gbm model (Generalized Boosted Regression Models)

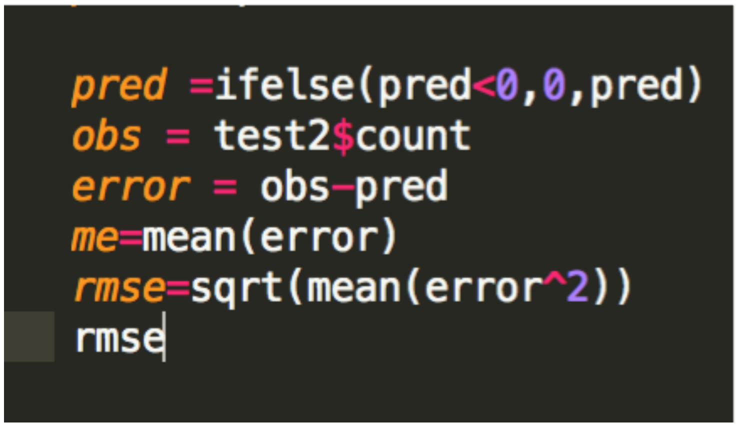
R: install.packages('gbm')

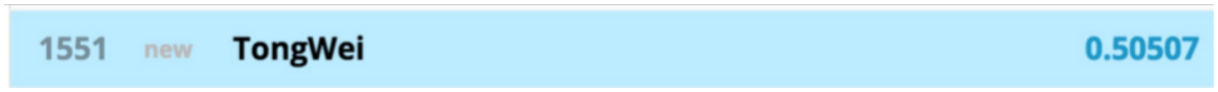

```
gbm_model <- gbm.fit(train[,7:9],train$relevance,distribution = "gaussian",interaction.depth = 2,shrinkage=0.05,n.trees=500)
test_relevance <- predict(gbm_model,test[,6:8],n.trees=600)
test_relevance <- ifelse(test_relevance>3,3,test_relevance)
test_relevance <- ifelse(test_relevance<1,1,test_relevance)
```

Model evaluation: k-fold validation

K-fold cross validation is one way to improve over the holdout method. The data set is divided into *k* subsets, and the holdout method is repeated *k* times. Each time, one of the *k* subsets is used as the test set and the other *k-1* subsets are put together to form a training set. Then the average error across all *k* trials is computed.

RMSE result:


```
pred =ifelse(pred<0,0,pred)
obs = test2$count
error = obs-pred
me=mean(error)
rmse=sqrt(mean(error^2))
rmse
```


1551 new TongWei 0.50507