Kaggle Machine Learning Competition: Predicting Relevance of Search Results on HomeDepot.com

Everton Lima

21 June 2016

Contents

Overview

Search relevance is a prevalent technical challenge faced by many online stores, and is quintessential for offering an accurate, and fast service. This technical challenge brought me to the Home Depot Kaggle competition, which ran from 18th of January to the 25th of April of 2016[1]. This competition presented a great learning opportunity. The objective is to help HomeDepot.com improve their search by predicting the relevance score of product and query pairs. For the company this is a significant shift, as they currently rely on human raters to score the relevance of queries and product pairs. Clearly, a costly and time consuming task. The objective of this study is to obtain a model that produces a search relevance value that closely approximates the value given by the human raters. In the upcoming sections the necessary steps for obtaining this model are presented. The first section gives a more in depth overview of the data made available by Home Depot at Kaggle. The following section discussing the preprocessing steps that were taken. Following this the feature extraction is discussed. The next focuses in exploring several models in order to choose the one best suited for the problem at hand. All together, this challenge presented a great learning opportunity to deepen my knowledge of machine learning, the result of which is shown in the coming sections.

Data

The data made available was the following. A training set, containing product titles, queries and relevance scores (The first 5 entries can be observed in Table 1). A test set used for evaluation of the algorithms. Product descriptions, which contains a text description of each product. Additionally, data with product attributes was also made available which expanded on the product descriptions (for some products). Lastly, Home Depot also made available the instructions supplied to the human raters when evaluating search relevance.

Table 1: First 5 entries in the train data.

product_title	search_term	relevance
Simpson Strong-Tie 12-Gauge Angle	angle bracket	3.00
Simpson Strong-Tie 12-Gauge Angle	l bracket	2.50
BEHR Premium Textured DeckOver 1-gal. #SC-141 Tugboat Wood and Concrete Coating	deck over	3.00
Delta Vero 1-Handle Shower Only Faucet Trim Kit in Chrome (Valve Not Included)	rain shower head	2.33
Delta Vero 1-Handle Shower Only Faucet Trim Kit in Chrome (Valve Not Included)	shower only faucet	2.67
Whirlpool 1.9 cu. ft. Over the Range Convection Microwave in Stainless Steel with Sensor Cooking	convection otr	3.00

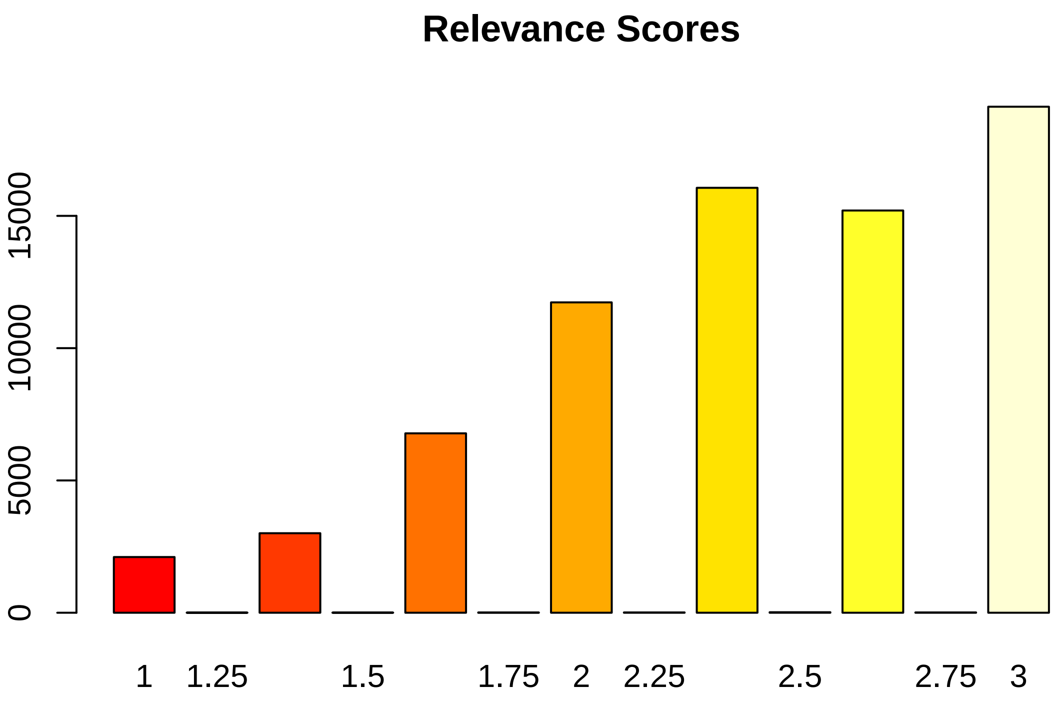
The instructions provided to human raters go into detail about score assignment, and the data provided to the raters. This document states that each rater was was asked to submit a score of either 1, 2 or 3 indicating the relevance of the query and product pair. A score of 1 indicates that it is irrelevant, 2 shows that it is somewhat relevant or mostly relevant, and 3 is a perfect match (a breakdown of the assigned scores can be observed in Table 2). Some specific instructions were also provided for when the rater should chose each value. To summarize, higher scores should be awarded when the product query pair match on the following; product type, brand, color, and dimensions. However, some problems can be observed from these instructions. From the supplied instructions, it is not completely clear if the raters had the product descriptions available at the time of the rating or had to navigate to a product page which may have affected the scoring. Also, product images were made available to the raters and were not provided by Home Depot. The relevance

[1]More information about this competition can be obtained at https://www.kaggle.com/c/home-depot-product-search-relevance.

score was also averaged among all the raters, and the original score was not made available.

Table 2: Summary of the train data.

product_uid	product_title	search_term	relevance
Min. :100001	Length:74067	Length:74067	Min. :1.000
1st Qu.:115128	Class :character	Class :character	1st Qu.:2.000
Median :137334	Mode :character	Mode :character	Median :2.330
Mean :142332			Mean :2.382
3rd Qu.:166884			3rd Qu.:3.000
Max. :206650			Max. :3.000



The next section presents the preprocessing steps that were taken.

Preprocessing

This section provides a summary of the teleprocessing steps that were taken. First, all non-alphanumeric symbols were removed from all entries in the training data. Following this step, the content was encoded in UTF-8. The training data was then mapped to lowercase. Stop words were also removed[ˆstop words]. The following step removed the punctuation.The next step performed spelling correction by making use of the popular Hunspell tool. Finally, as the last step the terms present in the training data were stemmed. Some features that made use of synonyms were explored however these were later excluded since they possessed a very high correlation and did not improve the model.

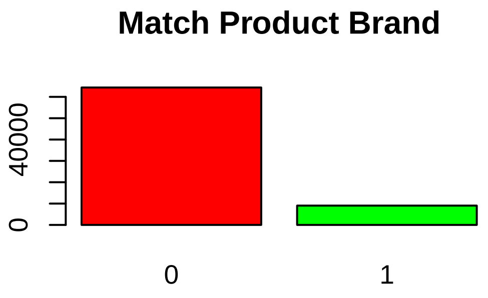
[ˆstop words]: List of stop words was obtained from the TM package. For this step the list of SMART words was used. Alternatively this list of words can be retrieved from http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop.

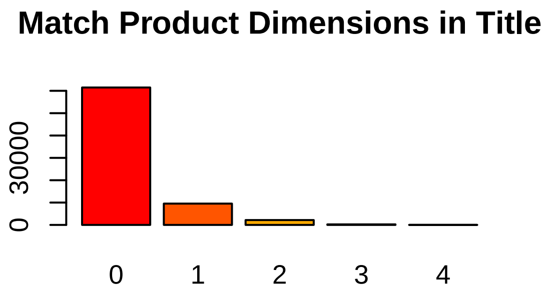
Feature Extraction

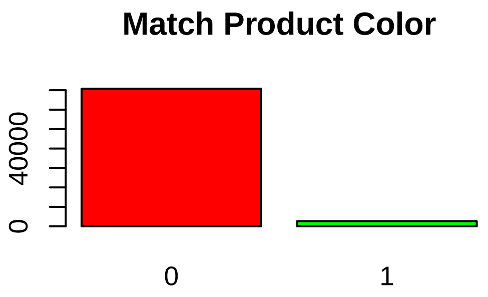
This section presents the intuition behind each feature extracted. The first section discussed basic features, such as matching of product brand in the query and product pair. The following sections extract information retrieval functions such as term frequency. Moreover, the cosine distance of word embedding is also discussed.

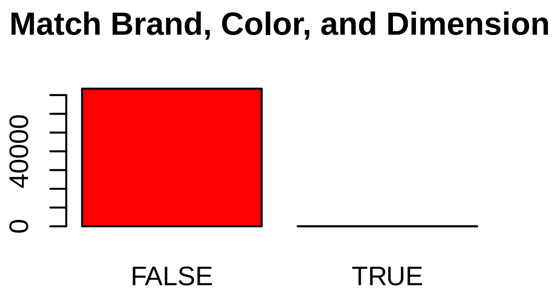
Basic Features

In this sections the features that relate to the instructions received by raters are discussed. These features are the following. Matching of query and product brand, matching of query and product dimensions, and matching of query and product color. The brand name for each product was obtained from the attributes data.This was later joined with the training data. In this manner one can easily retrieve the brand for each of the query product pairs. This results in a feature that indicates if there was a match or not (value of either 1 or 0). Similar approach was taken for product dimensions (for both product title and description), and color. However, product dimensions sums each dimension that match, instead indicating occurrence. Below you can observe a few box plots that represent which portions of the training data contain these features.









The figures above show that a significant portion of the data is not captured by any of these features. This is also true when all these features are combined (as can be seen from the rightmost plot in the bottom).

Table 3: Coefficients obtained when Regressing Relevance onto Brand, Dimension, and Color.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.3688305	0.0022888	1034.986802	0.0000000
score.brand_cover	0.0689619	0.0059736	11.544453	0.0000000
score.title_dimension	0.0508055	0.0062304	8.154433	0.0000000
score.description_dimension	-0.0194198	0.0070897	-2.739142	0.0061615
score.color_cover	0.0122833	0.0105326	1.166217	0.2435305

While a majority of the data is not captured by these features, training a linear regression model in this group of features yields the table of coefficients above. This model has a F-statistic of 55, showing that there is an association between the predictors and relevance. However, from the table one can note that color is not observed as significant at 95% confidence value. One should note that due to the size of the dataset many features are likely to appear related even when they aren't (in a smaller subset of the data), since there is a significant decrease in variance. However, significance is not of interest since this is a prediction problem and all the features improve prediction, even if marginally. Additionally, examining these features closely shows that the matching of title and query dimensions is very high correlated with matching product description dimensions. The second being slightly superfluous in the presence of the first.

Interestingly, although raters were directed to note on the occurrence of these features they present only a small portion of the training data. It seems that many user queries do not contain any brand, color, or dimensions and simply query for a type of product. HomeDepot customers seem much more likely to be interested in the type of product that suits their needs, rather than products for specific brands. Only 30 observations contain a match for brand name, at least one product dimension and color. This goes to show that a good feature for this problem should explore other characteristics such as product type rather than its attributes. In the coming section an additional group of features is explored that aims to achieve this.

Term Distance, Jaccard Distance, and Ratio

This subsection explores a group of distance features. Namely, query term distance, Jaccard distance, and query term ratio. Each of these features is explored in the aim that matching query terms may lead a better generalization than focusing in product attributes as was shown in the previously.

The term distance features indicates the smallest window that contain all query terms in the product description. Thus, a small values shows that the query terms occur close together and a large value show that there are many terms between all search terms. A higher than possible value was used to indicate the case where not all the query terms occurred.

Table 4: Coefficients obtained when Regressing Relevance onto Distance, Jaccard Distance, and Ratio.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.2610501	0.0075911	297.85724	0
score.distance	-0.0066936	0.0002154	-31.07454	0
score.title_jaccard	0.9464954	0.0152710	61.97990	0
score.description_jaccard	-0.3580107	0.0310627	-11.52541	0
score.description_ratio	0.1408337	0.0048517	29.02799	0

Jaccard distance is a well known measure of similarity between two documents. To put it simply, Jaccard distances is the number of matching terms, divided by the number of unique terms in both texts. The ratio measure is the same, however since Jaccard distance may give a big penalty for long texts ratio was also used as measure for matching search term and product descriptions. This measure divides the set intersection by the number of unique terms in the query only. As such, the term ratio measure was only used for the product description. The table above shows the coefficients obtained by regressing relevance onto these three features. Additionally, the jaccard feature seems to provide a greater prediction power when comparing to the discussed features as be noted by a higher adjusted R^2.

The next section continues to explore a group of features that aim to closely match the product and query pairs, in order to product a more general feature that is able to correctly match product type.

Term Frequency - Inverse Document Frequency.

In this section another well known information retrieval feature is explored. Namely, term frequency - inverse document frequency (TF-IDF). This feature is extracted by building a term document matrix using each product description. From this matrix a each term is weighted according to a weighting scheme. For this task three weighting schemes are used. Namely, term frequency, log term frequency, and the complete form of term frequency - inverse document frequency[2].

The scoring is obtained by calculating the vector inner product of the terms in the search and the document. These are then treated in several different ways. They are averaged, summed, the minimum value is taken, or the maximum value is taken. This results in a set of 18 features.

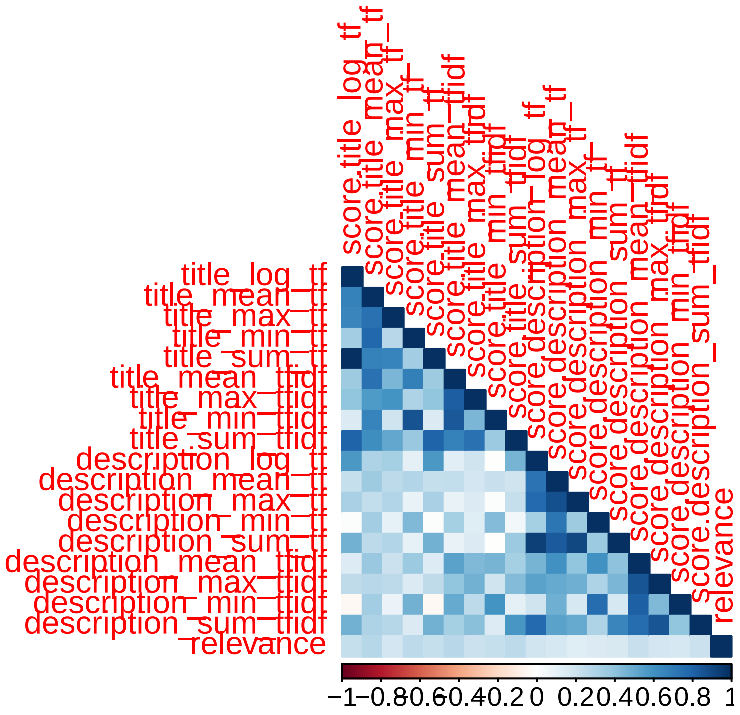


Figure 1: Heatmap of Tf-Idf Correlations.

Above you can observe a correlation heat map for this set of features. One can note from this picture that there isn't a significant correlation between many predictors and the target variable. Moreover, many of the features as correlated (as expected, since a similar approach is used in their derivation). It is clear that some feature selection is necessary in order not to overfit the model. This is discussed later in this paper.

In the following section another feature is explored that measures the similarity between search terms and the product title.

Extracting Word2Vec Features

The objective of Word2Vec is to map words to a numerical vectors. The vectors in question are obtained by training a two-layer neural network with the objective of predicting a word given a content (or vice-versa).

In practice the approaches used to obtain such a vector are either a continuous bag-of-words model (CBOW) or a skip a-gram model. These predict the occurrence of a word given a vocabulary and predict a vocabulary given a word, respectively. In the simplest case, the CBOW model is given a set of inputs where only the

[2]Each term weighting scheme was obtained by using the weightSMART function from R's tm package.

selected term has the value of 1. This is the only word whose weight is transferred to the following neural network layer, via a linear function. After this, the mid layer also transfers its output value to the next node. The training objective of this network is to maximize the softmax (conditional probability of observing the output word given the vocabulary). This equation is described as follows;

$$p(w_j|w_I) = \frac{v'_{wj} v_{wI}}{\sum_{j'=1}^{V} exp(v'_{wj'} v_{wI})} = y_j$$

where y_j is the output of the jth unit in the output layer,v'_{wI} is the input vector (weight coefficient of selected term), and v'_{wj} is the output vector (linear combination of weights of middle and output layer). The weights are then updated by using back-propagation according to the objective function. In the case of multiple word predictions there is one input layer for each word. The skip a-gram is the inverse approach where the the output layer output the softmax function for each target word.

The Word2Vec approach shows it is able to carry semantic meaning. For example, below you can observe closest terms to the vector represent ion of toilet and shower. This example shows that w2vec is able to determine that the term bathroom is indeed closer than the term restroom to the combination of terms toilet and shower.

```
## You seem to be using a VectorSpaceModel saved from an earlier version of this package.

## To turn on caching, which greatly speeds up queries, type

## yourobjectname@.cache = new.env()
```

Table 5: Nearest terms to the vector representation of toilet and shower in the pretrained word2vec model. The values shown are the cosine similarly calculated from the respective vector representations.

	toilet.and.shower
.	NA
of	NA
to	NA
and	NA
in	NA
a	NA
“	NA
’s	NA
for	NA

For this problem a pre-trained word2vec was used[3]. This choice was made to the long computation cost related to this feature. Moroever, even using a pre-trained model and calculating the cosine similarity took more than 24 hours in my machine (Macbook Pro Retina i7, Mid 2012).

In the following sections we turn away from feature extraction to explore modeling. Models are discussed in terms of both regression and classification.

Models

This section is divided among two groups of models, namely regression and classification. The classification problem simply being the correct classification into the bins 1, 2, and 3. For this task the training data that

[3]Pre-trained word2vec model http://nlp.stanford.edu/projects/glove/.

did not fall into these buckets was removed . Moreover, for model evaluation about 70% of the training data was used (this are selected by sampling from a binomial random variable). For the regression problem this results in 51337 rows used for training the model. The classification model uses 22827 observations for training. The remaining observations were left to be used for evaluation. These represent the test set. First, regression models are explored in the next sub section. Moreover, the metric used to compare models is the root mean square error as it was used in the original Kaggle competition leader board. The next section begins by exploring regression models.

Regression

Linear Regression

This section explores the linear regression model. Mixed step BIC was used to both determine terms of interaction (up to quadratic terms), and reduce over-fitting in the model. Below you can observe the coefficients obtained by this model in its entirety. Evaluating this model on the test set produced a root mean square error of 0.49333. This is quite high since it shows the model will deviate almost to another class from the actual value.

Table 6: Linear Regression coefficients obtained by performing mixed step BIC and linear regression in the complete set of features.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.0265382	0.0212292	95.460132	0.0000000
score.word2vec_Wiki	-0.0208426	0.0247376	-0.842547	0.3994858
score.brand_cover	0.0870198	0.0248076	3.507790	0.0004522
score.title_dimension	0.0651417	0.0072358	9.002631	0.0000000
score.description_dimension	-0.0361808	0.0138175	-2.618472	0.0088350
score.distance	-0.0012743	0.0006452	-1.975241	0.0482462
score.title_jaccard	1.7180458	0.1075381	15.976156	0.0000000
score.description_jaccard	-0.9218111	0.0805025	-11.450714	0.0000000
score.description_ratio	0.1119006	0.0086400	12.951489	0.0000000
score.title_log_tf	-0.6827599	0.1395087	-4.894032	0.0000010
score.title_mean_tf	-0.1440345	0.0333897	-4.313744	0.0000161
score.title_min_tf	0.1398993	0.0348704	4.011983	0.0000603
score.title_sum_tf	0.6712363	0.1400215	4.793807	0.0000016
score.title_mean_tfidf	-0.0507216	0.0208719	-2.430137	0.0150965
score.title_max_tfidf	0.0937502	0.0117920	7.950353	0.0000000
score.title_min_tfidf	0.0676276	0.0144119	4.692478	0.0000027
score.title_sum_tfidf	-0.0164777	0.0097349	-1.692652	0.0905278
score.description_log_tf	0.0172151	0.0057211	3.009046	0.0026220
score.description_max_tf	0.0103459	0.0036414	2.841177	0.0044965
score.description_sum_tf	-0.0117323	0.0049251	-2.382139	0.0172160
score.description_mean_tfidf	0.3368561	0.0325318	10.354656	0.0000000
score.description_max_tfidf	-0.0796111	0.0158153	-5.033796	0.0000005
score.description_min_tfidf	-0.1691271	0.0255112	-6.629525	0.0000000
score.description_sum_tfidf	-0.0376820	0.0158674	-2.374808	0.0175617
score.description_ratio:score.title_min_tfidf	-0.0216282	0.0040244	-5.374239	0.0000001
score.word2vec_Wiki:score.title_jaccard	-0.5519925	0.1388110	-3.976577	0.0000700
score.title_min_tf:score.title_sum_tfidf	0.0328815	0.0047109	6.979803	0.0000000
score.title_jaccard:score.description_sum_tfidf	-0.1490835	0.0180631	-8.253481	0.0000000
score.word2vec_Wiki:score.description_sum_tfidf	0.0845894	0.0142517	5.935383	0.0000000
score.title_max_tfidf:score.title_min_tfidf	-0.0167081	0.0027605	-6.052540	0.0000000
score.distance:score.title_sum_tfidf	-0.0005889	0.0001118	-5.266754	0.0000001

	Estimate	Std. Error	t value	Pr(>|t|)
score.title__min__tfidf:score.description__log__tf	-0.0119821	0.0013507	-8.871065	0.0000000
score.title__dimension:score.description__dimension	0.0272102	0.0064618	4.210928	0.0000255
score.description__jaccard:score.description__sum__tf	0.0238738	0.0079119	3.017449	0.0025504
score.word2vec__Wiki:score.title__max__tfidf	-0.0614574	0.0104921	-5.857518	0.0000000
score.description__mean__tfidf:score.description__min__tfidf	-0.0211764	0.0066625	-3.178434	0.0014816
score.description__log__tf:score.description__min__tfidf	0.0232633	0.0031609	7.359716	0.0000000
score.title__log__tf:score.description__max__tfidf	0.4384934	0.1182068	3.709545	0.0002079
score.brand__cover:score.description__max__tfidf	-0.0445434	0.0087422	-5.095232	0.0000003
score.title__mean__tfidf:score.description__min__tfidf	0.0160905	0.0043680	3.683767	0.0002300
score.description__dimension:score.description__ratio	-0.0342551	0.0099681	-3.436479	0.0005898
score.title__jaccard:score.title__min__tf	-0.3578619	0.0728671	-4.911162	0.0000009
score.title__mean__tfidf:score.title__sum__tfidf	0.0160378	0.0028602	5.607180	0.0000000
score.description__mean__tfidf:score.description__sum__tfidf	-0.0181948	0.0043256	-4.206346	0.0000260
score.title__max__tfidf:score.title__sum__tfidf	-0.0087285	0.0017549	-4.973841	0.0000007
score.title__sum__tf:score.description__max__tfidf	-0.4098084	0.1174721	-3.488560	0.0004860
score.word2vec__Wiki:score.brand__cover	-0.0967351	0.0330109	-2.930398	0.0033868
score.title__log__tf:score.title__mean__tf	0.0462030	0.0102565	4.504744	0.0000067
score.title__mean__tf:score.title__min__tf	-0.0640583	0.0173252	-3.697407	0.0002180

A small performance gain can be achieved by noting that the linear model will predict values that are higher than possible. The relevance scores range from 1 to 3, but this model predicts values as high as 3.275. Capping the prediction values for relevance at 3 reduced the root mean square error to 0.49329.

Regression Trees

The next model to be explored is classification trees. The package RPART was used in order to produce the tree. As shown the previous section, there are many terms of interaction present in this model, this indicates that a tree based model may perform well in the data. Unfortunately, this is not the case as the test root mean square error for this model is 0.505. Below you can observe the tree produced.

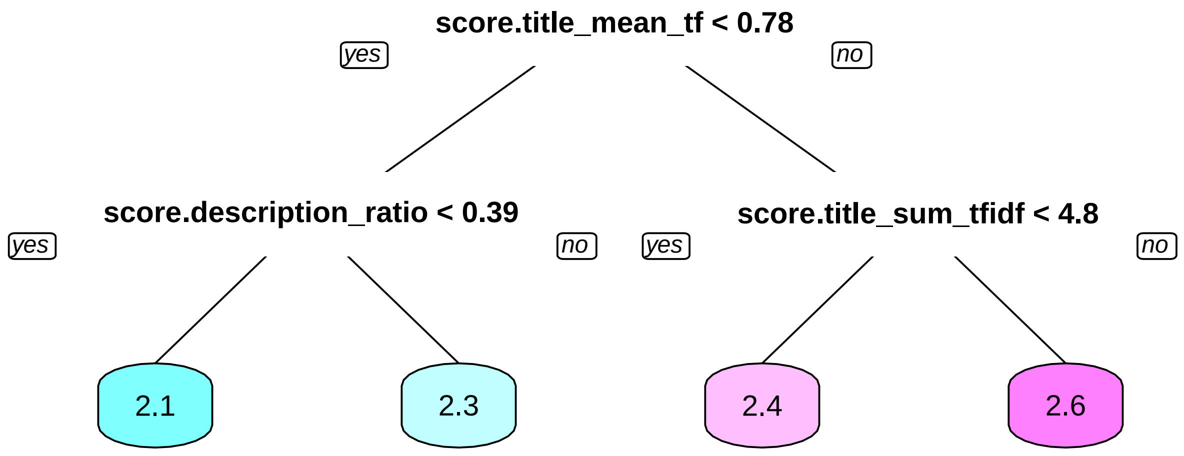
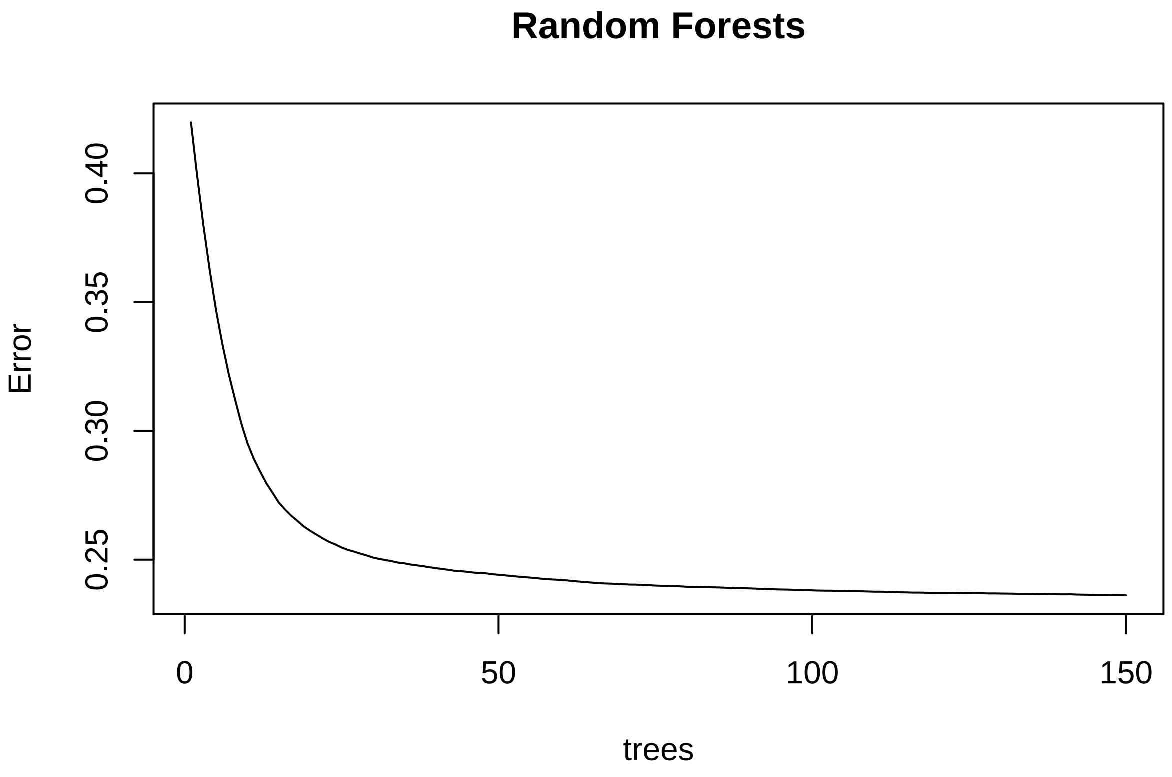

score.title_mean_tf < 0.78
yes
no
score.description_ratio < 0.39
yes
no
score.title_sum_tfidf < 4.8
yes
no
2.1
2.3
2.4
2.6

Figure 2: Regression Tree

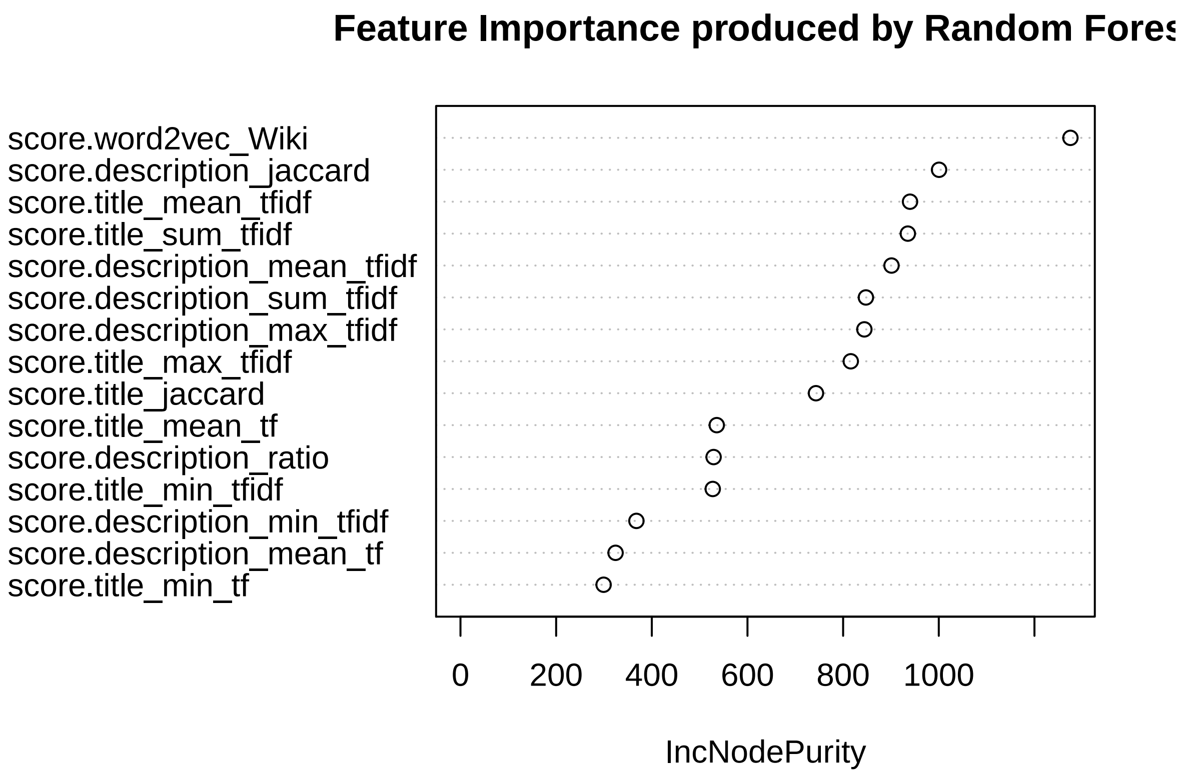
A slight reduction in the amount of pruning achieved by changing the complexity parameter (minimum R squared increase requirement for a split) yields a slightly better model, with root mean square error 0.497. However, this parameter change achieves a more complex model without significant increase in performance. Further reduction perform worse in the test data. The next section explores the random forest model, in which results are averaged among several trees.

Random Forest Regression.

Here random forest regression is used as the model. This model is the combination of several regression trees, each of which uses a random subset of the predictors. This has the advantage of growing a larger number of trees to improve prediction accuracy. Moreover, since the resulting prediction is averaged among the trees there is a reduction in over-fitting, so growing longer trees tends to increase performance more significantly (as compared to a single tree).



In practice, it does improve upon regression trees. Generating 50 random trees improves on the previously discussed regression trees model, as it achieves a root mean square error of 0.487. Increasing the number of trees further does not provide a significant improvement in the model.



Another advantage of using tree based models is that it allows for ranking of importance of the variables. This ranking is produced by determining the features achieve the most node purity (as these are considered for earlier splits). In figure 3 you can observer the top 15 features where word2vec is shown to be the best performing feature, followed by the description jaccard distance.

Classification

Multinomial Log-Linear Classification

This section presents the application of multinational logistic classification to the relevance search problem. As mentioned previously, the data provided in its entirety does not allow for a classification approach. However, by selecting observations whose relevance value is within a category enables one to use a classification algorithm. For this, the multinational log-linear model was applied using both the complete set of features, and the interaction terms found by performing step BIC. Below you can observe the confusion matrix of the former approach (the rows represent the gold standard, and columns the prediction value).

Table 7: Confusion Matrix produced by performing Multinational Log-Linear Classification.

	1	2	3
1	0	443	162
2	0	1741	1854
3	0	1069	4629

From the confusion matrix we can observe that this model is unsatisfactory. First, this model does not assign any observation to the 1 label. Moreover, many assignments are done in the other categories as well. This results in a classifications rate of about 35%. Adding interaction terms did not improve the performance of this model. Furthermore, when comparing with the majority class, which represents 57% of the test data, this is indeed a poor performance since assigning to it produces a classifications of 43%.

Ordinal Classification

The next model discussed is Ordinal Classification. This model is explored due to the target problem being in fact ordinal classification. In information retrieval problems typically the user is shown ranked results that best match the query, thus the order is more important then the actual class being matched. Moreover, all the features here positively correlated with the response variable, which also indicates that an ordinal approach will perform well.

Table 8: Confusion Matrix produced by performing Ordinal Classification.

	1	2	3
1	0	478	127
2	1	1773	1821
3	0	1053	4645

Above you can observer the confusion matrix obtained by applying ordinal classification. As before, no observation is assigned to the 1 bin, and many are misclassified. The miss classification rate on the test set is about 35%, and thus it does not produce a significant improvement over the multinational model. As before, adding interaction terms did not provide a significant increase in performance.

Conclusion

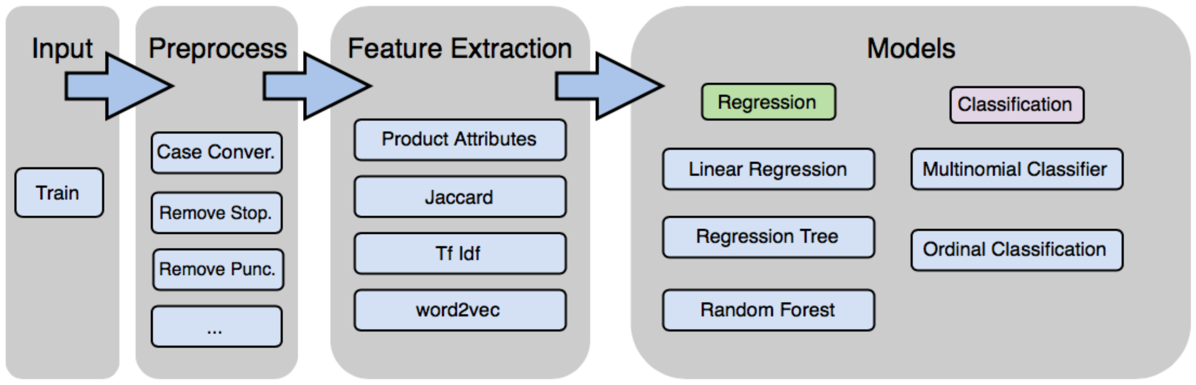


Figure 3: Flow Chart

In this study, several steps were taken to select a good performing model than can predict relevancy score of product and query pairs. The ideal matching as identified by Home Depot is when products match on their attributes and type. However, during feature exploration it was shown that only a small sample of the training data contains a matching in product attributes. This indicates that the customers of Home Depot typically do not care about product brand but instead look for products that fit theirs needs. Popular information retrieval features and natural language processing were used to explore this fact successfully, giving a significant improvement over the basic model. Additionally, the best performing model in this study was random forests . This model produced a root mean square error of 0.48 which is an improvement over the previous discussed regression models. As it was shown, the best performing feature for this approach was word2vec. As such, I expect that training such feature on texts from Home Depot.com could provide a bigger improvement. Moreover, the models explored in classification did not provide a significant improvement. The random forest model was also applied to classification (this was omitted due to space concerns) but it also achieves a classifications rate of 35%.

In conclusions, the HomeDepot search relevance challenge was a great case study for deepening my understanding of machine learning, but it also introduced me to diverse techniques in natural language processing such as term frequency and numerical representation of terms. These proved to be very time consuming, however worth it. As the results of random forest show, the node purity achieved by the word2vec model is significantly higher than the other features. The next being the jaccard description distance. This goes to show that the attempt of only dividing by query length proved to be unnecessary, as this feature still performed very well.

References

Rong, Xin. "word2vec parameter learning explained." arXiv preprint arXiv:1411.2738 (2014).

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.