Home Depot Product Search Relevance

Team DABOYS
Haitao Zhou
Yingzhi Yang
Xin Jin

ABSTRACT
In this paper, we introduce the data for the Home Depot Product Search Relevance competition and develop a model to predict the relevance of search results. By using text mining and gbm model, we finally improved the accuracy and beat the benchmark.

Keywords
Relevance, gbm, tf-idf, Product Search

1.INTRODUCTION

Shoppers rely on Home Depot's product authority to find and buy the latest products and to get timely solutions to their home improvement needs. With the click of a mouse or tap of the screen, customers expect the correct results to their queries.
Since the customers can't search for the products so accurate, the more quickly they can find what they want, the more satisfied they will feel for the shopping experience.
The objectives are that we should analyze the relevance between the products and the searching key words, and improve the searching system for the customers to find their product as soon as possible.

2.DATA

We got our data files from the Kaggle website. File train.csv describes the data that we used to find how the relevance is calculated using the relationship between product titles and search terms. We are going to use test.csv to test our solution. The relevance value will be calculated for every items in the profile. The other two files regarding to prescription and attributes can be utilized to improve the accuracy of results.

2.1 Overview of Data

The figure 1 gives us a perceptual intuition of the data. After removing the redundant data, There are 97460 unique products in test, 54667 unique products in the train, and 27699 of them are common. We can see the intersection of product_uid acroos the three files.

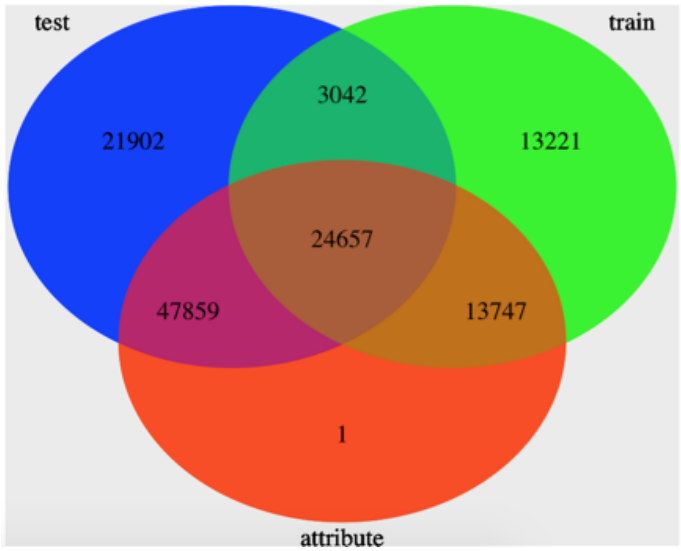

Figure 1 Overlap of products

Figure 2 shows that most relevance scores fall between 2 and 3, It helps us to build the model in future works and can serve as the reference when we calculate the final results.

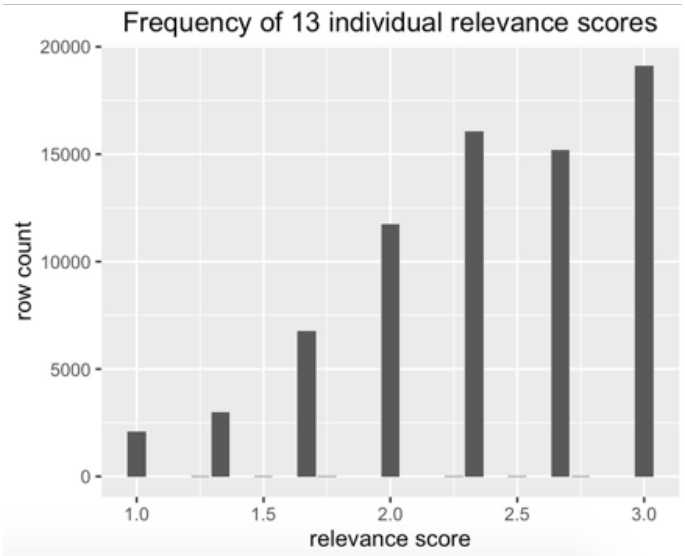

Figure 2 Frequency of relevance scores

In Figure 3, we produce the word cloud of search term in train.csv by sorting the frequency of the words in decreasing order.

Figure 3 Word Cloud of Search Term

2.2 TF-IDF

Tf–idf, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and text mining. The formula of its calculation is as following in figure 4.

$$\mathrm{tf}(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}}$$

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D)$$

Figure 4 Formula of TF-IDF

Based on the tfidf algorithm, we can deal with the data and reach the proportion of a word in a document. Figure 5 is an example that shows how to perform this calculation. And Figure 6 shows us the partial result.

```
dtm<-DocumentTermMatrix(all_text,control=list(tolower=TRUE,removePunctuation=TRUE,
                                              removeNumbers=TRUE,stopwords=TRUE,
                                              stemming=TRUE,weighting=function(x) weightTfIdf(x,normalize=T)))
```

Figure 5 TfIdf weighting of Product tile in train.csv

	door	kit	light	pack	steel	wall	white
1	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0	0.0000000
2	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0	0.0000000
3	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0	0.0000000
4	0.0000000	0.4100062	0.0000000	0.0000000	0.0000000	0	0.0000000
5	0.0000000	0.4100062	0.0000000	0.0000000	0.0000000	0	0.0000000
6	0.0000000	0.0000000	0.0000000	0.0000000	0.4953160	0	0.0000000
7	0.0000000	0.0000000	0.0000000	0.0000000	0.4953160	0	0.0000000
8	0.0000000	0.0000000	0.0000000	0.0000000	0.4953160	0	0.0000000
9	0.0000000	0.0000000	0.7896836	0.0000000	0.0000000	0	0.0000000
10	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0	0.0000000
11	0.0000000	0.0000000	0.0000000	0.5998131	0.0000000	0	0.0000000
12	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0	0.0000000

Figure 6 Result of TfIdf

3. MODEL OF SOLUTION

3.1 GBM

To find the relevance of the search-terms with product titles and product descriptions, we need to choose a right predict model. GBM is short for Gradient Boosting Model. Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. GBM package in R implements extensions to Freund and Schapire's AdaBoost algorithm and J. Friedman's gradient boosting machine. Includes regression methods for least squares, absolute loss, logistic, Poisson, Cox proportional hazards partial likelihood, multinomial, t-distribution, AdaBoost exponential loss, learning to Rank, and Huberized hinge loss. We can use GBM algorithm to improve the decision tree.
In the GBM model, four parameters will affect the result.

(1) Distribution
We need to choose a distribution function for our model firstly. Generally speaking, there are two main distribution functions –"Bernoulli"and "Gaussian". The Bernoulli function is used for classification problems and the Gaussian function is used for regression problems respectively. Since our model is a regression one, we should choose "Gaussian"for distribution.

(2) N. trees
The n.trees is the parameter chosen for number of iterations. Since our model need to learn from train.csv and predict the relevance in test.csv, the iterations should be large enough. We chose 500 for n.trees at the first time, and we found the result became better when we enlarge n.trees (to 1000).

(3) Shrinkage

Shrinkage of GBM represents the learning rate of the model. So we should set it as small as possible. As the shrinkage goes small we need to make n.trees larger to ensure the accuracy of the model. In our project, we chose 0.05 for it.

(4) Interaction depth
GBM uses interaction depth parameter as a number of splits it has to perform on a tree. We set three in our model.

After testing multiple times, we reached the best solution by setting depth to 3, n.trees to 1000 and shrinkage to 0.05 as is shown in figure 7.

```
gbm_model <- gbm.fit(train[,7:9],train$relevance,distribution = "gaussian",
                     interaction.depth = 3,shrinkage=0.05,n.trees=1000)
test_relevance <- predict(gbm_model,test[,6:8],n.trees=1000)
test_relevance <- ifelse(test_relevance>3,3,test_relevance)
test_relevance <- ifelse(test_relevance<1,1,test_relevance)
```

Figure 7 GBM Model

3.2 Other model

(1) LM
LM is a linear model in R. LM is one of the most basic regression models in R. It can use the data in train.csv to predict the relevance by using a linear regression. The result shows that LM is the best in all models other than GBM.

(2) SVM
SVM is short for "Support Vector Machines"and it's a model in Package e1071 in R. It can be used to carry out general regression and classification, as well as density-estimation. SVM are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. The R interface to libsvm in package e1071, svm(), was designed to be as intuitive as possible. Models are fitted and new data are predicted as usual, and both the vector/matrix and the formula interface are implemented. In our project, we use the default svm model for prediction.

(3) Rpart

Rpart in R is a model of recursive partitioning for classification, regression and survival trees. Recursive partitioning is a statistical method for multivariable analysis. Recursive partitioning creates a decision tree that strives to correctly classify members of the population by splitting it into sub-populations based on several dichotomous independent variables. The process is termed recursive because each sub-population may in turn be split an indefinite number of times

until the splitting process terminates after a particular stopping criterion is reached. The advantages for rpart is that it can generate clinically more intuitive models that do not require the user to perform calculation, and it allows varying prioritizing of misclassifications in order to create a decision rule that has more sensitivity or specificity. We used rpart to predict relevance but the result was not good.

4.TEST RESULT

Among the several models we built, when setting specific parameters, GBM model performs best and scored 0.50267 compared with other algorithms such as svm, lm, rpart(the smaller the point is, the better). A screen shot of result of prediction is shown in figure 9.

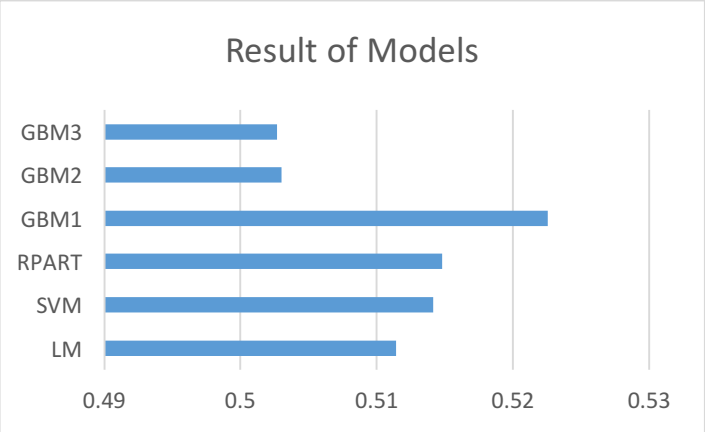

Figure 8 Comparison of different models

id	relevance
1	2.21375728
4	2.11174056
5	2.27679969
6	2.15719851
7	2.15719851
8	2.26167325
10	2.72872384
11	2.68597735
12	2.08494122
13	2.70230125
14	2.49239323
15	2.47000469
19	2.51663358
22	2.32785603
24	2.27679969
25	2.38802687
26	2.26167325
28	2.59308488
29	2.35243646

Figure 9 Prediction of Relevance in test.csv

5. References

[1] Rajaraman, A.; Ullman, J. D. (2011). "Data Mining". Mining of Massive Datasets(PDF) pp. 1–17.

[2] Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. https://cran.r-project.org/web/packages/e1071/

[3] Recursive Partitioning and Regression Trees
ftp://cran.rproject.org/pub/R/web/packages/rpart/rpart.pdf