Home Depot Product Search Relevance

[Kaggle Competition] [*]

Hao Mao
Northeastern University
Seattle, WA
mao.ha@husky.neu.edu

Wanting Jiang
Northeastern University
Seattle, WA
jiang.wa@husky.neu.edu

Mingzhe Xu
Northeastern University
Seattle, WA
xu.ming@husky.neu.edu

ABSTRACT

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. In this paper, after comparing with other models, we are going to apply Random Forest to predict the relevancy between search queries and the products in a gauge of number 1 to 3 in an effort to make accurate and efficient prediction.

Keywords

Search Relevance; Nature Language Processing; Home Depot; Random Forest; Stemming

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. From installing a new ceiling fan to remodeling an entire kitchen, with the click of a mouse or tap of the screen, customers expect the correct results to their queries quickly. Speed, accuracy and delivering a frictionless customer experience are essential.

Our goal is to develop a model that can improve Home Depot's customers' shopping experience and accurately predict the relevance of search results and products. Based on the data from Home Depot, we are going to predict a relevance score for the provided combinations of search terms and products. We have applied a couple of tool kits for our data processing and model application, such as Snowball-Stemmer, Sci-kit Learn, Pandas etc..This project contains three parts: data preprocessing, which includes cleaning, merging, stemming, feature selection and adding new attributes, application of Random Forest model, and the evaluation of the models based upon Root Mean Squared Error.

This paper is structured as follow. First, the original datasets we adopted from Kaggle is described and explained, which is followed by how the data was preprocessed so as to generate features that can be used in models including stemming and generating new features. Then, there will be a discussion of choosing Random Forest over other models and how it works. Finally the preliminary empirical result will be displayed and elaborated.

[*]This competition can be found at this site: `https://www.kaggle.com/c/home-depot-product-search-relevance`

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

2. DATASETS

Datasets for this project are from Home Depot's website which contains a number of products and real customer search terms. Each pair in the dataset was evaluated by at least three human raters. The rate values are from 1 to 3, where 3 is the most relevant, 2 somewhat relevant and 1 irrelevant. The provided relevance scores are the average values of the ratings. We adopt the data from Kaggle website including mainly the training data and testing data.

2.1 Datasets Description

Here are the original datasets provided by Kaggle.com:

Table 1: Datasets Description

Dataset Name	Dataset Description
train.csv	the training set, contains products, searches, and relevance scores
product_descriptions.csv	contains a text description of each product
attributes.csv	provides extended information about a subset of the products (typically representing detailed technical specifications). Not every product will have attributes.
test.csv	the test set, contains products and searches. (We will provide our prediction of the relevance of those pairs)

We merge the train.csv file and product_descriptions.csv file as our new training data so as to get the concrete description for the product. The attributes are also merged into the training data so as to make the most of the given data. After merging all the data into one dataset, we can preprocess the data into features that are usable for modeling.

2.2 Data Field Description

Table 2: Data Field Description

Field Name	Field Description
id	a unique Id field which represents a (search_term, product_uid) pair
product_uid	an id for the products
product_title	name of the product
product_description	the text description of the product (may contain HTML content)
search_term	the search query in text form
relevance	the average of the relevance ratings for a given id
name	name of the attribute
value	value of the attribute

The training data and product description data also provide the above data fields which can be later applied as features. However, features such as product description or search term can not be directly used and also have potential to be restructured to generate new features. So in bellow preprocessing part we will elaborate on how we manipulate the original data to generate features for our model.

3. ALGORITHMS

3.1 Preprocessing

There are two kinds of attributes in the dataset. The first one is some common attributes, which all products have that, such as title and product description. Another one is extended information which means not every product will have these attributes, such as Brand and Length.

Table 3: List of Features

Common Attributes	Extended Attributes
Title	Brand
Production Description	Material
	Length
	Weight
	Width
	Search_Term
	Bullets
	Bullet01
	Bullet02
	Bullet03
	Bullet04
	...

All common attributes are considered important here, we will use all of them in our predict model.

Some extended attributes are also very useful in calculating search relevance, such as Brand, Material and Functionality, which are as important as the Title. We will focus on these attributes.

Other extended attributes like Length, Bullet01, which might be helpful in our calculation. But it is hard to decide which one should be used in the model, so in our experiment, we will test several combination of the attributes, and choose the best as our final selected attributes.

3.1.1 Data Cleaning

The given raw data contains various forms of words and the difference in the style make is hard to give accurate prediction. Therefore, we first clean up the data with the following approaches. First, for all the capitalized or uppercase words/letters, we lowercase them so that the data is robust to the case sensitiveness. Second, we remove punctuations. Punctuation is helpful when we read, but in models it is hard to incorporate them without affecting the accuracy. Third, we digitalize all the quantities such as one oz transferred to 1oz. Fourth, units are unified into abbreviations, such as 5 inches transferred into 5in.. Fifth, we corrected mis-spelt words, such as 'correet' was updated to 'correct'.

3.1.2 Data Stemming

In order to improve the precision of the result, we will use Snowball to do stemming on the selected attributes and queries.[3] Other than applying given Snowball stemming tool to do basic stemming, we also explore the data and add more stemming methods in our preprocessing stage. There are many units shown in the data such as unit of length, unit of weight, unit of space, etc. We unified the various impressions into one. For example, "inches", "in" or "inch" are all stemmed into "in."; "pound", "pounds", "lbs" and "lb" are stemmed into "lb.". We also eliminate punctuations and redundant space in the text.

3.1.3 Data Normalizations

Instead of simply assigning same weight to all the words in the attributes, we applied tf-idf to normalize data fields that are consists of words. We calculate the term frequency-inverse document frequency ratio of a word in search query or product description.[10, 9] This feature is very useful in terms of the importance or the weight of a word in the text.

3.1.4 Feature Preparation

For the purpose of getting the importance of the word in the attributes, we calculate the ratio of each word in each attribute, based on the length of the query. In addition to the original features, we generated new features based on the original data. We generate bullets feature which combined all bullets of a product into one. We also combined search term with product title into a new feature called product info.

3.2 Modelling

With features prepared for the model, we then compare different models to apply based on their pros and cons. Random forest is better at feature selection given that we have many features to choose from.[2] It is also very robust against overfitting.[4, 5] Naive Bayes is easy to train and understand. However, it based on the "naive" assumption that all variables are uncorrelated with each other. What's more it is fragile to overfitting compared with Random Forest. Regarding logistic regression, it is hard for it to handle features that are non-linear. What's more, Random Forest is very powerful when handling large dataset with high dimensions.[6, 7] Therefore we choose Random Forests as our core method.

3.2.1 Model Selection

There are many features in products and items and we are not sure which features play important on search result relevancy. Also this project already provides us some training results. So we can use Random Forest model with current training data to predict the relevance for each pair listed in the test set by trying different feature sets. Given the Random Forest model, it is easier for us to pick the best feature set and an accurate predicting model.

3.2.2 Random Forest

Random forests is a notion of the general technique of random decision forests that are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

3.2.3 Mechanism of Random Forest

When the training set for the current tree is drawn by sampling with replacement, about one-third of the cases are left out of the sample. This OOB (out-of-bag) data is used to get a running unbiased estimate of the classification error as trees are added to the forest. It is also used to get estimates of variable importance.

After each tree is built, all of the data are run down the tree, and proximities are computed for each pair of cases. If two cases occupy the same terminal node, their proximity is increased by one. At the end of the run, the proximities are normalized by dividing by the number of trees. Proximities are used in replacing missing data, locating outliers, and producing illuminating low-dimensional views of the data.

3.2.4 Application of Random Forest Model

First, pick a feature set that contains relevant features. Here we choose 'id', 'relevance', 'search_term', 'product_title', 'product_description', 'product_info','attr', 'brand', 'bullets', 'bullet1', 'bullet2', 'bullet3', 'bullet4', 'material'. Second, we build a Random Forest Model based on this feature set. However, we did not simply put those features into the model. Pipeline is used to transform some of our key features. For "search_term", "product_title", and "brand", we calculate the tf-idf and truncated the tf-idf matrix into one-dimensional feature by using singular value decomposition (SVD) so as to increase the accuracy and improve the efficiency. We then union the features into the data, and prepare the features thoroughly for the model.

Then, we apply grid search for optimizing hyper-parameters. Instead of using randomized search, grid search is slightly slower, yet it is more accurate and has better performance than randomized search.[8] After setting up the estimator and parameters for grid search, we run fit the model with our training data and predict the parameters for our model.

Finally, After fitting the training data and get the best params and best score, we calculate the Root Mean Squared Error (RMSE) evaluation for this model to gauge the model. We also submitted our result to Leaderboard and got our first score on Kaggle Leaderboard which beats the benchmarks and better than hundreds other submissions.

4. EVALUATION METHOD

The evaluation we use, or based on Kaggle's evaluating standards, RMSE is used, which is the square root of the mean/average of the square of all of the error. The use of RMSE is very common and it makes an excellent general purpose error metric for numerical predictions. [11] Compared to the similar Mean Absolute Error, RMSE amplifies and severely punishes large errors.

$$RMSE \quad = \quad \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\mathbf{y}_i - \hat{\mathbf{y}}_i)}$$

In python, RMSE = mean_squared_error(y, y_pred)**0.5. Kaggle uses this number as the score reference in it's Leaderboard.

5. EMPIRICAL RESULT

We've conducted five sets of experiment with random forest with different set of features and our rank in leaderboard varies for each submission. These three experiments are called easy forest, complex forest with less features, and complex forest with more features.

In the first experiment, we incorporate features from original data without any modification such as cleaning or stemming. This model yields a RMSE of 0.48721. In the second experiment, we modify the features by cleaning them and processed with stemming. Based on the data we acquired after the above cleaning, stemming, and normalizing, there are 11 features selected to build a satisfying model. Those features we included are id, product_title, product_uid, relevance, search_term, product_description, brand, product_info, len_of_query, word_in_title, word_in_description. The score has improved by 0.003, which is 0.48462. However, none of these two experiments adopted data transformation, which is a deep modification of the original data.

In our third experiment, we generated a bunch of new features. The features we've adopted are id, product_title, product_uid, relevance, search_term, product_description, brand, product_info, len_of_query, word_in_title, word_in_description, brand_feature, and search_term_feature. In total of 13 features. However, we used tf-idf on search_term, product_title and brand and generated new features for this model. In the meantime, some of the features, such as id, relevance, brand, search_term, product_description, product_title, and product_info, after being transformed or found less significant, are discarded. In this experiment, the score is 0.47939, which is higher than the previous experiment by 0.005.

6. CONCLUSION

7. REFERENCES

[1] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome. *The Elements of Statistical Learning*. Springer, 2001.

[2] Leo Breiman. *Random Forests*. Machine Learning, October 2001, Volume 45, Issue 1, pp 5-32.

[3] SnowBall: `http://snowball.tartarus.org/`.

[4] Random Forest Wikipedia, `https://en.wikipedia.org/wiki/Random_forest`

[5] Liaw, Andy, and Matthew Wiener. "Classification and regression by randomForest." R news 2.3 (2002): 18-22.

[6] Ho, Tin Kam (1995). Random Decision Forests (PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14-16 August 1995. pp. 278-282

[7] Breiman, Leo (2001). "Random Forests". Machine Learning 45 (1): 5-32.

[8] Scikit-Learn: Comparing randomized search and grid search for hyperparameter estimation: http://scikit-learn.org/stable/auto_examples/model_selection/randomized_search.html

[9] Ramos, Juan. "Using tf-idf to determine word relevance in document queries." Proceedings of the first instructional conference on machine learning. 2003.

[10] Joachims, Thorsten. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. No. CMU-CS-96-118. Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.

[11] Kaggle Competition: https://www.kaggle.com/wiki/RootMeanSquaredError