Home Depot Product Search Relevance

Kewen Zhang, Uni: kz2246, Pengfei Wang Uni: pw2406
Xiaoci Xing, Uni: xx2203, Ziyue Wu, Uni: zw2338

Abstract

In this search relevance project, our goal was to build a model to predict the relevance of search items and product on homedepot.com, given the searching terms, product properties. It can be considered as a text mining and regression problem. Our solution mainly consists of the following three parts: 1). text cleaning, where we clean our data for building better models. 2). Feature extraction, turning text content to meaningful predictors, where we used counting features, distance features, tf-idf features with cosine similarity and customized features. 3). Regression model selection comparison, where we applied multiple linear regression, ridge regression, random forest and extreme gradient boosting. Among those methods, a combination of all features and random forest or extreme gradient boosting gave us the best performance result (least RMSE).

Introduction

Problem of interest

When searching on Google, Bing, and those search system in e-commercial website, we sometimes may not be given a satisfactory result. Especially for e-commercial website, a better system of "search and match" is a rigid demand. For this motivation, we think that improving the relevance between search term and its corresponding items will help us improve the searching system since the item with highest score of relevance will rank first. If we provide a model which can predict the relevance with given search term and all the products information, the problem is solved. Therefore, the main purpose of our project will become building a model to predict the relevance scores between search terms and given products.

Data Set

We obtained our data set from Kaggle.com, which is posted by Home Depot. Our raw data sets include the following information:

Products description - includes a brief text content for describing products.

Products attributes - includes properties of corresponding products, like color, materials, brand.

Products title - text content like product name.

Search term - text content customers might type in for searching purpose.

Relevance - a score provided by Home Depot indicates the relevance of search term and corresponding product. 1 stands for least relevant and 3 most relevant. The score is an average value rated by three or four humans. e.g.

	id	product_uid	product_title	search_term	relevance
0	2	100001	Simpson Strong-Tie 12-Gauge Angle	angle bracket	3.0
1	3	100001	Simpson Strong-Tie 12-Gauge Angle	l bracket	2.5
2	9	100002	BEHR Premium Textured DeckOver 1-gal. #SC-141 ...	deck over	3.0

	product_uid	name	value
279	100010	Color/Finish	Silver/Gray
282	100010	Material	Steel
283	100010	MFG Brand Name	Valley View Industries

There are totally 74067 observations in raw data.

Organization of the Report

Rest of the report follows the following structure:

1. Model Setup (1.1 assumption, 1.2 preprocessing data, 1.3 Statistical Model Employed 1.3.1 feature selection, 1.3.2 linear regression model building)
2. Statistical Inference (2.1 multicollinearity analysis, 2.2 backward stepwise)
3. Model Checking (3.1 ridge regression 3.2 random forest, 3.3 extreme gradient boosting, 3.4 performance comparison)
4. Conclusion

1. Model Setup

1.1 Assumption

Since most of our data are text contents, we proposed the following assumptions for our dataset.

1. All texts may contain typo and spelling mistakes. e.g. “helloWorld”, “helo World”.
2. English stop words in the text content has no information for our modeling, e.g. “a”, “the”, “and”
3. Punctuations other than those described in “assumption 4” has no information for our modeling.
4. Special punctuations between letters are meaningful, they are usually part of a phrase instead of meaningless. For example, 12-gallon, 12X12X12 and so on.
5. People may use different words to describe the same thing. Synonym in our text can be replaced by a common word and the content meaning stays the same.
6. Words with the same stem contain same amount of information for modeling, e.g. information of “feet” = information of “foot”, information of “played” = information of “play”
7. Distinct letter cases contain the same amount of information. e.g. info (“Apple”) = info (“apple”)

1.2 Preprocessing Data

With the above assumption, we cleaned our data in the following manners in order to make our modeling process more efficient; 1). Corrected the typo and spelling errors, by leveraging the auto detection of google searching. 2). Deleted English stop words. 3). Deleted punctuation except those mentioned in assumption 4. 4). Replaced Synonym. 5). Turned all words in their stem format and lower cases.

1.3 Statistical Model Employed

We divided our statistical models as two main parts, feature selection and regressor building. In term of feature selection, we refer to the logic of backward selection. We firstly selected as much as predictors (features) and took certain amounts of them out of the model in order to have better model. To start with, we have selected couple common text mining features including counting, distances features, tf-idf features. Since the dataset also contains attributes for products, we added customized features mainly extracted from attributes, including color, brand, materials.

1.3.1 Feature Selection

1. Counting features:

We generate counting features by calculating the total number of words in three selected raw data variables. If people input more words in search item, they probably provide more information about the product they want. For some search terms, it only contains one word, like some ambiguous word ‘car’ or some meaningless stopword ‘to’, ‘or’ (may because of the mistake in raw data collection) . While there

are still many meaningful one-word search term, multi-word search items are more likely to be specific and clear. Therefore the relevance between these search input and product they finally bought should be higher. We chose the following three features; **number of words in product title**, **number of words in search term** and **number of words in brand name of each item**.

Another counting features are the **number of common words between product title and search term** and those **between brand name and search term**. It is expected that when the numbers get larger, the relevance between product and search input becomes higher. In total, there are five counting features.

2. Distance features:

We defined our distance features by Jaccard Coefficient as follow;

$$Distance(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

where A and B denoted as two sets respectively. In our case, we obtained our distance features as

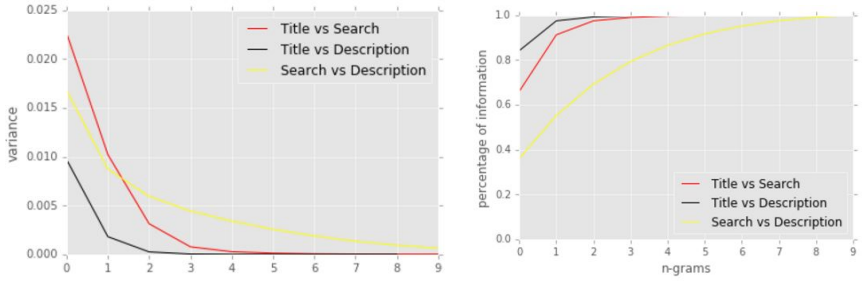
1. Distance(ngram(s,n), ngram(t,n))
2. Distance(ngram(s,n), ngram(d,n))
3. Distance(ngram(t,n), ngram(d,n))
   (Although it seems not very meaningful for our prediction, we have kept the distance between title and description for later statistical inference, and model selection.)

Where "t", "s", "d" stand for set of words in product title, set of words in search term, and set of words in product description respectively. According to Broder, " an *n*-gram is a contiguous sequence of *n* items from a given sequence " (Broder). For example, if a description has the following sentence, "Used for fixing doors", then it's 1-gram set will be ["Used", "for", "fixing", "doors"], while 2-gram set will be ["Used for", "for fixing", "fixing doors"] and so on.

Since most of our text length in search term and product title is smaller than 10, so we obtain mostly zero for our distance features in n larger than 10. The larger n is, the more likely we have zero for our distance features. Thus, we want to have deeper consideration regarding of selecting n for our n-grams.

Selecting N for features selection;

As we noticed, after a specific number of n, our distance features for all sample become zero. Meanwhile, the variance turns into zero. Thus, we choose variance as our indicator of meaningfulness. If the variance of a feature become zero, it implies that the feature has no variance among all samples and contains no additional information. By observing the change of variance along n of n-grams we have the following plot.



Thus, we chose

Distance(ngram(s,1), ngram(t,1)), Distance(ngram(s,2), ngram(t,2)),
Distance(ngram(s,3), ngram(t,3)), Distance(ngram(t,1), ngram(d,1)),
Distance(ngram(t,2), ngram(d,2)), Distance(ngram(s,1), ngram(d,1)),
Distance(ngram(s,2), ngram(d,2))... Distance(ngram(s,8), ngram(d,8)).
Totally 13 predictors as our distance features.
Further inference analysis will be hold in our next step.

3. tf-idf features with cosine similarity:
Tf-idf, short for 'term frequency–inverse document frequency', is important statistics representing the importance of a word to its document. Firstly, TF stands for the term frequency, indicating the frequency of a certain word appears in a document. Since some terms present so frequently in much of the document that make them bring little information useful, we introduce Idf to reweighting these terms. Idf stands for inverse document frequency, measuring the importance of the word included in each document. There are various reweighting schemes to calculate the idf, including unary, probabilistic inverse document frequency and the like. In this case, we opt for the most common weighting scheme that with smooth, $log(1+\frac{N}{n_t})$, in which N represent the number of document in corpus and the $n_t$represents the the number of documents that included term t. Thus the combined statistics Tf-idf is the product of these two separate terms.

Similar to distance features we applied our tf-idf features to n-grams set of words and compare the variance as above and obtained same selection of n.

After we have tf-idf scores for each set of words for search term, product title and product description, we calculate the cosine similarity between each two sets. Thus, we obtained total 13 tfidf features as;
CosSim(tfidf(ngram(s,1), ngram(t,1)))... CosSim(tfidf(ngram(s,3), ngram(t,3))),
CosSim(tfidf(ngram(t,1), ngram(d,1))), CosSim(tfidf(ngram(t,2), ngram(d,2))),
CosSim(tfidf(ngram(s,1), ngram(d,1)))… CosSim(tfidf(ngram(s,8), ngram(d,8)))

4. Customized features:
As most of items have their attributes in data, we choose the most three common attributes among products; **brand, color** and **material**. It's reasonable to compare between search term and these three product attributes. In this case we used same Jaccard coefficient method. The number of grams here is 1, as many items only have one word in their color and material description.

Multiple linear regression
After selecting those features, we used multiple linear regression for our initial models. The multiple linear regression is one of the simplest way to model the relationships between several explanatory variables and the response variable. We could find out which explanatory variable has relevance to the response variable while others not. The main assumptions underneath the multiple linear regression are linear relationship, multivariate normality, no or little multicollinearity, no autocorrelation and homoscedasticity.

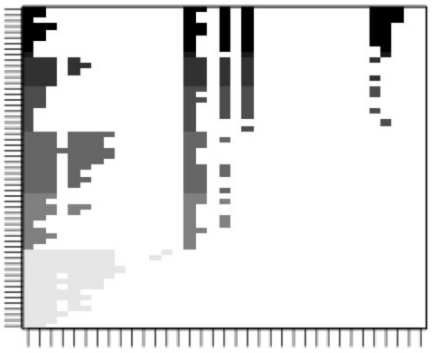
The multiple linear regression is just a crude estimation of the data, while it still ignores a bunch of problems, like multicollinearity among the homogeneous variables. The main reason is that we consider it as a benchmark, basing on that we could dive into much more complicated models, such as Random Forest and Extreme Gradient Boosting. By comparing the output respectively, we could figure out the model that fit the data best.

2. Statistical Analysis/Inference

2.1 Multicollinearity Analysis

We could find that there exist severe correlations among different variables, which is shown in the correlation matrix plot below. Since it seriously violates the assumption of multiple linear regression, we should turn to other methods to undermine the influence. In this case, we used backward selection as an alternative, basing on Mallow's Cp.

the correlation matrix



the backward selection

2.2 Backward Selection

During the backward selection, we started with all the possible variables and then abandoned the variable with the lowest significance to the model at each round. After the backward selection, there remains eight variables, including ts1, ts1_tfidf, ts2_tfidf, td1_tfidf, sd1_tfidf, ProductNum, SearchNum and brandNum, and the corresponding p-values are all close to 0, which are significant under the level of 5%. The R-square is 0.1428, which is quite impressive since we dealing with a huge data set.

```
                Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.294184   0.008835 259.679  < 2e-16 ***
ts1          0.279446   0.035075   7.967 1.65e-15 ***
ts1_tfidf    0.812688   0.024887  32.655  < 2e-16 ***
ts2_tfidf    0.094180   0.022809   4.129 3.65e-05 ***
td1_tfidf   -0.290473   0.014645 -19.834  < 2e-16 ***
sd1_tfidf    0.758982   0.021367  35.521  < 2e-16 ***
ProductNum   0.009230   0.000531  17.382  < 2e-16 ***
SearchNum   -0.061364   0.001586 -38.689  < 2e-16 ***
brandNum    -0.042149   0.002762 -15.263  < 2e-16 ***
```

However, the backward selection method has its own disadvantage, it won't guarantee to generate a subset of the variables that best fit the data. As a result, we should find a more sophisticated method to deal with multicollinearity.

3. Model Improvement

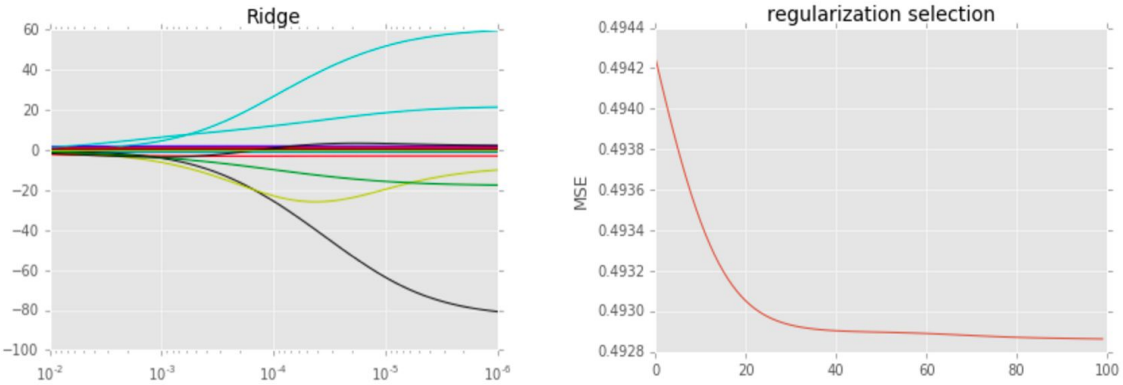
After having the result from multiple linear regression , we decided to try out different models to solve this problem, like ridge regression, machine learning method and neural network. However, neural network doesn't have good performance for this question. One of the reason can be the dataset is not large enough and our training time is relatively short. We decided to omit this part in the report. Random Forest and Boosting are the selected ML model, as they are well-known ensemble learning methods for these kinds of problem. In order to compare and monitor the performance of these feature selection methods and regression models. We used average RMSE obtained from 5-folds cross validation of each combination of features and regression models as our measurement.

3.1 Ridge Regression

One way to revise the multicollinearity problem is through the ridge regression. It is an elegant technique to address this problem by adding a regularization of 2-norm. In this way, the least square estimation would not be unbiased anymore, but it greatly reduces the variance of the model which would otherwise result in high bias from the true values. Since this model incorporates a regularization parameter and there are no specific rules to choose an optimal value for the model, we should take both bias and variance into considerations.

Selecting the regularization parameter:

As described above, we take variance as our criterion to find the best parameters of the model. After taking a close scan to the output according to the regularization parameter, we decide to choose it as 1.63e-5, around the 30th point indicated in the right panel. Though at this point, the model would not achieve the smallest MSE, it is good enough since there is merely difference compared to the minimum. In addition, it would be shortsighted to take MSE as the only standard when choosing the optimal point, it could end up with overfitting.



3.2 Random Forest

Random forest is a substantial modification that builds a large collection of de-correlated trees and then averages them. It is a popular ensemble selection method as the performance of random forest is very similar to boosting and corresponding parameters are simpler to train. We choose to tune two parameters in this model, the number of trees in forest and maximum depth of each tree. Grid search method is applied to find the optimal parameters pair. The search range for parameters are

- Number of trees in forest: 10-35
- Maximum depth of tree: 5-15

For each possible pair of parameters in the specified search range, we conduct 5-fold cross-validation and record corresponding average RMSE. The optimal pair, 32 trees in forest and maximum 10 tree nodes, is chosen with the smallest average RMSE.

3.3 Extreme Gradient Boosting

XGBoost, short for 'Extreme Gradient Boosting', is also an ensemble model for supervised learning problem. It applies a more regularized model formalization to control overfitting, which usually gives a better performance. Similar to the previous method implemented in random forest, we also decide to tune two parameters, number of boosted trees and number of tree depth for base learners. Grid search is also used in this parameter optimization process. The search range for parameters are

- Number of tree depth for base learners: 1-15
- Number of boosted trees to fit: 25-40

We also conduct 5-fold cross-validation to every possible pair of parameters in the above range. For each pair we calculate the average RMSE from the validation step. The result shows the optimal pair is, 33 boosted trees and maximum 5 tree nodes.

3.4 Performance Comparison

We have the following comparison for our models and features;

Feature\Regressor	Linear Regression	Ridge Regression	RF	XGB
Customized	0.5329	0.5329	0.5322	0.5323
Count	0.5211	0.5211	0.5143	0.5154
Distance	0.5133	0.5132	0.5100	0.5232
Ti-idf	0.5006	0.5006	0.4968	0.4983
All	0.4944	0.4929	0.4829	0.4829

The value in the chart is RMSE. Linear Regression model above for each feature applied backward selection separately for modeling building. From above, we can see the best model is XGBoost and Random forest. Linear model performs the worst as many variables are correlated with each other. Also we can see Tf-idf is the most important feature among all the variables.

4. Conclusion

In conclusion, tf-idf feature contributes most to the prediction accuracy improvement, no matter which model we used. This is not surprising, as this method quantifies the importance of a word in the document

and is offset by its frequency in the corpus. Many of our prediction variables are highly correlated and exist multicollinearity. By comparing the results from four different models, we can see that random forest performs the best among four models. This is probably because it decorrelates the members of its ensemble, i.e. decision trees, and averages these uncorrelated members to make the prediction more reliable and less variable. Linear model gives an unsatisfactory result as many of its assumptions, like little multicollinearity and linear relation, are violated. These prerequisites are the things we always need to care about when we conduct a further statistical analysis. In the future, the model can be improved by adding more significant features from product attributes.

5. Appendix

Contribution Statement

We shared most of works together and the following part indicated the main contribution of a team member.

Kewen Zhang: Data Cleaning, Count features, Customized features, Random Forest, Extreme Boosting, Final Report

Pengfei Wang: Tf-idf features, Multiple Linear Regression, Ridge Regression, Final Report

Xiaoci Xing: Code Reviewing, Topic Interpreting, Final Report

Ziyue Wu: Data Cleaning, Distance features, Tf-idf features, Neural Network, Presentation, Final Report

Our Codes and Works

https://github.com/Zac2116/TextMining

Reference

1. www.kaggle.com
2. Broder, Andrei Z.; Glassman, Steven C.; Manasse, Mark S.; Zweig, Geoffrey (1997). "Syntactic clustering of the web". *Computer Networks and ISDN Systems* **29** (8): 1157–1166. doi:10.1016/s0169-7552(97)00031-7.
3. https://github.com/tensorflow/skflow
4. https://github.com/ChenglongChen/Kaggle CrowdFlower
5. https://github.com/dmlc/xgboost