Home Depot Query-Product Relevance Score Prediction

Project Final Report

Yu Wen, Ehsan RahimiNasab, Steven Xu

Problem Description:

In Kaggle website, there is a competition hosted by Home Depot Company which its underlying problem is designing a model that helps them to improve their search engine by ranking pairs of search queries and search results based on their relevancy. More specifically, in our problem, data is given in the form of (q, p, s), where q is the query, p is the product returned by the search algorithm and s is the relevance score of the pair (q, p) rated by a human rater. After a model have been successfully trained, the input of this model would be some pair like (q, p) and the output will be s which is considered as an automatically generated relevance score.

Input examples:

Here are two input examples for our system:

id	Product_uid	Product_title	Product_description	Search_term
210	100037	Husky 9-Pocket Maintenance Pouch	Made from 100% recycled quality post consumer HDPE plastic, the Lifetime Professional 40 ft. …	husky tool bag
549	100090	LG Electronics 5,000 BTU 115-Volt Window Air Conditioner	…LG window air conditioners come with multiple innovative features aimed at making your life cooler. Help keep your room cool with the LG Electronics 5,000 BTU Window Air Conditioner (LW5015). The air conditioner can cool a room of up to 150 sq. ft. …….	A/c window unit

Additional information for each product in the database are gathered as product attributes. Each entry may have 1 to 6 or more attributes.

Product_uid	Material	MFG Brand Name	...
100037	Fabric	Husky	...

Product_uid	Automatic shutoff	Product depth (in.)	...
100090	No	14.38	...

Output Examples:

The corresponding output example for the above inputs would be:

id	relevance
210	2.5
549	3

Related Work:

Ranking is a central part of many information retrieval problems, such as document retrieval, collaborative filtering, sentiment analysis, and online advertising. One of them which is the underlying problem of our project is designing a model for query-document pairs where each pair has a relevancy score. In this problem training data consists of queries and documents matching them together with relevance degree of each match. It may be prepared manually by human assessors (Like Home Depot), who check results for some queries and determine relevance of each result. It is not feasible to check relevance of all documents, and so typically a technique called pooling issued, only the top few documents, retrieved by some existing ranking models are checked. Alternatively, training data may be derived automatically by analyzing click-through logs (i.e. search results which got clicks from users), query chains, or such search engines' features as Google's SearchWiki. There are several approaches toward this problem, the one that is relevant to our goal is called Pointwise approach in which it is assumed that each query-document pair in the training data has a numerical or ordinal score. Then learning-to-rank problem can be approximated by a regression problem –given a single query-document pair, predict its score. A number of existing supervised machine learning algorithms can be readily used for this purpose. Ordinal regression and classification algorithms can also be used in pointwise approach when they are used to predict score of a single query-document pair, and it takes a small, finite number of values.

Other approaches toward this problem are Pairwise approach and List-wise approach which in the first one, learning-to-rank problem is approximated by a classification problem – learning a binary classifier that can tell which document is better in a given pair of documents, and the other one tries to directly optimize the value of one of the evaluation measures, averaged over all queries in the training data. This is difficult because most evaluation measures are not continuous functions with respect to ranking model's parameters, and so continuous approximations or bounds on evaluation measures have to be used.

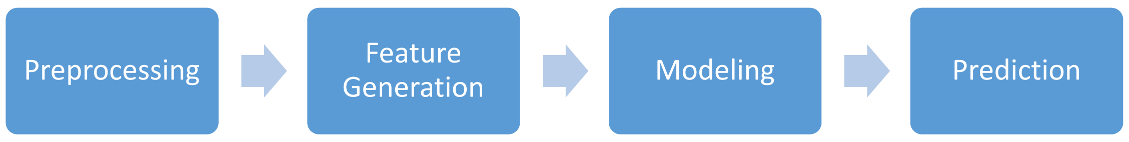
A partial list of published learning-to-rank algorithms with pointwise approach, is shown below with years of first publication of each method:

Year	Name	Notes
1989	OPRF	Polynomial Regression
1992	SLR	Staged logistic regression
2002	Pranking	Ordinal regression
2007	McRank	
2010	CRR	

We have used various models in order to give a solution to this problem which were not discussed in the above examples of pointwise approach.

Methodology:

The steps that we have followed in this project can be summarized in the following flow diagram:



The description of each step is as follows:

Preprocessing:

1. **Typo Correction:**
   It is obvious that the query terms can have several typos as they are entered by a human user. In order to be able to compute a true value of tf-idf, we had to correct the spelling of the words in query terms.
2. **Remove Punctuation.**
3. **CamelCase Split:**
   in order to increase the precision, we split the camel-case words like “DeckOver” to “Deck Over”.
4. **Non-English Characters Remove:**
   we did this to reduce the size of the vocabulary.
5. **Remove Stop Words.**
6. **Converting to lower-case.**
7. **Stemming the words:**
   Mapping all derivations of a word to its root, like (“fishing”,”fished”,”fish”,”fisher” -> “fish”).

Feature Generation:

In order to prepare the data to be capable of a model fit, we had to map each pair in the dataset into a feature vector. The features that first came to our mind were:

1. Number of words (in query or in product)
2. Number of characters (in query or in product)
3. Tf-idf score for a pair of query and product.

Since these 3 features cannot represent a good stem of our data set, we extended the number of features by dividing each product information to 5 pieces, which were product_title, product_description, product_bullet_information, product_brand_name, and product_other_information, where the last three were gathered form the product_attributes dataset. Then we recomputed the above three features for each piece which yielded to 17 features.

An example of a tf-idf matrix would be:

	deck	aluminum	rectangle	porch	stand	metal	water	…
(q1,p1)	0.3	0.78	0.65	0	0	0	0	…
(q2,p2)	0	0	0	0.45	0.2	0.4	0	…

Where the columns are the words in the vocabulary, and the rows are the pairs of queries and products. Then if we sum each row, we have the tf-idf scores for each pair.

As results were not satisfying with this number of features we generated more features by means of truncated SVD. The multiplication of the right truncated singular vectors with the tf-idf matrix, generated 50 features for each of the above pieces. Hence at last we came up to 238 features in total for each pair of query and product.

To be more specific, assume A is the tf-idf matrix similar to the matrix above. Then if we use SVD decomposition, this A can be decomposed as $A = U\Sigma V^T$, where the rows of V are corresponding to the words in the vocabulary, and its columns corresponds to several concepts in the corpus of products (for example, a concept can be regarded as categories based on the products usages. Now if we truncate V and only keep the most significant singular vectors of it (say 50), then we have V^* with number of rows equal to the number of words in vocabulary, and 50 columns where the columns are the most important categories among the products. Now if we compute $A \times V$, the result matrix will assign 50 numerics to each (q, p), pair where they indicate how much each pair is involved in each of those categories.

Modeling

We have tried the following methods in order to predict the relevance score.

SVM Regression

SVM is a kind of maximum margin model, which has a very high accuracy. SVM can also be applied for regression with modification of hypothesis function and loss function. We applied the SVM Regression Model with C=1, epsilon=0.1 and kernel of Gaussian kernel.

Linear Regression

We think it is possible that relevance score could follow a Gaussian distribution with mean equals to the linear combination of the features we have generated.

KNN Regression

KNN model has rich hypothesis space since no assumptions about the decision boundary are made. KNN could also be applied to this regression problem we have by calculating the average of the relevance scores of the K nearest neighbors. We used the Euclidian distance when we applied the KNN regression.

Feedforward Neural Network

Neural network has excellent capability of approximating complex functions and rich hypotheses space. Since it's difficult to make assumptions about the relationship between the feature we have generated and the relevance score, we expected that this model could gave us a good result. We chose the topology of the network with 20 hidden layers and 20 nodes in each hidden layer. The activation function we chose was hyperbolic tangent. In the output layer, we only used one output node of rectified linear unit to do the task of regression. We trained the model by iterating 50 epoch.

Random Forest with Bagging

Since random forest is robust to outliers and errors, we expected that this model could achieve a good result since the "true label" in our supervised dataset is given by human raters, which is subjective and has a high probability of being errors. In order to further reducing this subjective influence, we also applied the Bagging along with the Random Forest. During each training of single predictor, we create a boot strap sample by sampling a subset of training data to train this single predictor, which we expected to be useful to reduce the influence of training error. We created 15 trees for each random forest and 45 forests for bagging.

XGBoosting

XGBoosting is an accurate and effective off-the-shelf procedure that can be used for both regression and classification problems. Compared to bagging ensemble strategy, XGBoosting will train estimators sequentially and the new added estimator will focus on correcting the incorrect predictions that previous estimators make. It will make XGBoosting usually provides more precise prediction results than Random Forest. Since XGBoosting has the advantage of handling mixed type data and features that we extracted do have different types, it will be reasonable to make use of XGBoosting Regression model to predict relevance. In the project, we have several different features, like tf-idf, number of word, tvsd and number of characters. But XGBoosting also has the disadvantage that is lack of scalability when it is compared to Random Forest Regression. In this project, we set the attribute warm_start = True, in order to reuse the solution of previous call; created 45 random decision trees with depth as 6.

Experiments

Datasets

The original supervised dataset from Kaggle contains about 74,000 examples, which leads to a very slow training speed for our modeling. As a result, we reduced the size by sampling 15% samples from the original dataset. Finally, the training set we have used containing 8,800 examples; the test set we have used containing about 2,200 examples. Note that the relevance score in the supervised dataset is the average of scores given by 3 human raters.

Evaluation Metrics

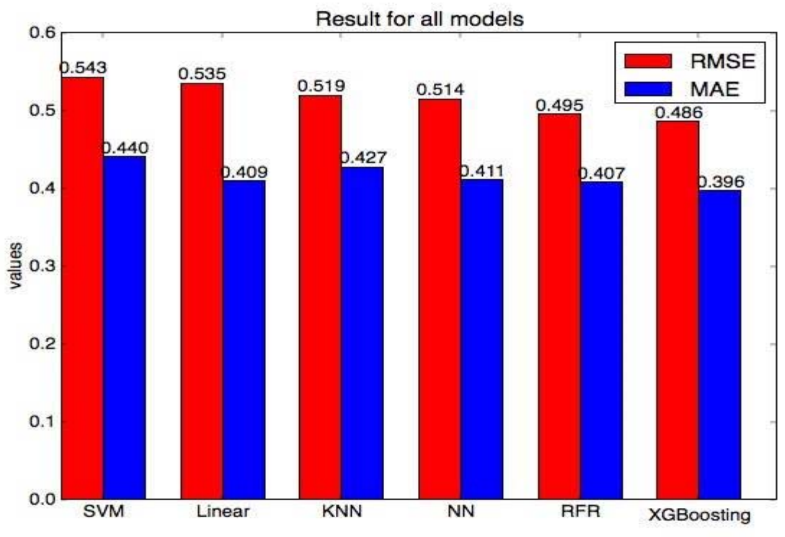
We have used Rooted Mean Square Error, $\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(y_i - y'_i)^2}$, and Mean Absolute Error,

$\mathrm{MAE} = \frac{1}{n}\sum_{i=0}^{n}|y_i - y'_i|$. Note that RMSE is sensitive to large deviations from the true score.

Baseline

We generated the baseline by randomly guess a score for all test examples from 1 to 3. As a result, RMSE =0.843 and MAE =0.6943.

Results



Above are the results we have acquired. From these results, we can conclude that the ensemble methods which are more robust to outliers and errors gained better performance. The reason is that the supervised dataset has subjective scores and contain many errors.