Home Depot Search Term Learning Project

Pengjie Jiang

May 5, 2016

Project discription

In this project, Home Depot is asking to help them improve their customers' shopping experience by developing a model that can accurately predict the relevance of search results. Search relevancy is an implicit measure Home Depot uses to gauge how quickly they can get customers to the right products. Currently, human raters evaluate the impact of potential changes to their search algorithms, which is a slow and subjective process.By removing or minimizing human input in search relevance evaluation, Home Depot hopes to increase the number of iterations their team can perform on the current search algorithms

```
## this project is done on Window environment,
## therefore, the current working derectory will change
## depending on the platform that using this script.
rm(list=ls())
setwd("C://CS//Kaggle")
# Load Packages needed
Needed <- c("tm", "ggplot2", "wordcloud","Snowball",
            "RWeka", "arulesViz", "RWekajars","randomForest")
  install.packages(Needed, dependencies=TRUE,repos="https://cloud.r-project.org/")
```

```
## Installing packages into 'C:/Users/USER/Documents/R/win-library/3.2'
## (as 'lib' is unspecified)
```

```
## Warning: package 'Snowball' is not available (for R version 3.2.5)
```

```
## Warning: dependencies 'Rcampdf', 'Rgraphviz', 'Rpoppler',
## 'tm.lexicon.GeneralInquirer', 'graph' are not available
```

```
##
##   There are binary versions available but the source versions are
##   later:
##             binary  source needs_compilation
## RWeka       0.4-26  0.4-27             FALSE
## RWekajars 3.7.13-1 3.9.0-1             FALSE
##
## package 'tm' successfully unpacked and MD5 sums checked
## package 'ggplot2' successfully unpacked and MD5 sums checked
## package 'wordcloud' successfully unpacked and MD5 sums checked
## package 'arulesViz' successfully unpacked and MD5 sums checked
## package 'randomForest' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\USER\AppData\Local\Temp\RtmpYDd1DU\downloaded_packages
```

```
## installing the source packages 'RWeka', 'RWekajars'
```

```
## Warning: running command '"C:/PROGRA~1/R/R-32~1.5/bin/x64/R" CMD INSTALL -
## l "C:\Users\USER\Documents\R\win-library\3.2" C:\Users\USER\AppData\Local
## \Temp\RtmpYDd1DU/downloaded_packages/RWekajars_3.9.0-1.tar.gz' had status 1

## Warning in install.packages(Needed, dependencies = TRUE, repos = "https://
## cloud.r-project.org/"): installation of package 'RWekajars' had non-zero
## exit status

## Warning: running command '"C:/PROGRA~1/R/R-32~1.5/bin/x64/R" CMD INSTALL -
## l "C:\Users\USER\Documents\R\win-library\3.2" C:\Users\USER\AppData\Local
## \Temp\RtmpYDd1DU/downloaded_packages/RWeka_0.4-27.tar.gz' had status 1

## Warning in install.packages(Needed, dependencies = TRUE, repos = "https://
## cloud.r-project.org/"): installation of package 'RWeka' had non-zero exit
## status

##
## The downloaded source packages are in
##   'C:\Users\USER\AppData\Local\Temp\RtmpYDd1DU\downloaded_packages'
```

```
library(arules)
```

```
## Loading required package: Matrix

##
## Attaching package: 'arules'

## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(tm)
```

```
## Loading required package: NLP

##
## Attaching package: 'tm'

## The following object is masked from 'package:arules':
##
##     inspect
```

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:NLP':
##
##     annotate
```

```
  library(randomForest)
```

```
## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

Data read

```
##read in data
train<-read.csv(file ="train.csv", header=TRUE, sep = ",")
##head(train, 3)
test<-read.csv(file ="test.csv", header=TRUE, sep = ",")
##head(test, 3)
attr<-read.csv(file ="attributes.csv", header=TRUE, sep = ",")
##head(attr, 3)
prod_des<-read.csv(file ="product_descriptions.csv", header=TRUE, sep = ",")
##head(prod_des, 3)
```

Combine dataset of description with train and test data

```
train <- merge(train,prod_des, by.x = "product_uid", by.y = "product_uid", all.x = TRUE, all.y = FALSE)
test <- merge(test,prod_des, by.x = "product_uid", by.y = "product_uid", all.x = TRUE, all.y = FALSE)
```

Customized function for text mining of discription

Text clean function

```
clean_text <- function(text)
{
  gsub(" ","  ", text)
  gsub("°", "degrees", text)
  gsub(" v ", "volts", text)
  gsub("^", " ", text)
}
```

Word match function

```
word_match <- function(words,title,desc)
{
  n_title <- 0
  n_desc <- 0
  words <- unlist(strsplit(words," "))
  ## text mining on desc
  pdCorpus <- Corpus(VectorSource(desc))
  # to lower case
  pdCorpus <- tm_map(pdCorpus, content_transformer(tolower))
  # remove punctuation
  pdCorpus <- tm_map(pdCorpus, removePunctuation)
  # remove stopwords
  pdCorpus <- tm_map(pdCorpus, removeWords, stopwords(kind="en"))
  pdCorpus <- tm_map(pdCorpus, stripWhitespace)
  # remove stemming words
  dictCorpus <- pdCorpus
  pdCorpus <- tm_map(pdCorpus, stemDocument)
  #pdCorpus <- tm_map(pdCorpus, stemCompletion, dictionary = dictCorpus)
  # # Building a Document-Term Matrix
  # pdDTm <- TermDocumentMatrix(pdCorpus, control = list(minwordlength=1))
  # pdDTm <- as.matrix(pdDTm)
  # sort by freq
  #v <- sort(rowSums(pdDTm), decreasing = T)
  nwords <- length(words)
  for(i in 1:nwords)
  {
    n_char <- nchar(words[1])

    pattern <- paste("(^| )",words[i],"($| )",sep="")
    #pattern <- words[i]
    n_title <- n_title + grepl(pattern,title,perl=TRUE,ignore.case=TRUE)
    n_desc <- n_desc + grepl(pattern,pdCorpus$content,perl=TRUE,ignore.case=TRUE)
  }
  return(c(n_title,nwords,n_desc))
}
```

Model built

```
train$product_title <- clean_text(train$product_title)
train$product_description <- clean_text(train$product_description)
train$search_term <- clean_text(train$search_term)

test$product_title <- clean_text(test$product_title)
test$product_description <- clean_text(test$product_description)
test$search_term <- clean_text(test$search_term)

cat("Get number of words and word matching title in train\n")
```

```
## Get number of words and word matching title in train
```

```
train_words <- as.data.frame(t(mapply(word_match,train$search_term,train$product_title,train$product_des
train$nmatch_title <- train_words[,1]
train$nwords <- train_words[,2]
train$nmatch_desc <- train_words[,3]

cat("Get number of words and word matching title in test\n")
```

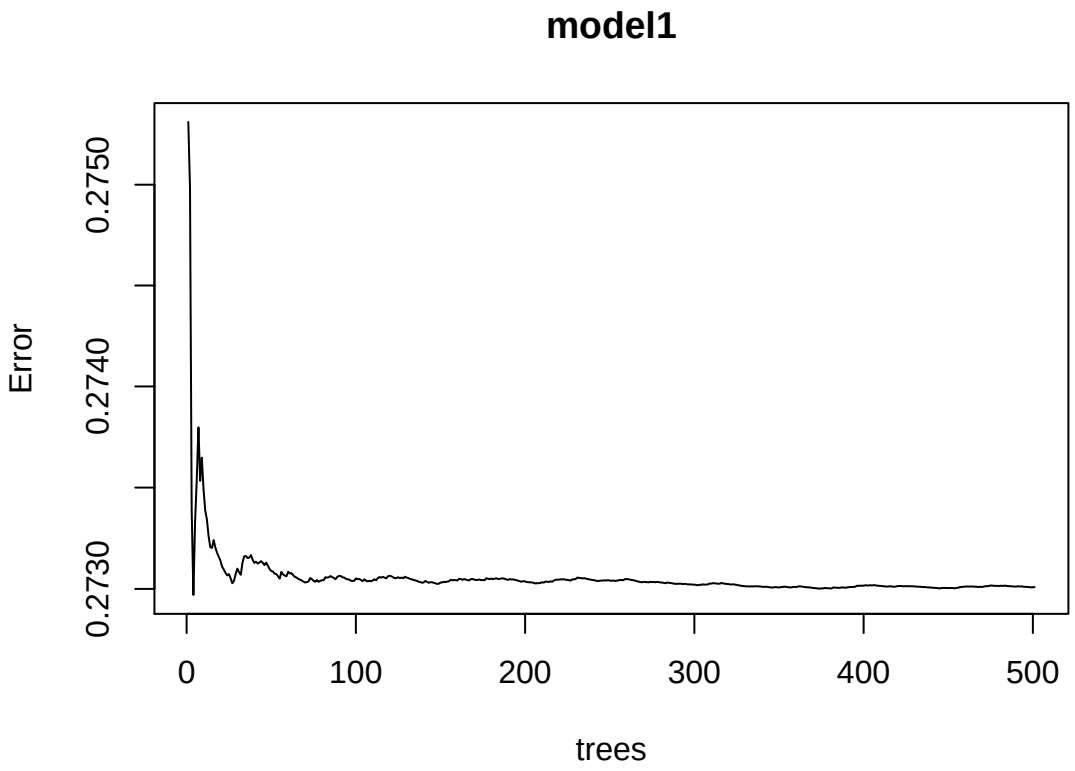
```
## Get number of words and word matching title in test
```

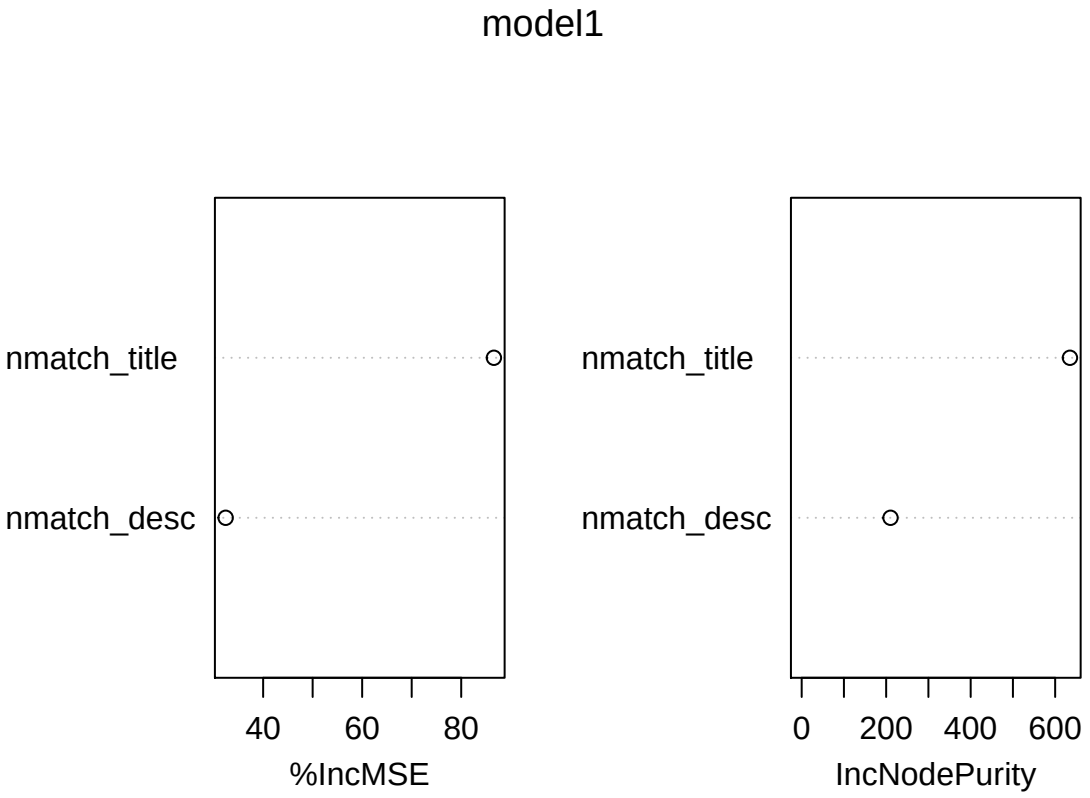
```
test_words <- as.data.frame(t(mapply(word_match,test$search_term,test$product_title,test$product_descri
test$nmatch_title <- test_words[,1]
test$nwords <- test_words[,2]
test$nmatch_desc <- test_words[,3]

model1<-randomForest(relevance~nmatch_title+nmatch_desc,
                     data=train,
                     type=regression,
                     importance=T,
                     na.action = na.omit,
                     ntree=501
                )
```

Prediction

```
test_result<-as.data.frame(predict(model1, test, type="response"))
test_result<-cbind(test$id,test_result)
colnames(test_result)<- c("id","predict_relevance")

write.csv(test_result,"test_result_submission.csv")
```

Plots


model1
Error
0.2730
0.2740
0.2750
0
100
200
300
400
500
trees


model1
nmatch_title
nmatch_desc
40
60
80
%IncMSE
nmatch_title
nmatch_desc
0
200
400
600
IncNodePurity

####

Conclusion In this project, I used the text mining “tm”" package in R to match the words cleaned from product description, in order to test the relevance with both search term and product title as well as product description. The model is built using random forest method with 501 trees. Shown in the plots above, the most important variable in this regression is world match of product description then the match of product title.