Home Depot Product Search Results Relevance Data Mining

Antonios Andronis
Student, MSc (Data Science)
aa3e15@soton.ac.uk

Adisorn Jaroensathayatham
Student, MSc (Data Science)
aj7g15@soton.ac.uk

Napoleon Koskinas
Student, MSc (Data Science)
nk5g15@soton.ac.uk

Emmanouil Petsanis
Student, MSc (Data Science)
ep1g15@soton.ac.uk

Nikolaos Perrakis
Student, MSc (Data Science)
np4g15@soton.ac.uk

Syed M. Ali Shah
Student, MSc (Data Science)
smas1c15@soton.ac.uk

ABSTRACT

In this paper we discuss the approach of team Phi1337 on the competition "Home Depot Product Search RelevanceâĂİ offered by Kaggle and Home-Depot. Team Phi1337 as part of the group coursework of the Data Mining Module taught at the University of Southampton in the academic year 2015-2016. We were asked to carry out a supervised learning challenge as group coursework. The aforementioned competition constitutes a demanding challenge simulating a real business product faced by Home Depot with a real and noisy data. We started by exploring our dataset and cleaning it. We then moved in preprocessing and feature engineering. Those are the most important steps in tackling this challenge and we mention in detail the methods we applied. Subsequently we describe the supervised learning methods we used and their predictive accuracy. Lastly we comment on our results and what we have learnt from this project.

Keywords

Data Mining, kaggle, Home Depot, Product Search Relevance

1. INTRODUCTION

Businesses are always working towards finding ways to optimize processes and services in order to improve sales, maximize the profits and exceed customer services expectations. There are various factors contributing to the increase of sales. Our project deals with an aspect of online sales, product relevance on a customer's search query.

The relevance of a search result to the target product is a big challenge for the online retailers. If the customer finds a product they are looking for, covering their needs, it is highly likely that the customer will decide to purchase. If an online retailer knows how to suggest a customer the products they are looking for, may improve their business performance in its respective market, if not, they may be in danger of running out of business.

Online product retailers, including selling platforms like Amazon and eBay, use complex predictive algorithms to ensure that the appropriate product is displayed to the potential customers based on their search queries. Customer queries are noisy. Sometimes they contain spelling errors, sometimes incorrect terms or special characters. What is more, the product information does not always accurately represent a product. The Machine Learning models run by online retailers aim to accurately access the relevance of each product in order to provide the customer with the most relevant query.

Our project team was highly motivated to research on this challenging subject by participating in a Kaggle competition[1] where a US based company, called "Home Depot", provided a dataset and hosted a competition. They are a renown home improvement specialty retailer worldwide that has provided the opportunity to kagglers to work on their dataset and develop a model which can accurately predict the relevance of customer search queries.

In this competition, the expectation is to develop a model that can "accurately predict the relevance of search results" [1]. The Home Depot team has provided the search queries and a product to be compared for each query. On the training set the comparison was classified as:

Rating 1: Product irrelevant compared to the search query.
Rating 2: Partially relevant product.
Rating 3: Highly relevant product.

In this paper we describe our work in creating a supervised learning model that accurately predicts the relevance of a product compared to a particular search result.

2. MACHINE LEARNING AND SYNCHRONIZATION TOOLS

We start by describing what tools we used in our work. The data modeling for this project was done in Python (version 2.7). In order to exercise the supervised learning algorithms we have used Python's Machine Learning/Data Mining supporting software called "scikit-learn"[3]. Scikit-learn is very efficient tool for data analysis and contains a wealth of Machine Learning/Data Mining algorithms.

We have used "Pandas" and "Numpy" which are two very powerful libraries providing easy-to-use data manipulation and analysis tools. We have also used matplotlib library to visualize the results of our exploratory data analysis and learning outcomes. In order to view our analysis results inline with the codes we have used "Jupyter IPython Notebook", this has also helped in team collaboration by sharing within team.

Furthermore, we used Microsoft OneNote for the purpose of synchronization and keeping track of meeting notes. For the creation of the report we used the LaTeX editor and the Overleaf online editor to write different parts of the report in parallel. For the presentation, we have used Powerpoint.

The source codes are placed in GitHub repository[4]. This enabled us to work separately on the assigned task and share our work within the team. Github has also helped us to divide the work very efficiently within the team and maintained trace-ability of version control as well.

3. DATASET

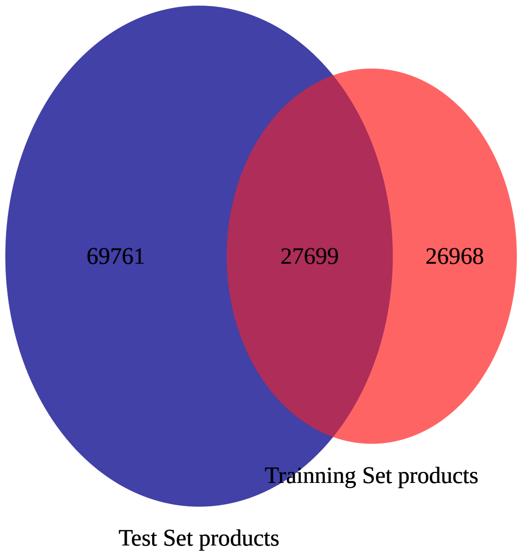

Figure 1: Distribution of unique products between the training and test sets.

Home-Depot provided the data needed to the Kagglers taking part in this competition.

The training set consist of rows of search queries identified by a unique id, a product id that the search is referring to, the product's title the search query and a relevance score. The relevance score is the average of three human raters who rate whether a search term was relevant to the product with values one(1) to three (3).

On the other hand, the test set has the same columns as in the training set, but the relevance score is not provided in this case. The task for the competitors is to predict a relevance score for every row of the test set representing a search. When the results are submitted , they are evaluated to the expected values using a RMSE.

Home-Depot provide extra information about the products in two additional datasets which enablers the competitors to expand the data from which they will draw their feature space.In particular, one file containing a description per product and another file containing attributes per product rare offered. All those fields are connected by the unique product id which acts as a primary key in a relational database.

It should be mentioned that description and attributes are not available for every for every product, and that attributes are not the same for every product as there are several categories. That would create high sparsity in case we attempted to construct our features depending on the attributes.

4. PREPROCESSING

4.1 Non-alphanumerical handling

When exploring the dataset and more specifically in product title and description string fields we observed characters which do not belong to the English language and non-ASCII characters. These characters were decoded using the Python built-in library called "Unidecode" which transforms an non-alphanumeric character out of range (0-127) into an original character range. On the other hand, non-ASCII characters were located and ignored as we believe these do not have an impact on the search relevance results.

4.2 Punctuation characters

We decided to use a bag of words approach for feature construction. For this reason, we focused on the string fields and replaced punctuation like periods, commas, question marks etc. corresponding to strings of characters so that the format of our text is consistent. For example, if there is no space between two words which belong to different sentences and are separated by a comma or a period, the new format of text adds one space on either side of the punctuation character, so that they can be easily separated by space later. We could replace those characters directly by space, but the approach we selected is useful in cases like measurement units (e.g. "in." for inches) which are useful as features.

4.3 Unit of Measurement

Important features can be extracted by product title, features and description related to unit of measurement of products. The issue we have noticed is that the unit of measurement are not correctly and consistently entered in the dataset. They appeared in various forms creating the repetition of different feature values which in fact represent the same value. In order to avoid the different style entries of the same unit of measurement, we identified and replaced with the same unit of measurement to maintain the consistency. Our approach on unit of measurement was influenced by a script published from Kaggle[7].

4.4 Typing errors in search queries

We have found that the search queries have typographical errors including spelling mistakes. We have followed a nice approach to handle this issue as done by one of the fellow Kaggler "Steubk"[2]. We used a simple function that identified misspelled words in the search query and corrected it using Google's auto-correct.

4.5 Tokenizing, Stemming and Stopwords

We created the bag of words by tokenising and stemmming the words which appear in string fields. Firstly, we tokenized the search query by sentence and then by word to be sure that punctuation is caught in its own token. After this step, we applied stemming on each word using an English language stemmer ("SnowballStemmer") included in NLTK[5] library), corresponding words with the same root to the same term. Finally, we removed the English stopwords in product titles, attributes and descriptions using the NLTK library of Python.

After the completion of the pre-processing phase we had a bag of words for every product coming from the product's title, description and attributes. We took advantage of that bag of word for creating our feature space.

5. FEATURE ENGINEERING

The choice of features directly affects the data mining techniques which will be later applied as well as the efficiency of those techniques. In fact, we realized that when we created features which were representative of our dataset, we observed considerable improvement in our algorithms' results. Understanding the dataset is a prerequisite to creating the right feature space which constitutes a compact and accurate representation of the available data and characterizes in a better way the underlying problem.

In a text mining case, like the one we are facing, a first thought would be to create a feature space with all the terms that appear in every text field. That would create a very high dimensional space and a very sparse corresponding data matrix, which would exponentially increase the calculations' complexity and subsequently decrease the performance of the algorithms.

As a result and since automatic generation of the feature space is difficult, expensive and requires models that are very complex, slow and difficult to maintain, manual feature engineering is suggested and was adapted throughout our approach. The script implemented and published at Kaggle[7] was helpful in implementing a nested Series-wide mapping function and is also documented by comment in our code[4].

5.1 Common word ratios

The first intuitive approach that we had was to create ratios of the common words between those in the search queries and the descriptive fields of the products (title, description and attributes). Therefore, we created three features by dividing the number of common words between the search query and the title, description and attributes, joining by the product's id. The idea for the ratio came from the belief that the importance of a matching word within a query is inversely proportional to the length of the query. For example, two different queries with the same number of common words but different length should be evaluated with different weights.

5.2 TF-IDF

As a second approach, we applied the popular TF-IDF algorithm to the same fields(title, description and attributes). We decided to use this method, as it is a basic way to get the most descriptive terms in a document and it helped in building three more features, representing the similarity (on this case we calculated the cosine similarity) score between search query and the fields mentioned above.

5.3 LSA with SVD

Our third approach was to use an automatic feature generation method and perform dimensionality reduction on the generated features using Truncated-SVD (also known as Latent Semantic Analysis - LSA) to transform the sparse generated feature matrix to a "semantic" space of lower dimensions. In our case this proved useful, as LSA captures synonymous and polysemous words which intensify the sparsity of the matrix. This greatly improves the results of metrics such as cosine similarity.

5.4 Analyzing the search query

By exploring the search queries we observed that the users tend to search using only a few words. We therefore made the assumption that the existence of the search query in a field and especially in the title or the description would reveal such a high relevance that it makes it worthy of being a feature on its own. In addition, the last word of a query is usually a noun following adjectives or attributes of the product so we believe that it characterizes the kind of the product. Thus, we isolated it and searched for the its occurrences in the same fields.

5.5 Exploring the brands

During our data exploration on the attributes dataset we observed that the most common attribute was the MFG Brand Name. More specifically, it was included in 86250 out of 86264 products which had at least one attribute. We decided to take advantage of this attribute and the fact that it occurred for almost every product and we had enough distinct brand values (4290) as we thought it was the most suitable to create features out of. More specifically, we created three features. The first one is a counter of the number of brand name words. The second one is a ratio of the common words between brand and search query. We chose as a denominator the brand word count as we believe that it is more relevant to divide with this than the count of the query words. Lastly, the third one is a mapping of the brand names into distinct numerical integer values, separated in intervals of three (3) to increase the variance of this variable.

5.6 Simple word counts as features

After applying the machine learning algorithms and extracting feedback, as a last effort to improve the accuracy of the model we embedded a few extra simple features, which did not change dramatically the results, although they offered some minor improvements. We briefly mention them below:

1. Count of words in search query.
2. Count of words in title.
3. Count of words in description.
4. Count of common words between search query and title.
5. Count of common words between search query and description.
6. Length of search query.

Feature Engineering is a continuous process which relies on understanding the dataset and results after the application of data mining algorithms out and using the knowledge extracted to construct the right feature space which can enhance the algorithms predictive ability. In fact, we observed while experimenting that adding the right features improved the performance of algorithms in RMSE metric by 2-2.5%, a much better improvement than the one we had when we attempted to tune the parameters of algorithm functions.

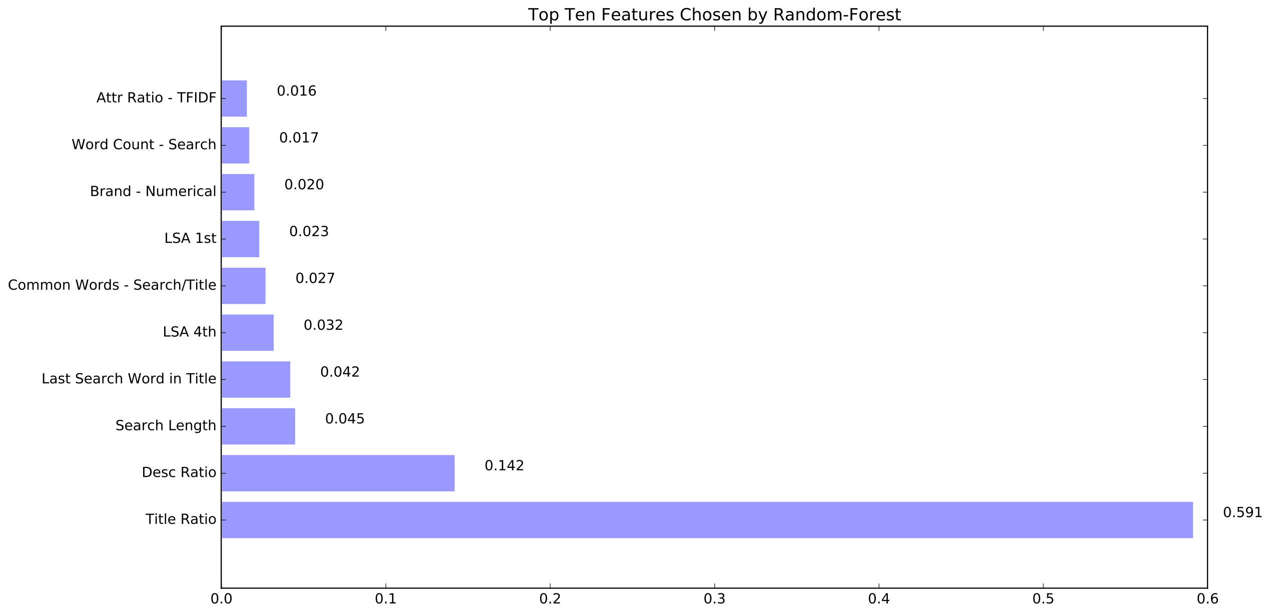

Figure 2: Features ranked by importance according to contribution in random forest algorithm.

6. SUPERVISED LEARNING MODELS

The goal of the project is to explore the model which predicts more accurately the relevance of product searches. To achieve this we tried various approaches by constantly experimenting and evaluating their performance. We focused on three main approaches:

1. Linear Regression.
2. Random Forest Classifier.
3. Support Vector Machines.

One of the most important parts of the data mining pipeline is to get a better understanding of the given data after applying a supervised learning algorithm. Then the team is able to get feedback for the preprocessing stage in order to construct a more appropriate feature space, to better fine-tune the model or even choose a different algorithm.

6.1 Linear Regression

To begin with, we decided to try a simple approach to get acquainted with our dataset. The first technique we tried was a linear regression model using as independent variables the ratios of common words between the search query and the product's title, description and attributes. Our results are assessed with root mean square error (RMSE) and are presented in the following table:

Training Set:	0.49311
Validation Set:	0.49116

Subsequently we applied linear regression after using Term-Frequency Inverse Document Frequency (TFIDF) similarities as independent variables. We got the following results:

Training Set:	0.51502
Validation Set:	0.51326

Lastly, we applied linear regression after vectorizing word counts and applying Latent Semantic Analysis (LSA) on them keeping the 4 features with highest eigenvalues. Our results were:

Training Set:	0.51295
Validation Set:	0.51042

In both cases we got worst performance compared to linear regression on the common words ratios. This tells us that the common words ratios are features with better predictive power compared to TFID and LSA.

Our results were still not satisfactory so we decided to go back to the drawing board of feature construction. This resulted in the full preprocessing and feature engineering analysis described in the earlier sections. After we run the linear regression algorithm on our full feature space we got the results on table below:

Training Set:	0.48063
Validation Set:	0.47979
Test Set:	0.48765

As we can see our linear regression model has a consistent performance between the training, validation and test sets. There is a hidden caveat however. We had to prune results with less than 1 and higher than 3 relevance. Without it our results were more inconsistent. Given that the relevance score lies in between 1 and 3 we are allowed to do that.

6.2 Support Vector Machines

The next supervised algorithm we tried is Support Vector Machines. We tried various kernels and appropriate parameters. We had to be careful not to overfit. The best results we got are presented in the table below:

Training Set:	0.45059
Validation Set:	0.47641
Test Set:	0.49091

As can be seen even in this result we had some overfitting. We could resolve this by decreasing the model's performance but it would decrease significantly. We decided to stay with a model that had the drawback of some overfitting but performed better in the test set overall.

6.3 Random Forest Regressor

Last but not least we tried Random Forest Regressor. It is a very powerful ensemble method that proved to be very efficient. We also had to be careful not to overfit. Random Forest was the model most prone to overfitting[1] Random Forest proved to be our best predictive model with results shown in the table below:

Training Set:	0.47444
Validation Set:	0.47777
Test Set:	0.48815

As we can see our model proved to have a good generalization ability. The error in the test set increased only lightly and we can accept it given how noisy our dataset is and how many products are unique to the test set (see figure 1). Moreover the random forest algorithm gave us a way to assess our work in feature engineering. For this reason we used the feature importance property of the trained model and created figure 2 which shows the most useful features for accurate relevance score prediction. The title and description common words ratios are the two top features, a result that collaborates with our initial results from the linear regression algorithm.

7. RESULTS AND CONCLUSIONS

During our training we noticed that extra care had to be taken on using Support Vector Machines and Random Forest Regressor so that they did not overfit. Since the given dataset was very noisy, our models were prone to overfitting. Random forest however was pretty good in giving us both good and consistent performance.

Most importantly from the beginning of our project we understood that preprocessing and feature extraction was the key step which we should focus on. We tried several Natural Language Processing techniques to extract features from our data but the simplest, a bag of words approach, proved to be the most effective. As we can see from the winning team[6] their work revolved mainly around feature construction. After going through the complete feature engineering process a single ensemble method was able to get RMSE 0.435 which is in the top 10.

As we can see from figure 2 all the additions we did on our feature space during feature engineering added on the random forest's algorithm performance. Albeit not all of them as we would expect. The word count of the search query is the first surprise because it does not relate to the product in question. The last search word in search title is a feature whose importance demonstrates how domain knowledge can help more than standard machine learning techniques, in this case LSA and TFIDF. Those two methods however do contribute as we can see in the figure.

To conclude what we learnt from this project is that we should be very careful and dedicated in feature engineering. Starting simple and using the results as feedback to go back to the drawing board is a very useful approach that we practiced in this project and are likely to practice many times in our lives.

[1]The fact that Random Forest had short computational time helped us experiment even more with it.

8. REFERENCES

[1] Home depot product search relevance competition page. Available from: https://www.kaggle.com/c/home-depot-product-search-relevance [Accessed 4 May 2016]

[2] Steubk K, (2016) Home depot product search relevance. Available from: https://www.kaggle.com/steubk/home-depot-product-search-relevance/fixing-typos [Accessed 5 May 2016]

[3] Pedregosa et al., Scikit-learn: Machine Learning in Python, *JMLR 12*, pp. 2825-2830, 2011.

[4] Team GitHub repository: https://github.com/Iolaum/Phi1337

[5] Steven Bird NLTK: the natural language toolkit, *'06 Proceedings of the COLING/ACL on Interactive presentation sessions* Pages 69-72

[6] Kaggle Forum Post: Congratulation to the winners. https://www.kaggle.com/c/home-depot-product-search-relevance/forums/t/20427/congrats-to-the-winners [Accessed 5 May 2016]

[7] Random Forest Mean Squared Error script. https://www.kaggle.com/the1owl/home-depot-product-search-relevance/rf-mean-squared-error [Accessed 5 May 2016]