Search Relevance Prediction for Home Depot Products

Natural Language Processing Course Project- MSc in Data Science 2019-2020

Patrikios Georgios, Tsironidou Eleni

School of Science and Technology

International Hellenic University

Abstract – Shoppers rely on Home Depot's product authority to find and buy the latest products and to get timely solutions to their home improvements 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.

In this paper, we use Natural Language Processing and Machine Learning to help Home Depot improve their customers' shopping experience by developing a model that can accurately predict the relevance of a search results.

I. INTRODUCTION

This report contains the findings of our coursework regarding the course of Natural Language Processing (NLP) under the supervision of Professor Apostolos N. Papadopoulos at International Hellenic University. Specifically, we are given the "Search Relevance Prediction for Home Depot Products" problem. Our goal is to develop a model that can accurately predict the relevance of search results trying to achieve the lowest Root Mean Square Error (RMSE).

The whole process takes place on Google's Colab, which is an open–source cloud service based on Jupyter Notebook that supports free CPU [1]. The remainder of this paper is structured as follows. Firstly, there is the description of the data along with the preprocessing procedure and feature engineering. Secondly, the results of the performance of the models are presented.

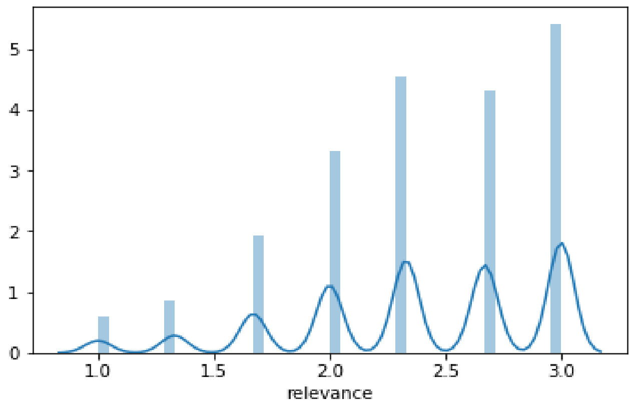
II. Data Exploration

The dataset for addressing the Search Relevance prediction problem contains a number of products and real customer search terms from Home Depot's website. To create the ground truth labels, Home Depot have crowded sourced the search/product pairs to multiple human raters. It consists of four different files: train.csv, test.csv, product_description.csv and attributes.csv.

The training set contains the products as well as the corresponding search terms that were used to retrieve the product. Moreover, this dataset contains the relevance score for each product-search pair. These scores supplied to the human graders provide a relevance grade of either:

- Irrelevant (1)
- Partially relevant (2)
- Perfect match (3)

The provided relevance scores are the average value of the ratings. There is a total of 13 unique values for relevance in the train data. A bar chart of the frequency is displayed below.



Relevance is the attribute that is going to be predicted for a given search term. The product_description.csv contains the description for the corresponding product in the train.csv and test.csv

In order to extract more features a manipulation of attributes.csv takes place. The "MFG Brand Name" and "Material" attributes are used for the creation of the brand and material features respectively. However, the brand or the material may not be mentioned in the attributes.csv for any product. Furthermore, the "Bullet" attributes are extracted since they contain useful description of the products. As previously pointed, a product may not have bullets however there are cases where some products contain up to twenty-two bullets. For this reason, the bullets for each product are extracted and joined in one string. So, a description for each product is created. Finally, the above features are merged with products on "product_uid". Simply, if a product does not contain any of the values above, it takes "NaN" value.

The approaches of 'Preprocessing' and 'Feature engineering' below are performed on both train.csv and test.csv. In the 'Model Performance' and 'Results and Comparative Experiments' the results are obtained working only with the train.csv.

III. Preprocessing

Text preprocessing is an important phase before applying any machine learning NLP algorithm. Although text preprocessing can be technically done within feature generation process, the preprocessing takes place first and then features' generation. This happens since the same preprocessed text is used to generate several different features. One of the most common preprocessing techniques in NLP the stop words removal. Stop words usually refer to the most common words in a language like: is, at, which, to, on etc. Due to the fact that some of these words probably will appear multiple times, they should be removed. In Python there is the "StopWordRemover" module, which is used for all datasets preprocessing.

In addition, word stemming preprocessing technique is used. Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base root form – generally a written word form. The "Snowball Stemmer" Python library is used. Word stemming is applied to train, test, product_description and attributes datasets.

Another preprocessing approach which is important to be pointed is the spell correction. After the detailed analysis of the search terms several misspellings are observed. Hence, an open source check-spelling dictionary for this purpose is used [10].

IV. Feature engineering

After the preprocessing phase we proceed to the features' generation stage. The first feature is implemented to attributes of interest while the rest are applied between the following pairs:

- search term-product description
- search term-product title
- search term-brand
- search term-material
- search term-bullets

- Length of string

This feature targets to find the length of a string in terms of words of an attribute. The results can be proven extremely helpful.

- Common words

Moreover, another important feature is the number of common words between two strings.

- Common whole word

The difference between this feature with the previous one is that it counts the occupancies that the whole search team appears inside in the strings of interests.

- Fuzzy Logic Ratio

Fuzzy logic [2] is a form of multi-valued logic that deals with reasoning that is approximate rather than fixed and exact. Fuzzy String Matching, also known as Approximate String Matching, is the process of finding strings that approximately match a pattern. For the purpose of this project "fuzz.ratio()" method is used in order to calculate the edit distance between some ordering of the token in both input strings.

- Jaccard Similarity

Jaccard similarity is extremely popular in computing the textual similarity among documents, strings. Jaccard Index of sets, also known as the ratio of the size of intersection to the size of the union.

- Cosine Similarity

Cosine Similarity is a measure of similarity between two non-zero vectors of an inner product space. Firstly, TF-IDF transformation is applied in order to get two real-valued vectors and then the cosine similarity is calculated.

- Ratio

The following formula is used in order to create features based on ratios.

$$ratio = \frac{common\_words}{len\_of\_query}$$

- Word Embeddings

The code for creating the embendings.csv is provided in the 'embendings.ipynd'. The "vector_similarity.csv" contains the cosine similarities of the vectors created using Word2Vec [3].

Firstly, a vector-dictionary based on the words provided by the problem is made. Passing text to the Word2Vec model happens in the form of documents. Specifically, for every row in the dataset search term, product title, product description and bullets are merged in a single string. It is worth mentioning that search terms do not necessarily contain words existing in product titles, product descriptions and bullets. Furthermore, the order of merging the strings has a significant role. In this approach the search term along with the product title are placed in the middle so the order is the following: product description, search term, product title, bullets. This way, the words in the search term strings are correlated in an equal sense both with product description and bullets.

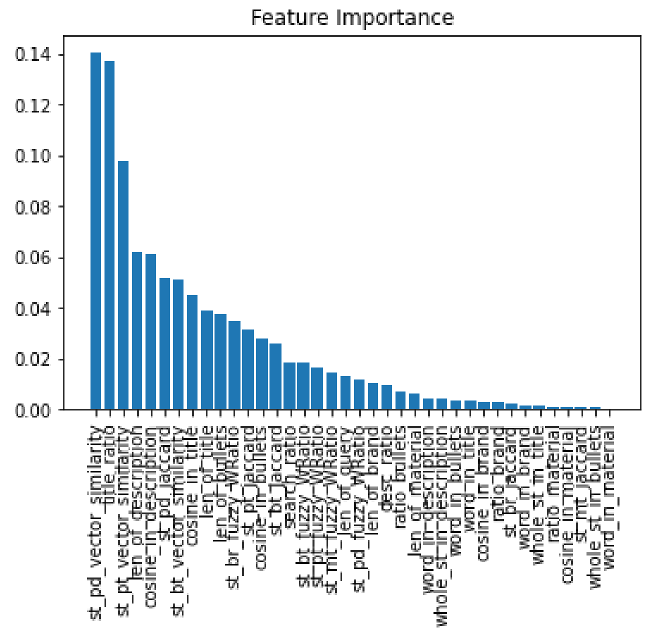
After that, the size of the vector and window parameters of the Word2Vec model defined. After several tests, the size of the vector is set to 150 and the window to 25. The above parameters minimize the RMSE according to this approach.

In order to compute the cosine similarity, the strings should be represented by one vector. To accomplish that, the mean vector (of the Word2vec vectors) for every string of interest is computed. The outcome is one mean vector of size 150 for each search term, product title, product description and bullets.

Finally, having the above mean vectors the cosine similarity is computed between the following mean vector pairs: search term-product title, search term-product description and search term-bullets.

- <u>Feature Importance</u>

Before proceeding to the models training and evaluation each feature's importance is measured using Random Forest Regressor. Feature importance refers to techniques that assign a score to input features based on how useful they are at predicting a target variable. The chart below contains the results.



Despite the fact that some features achieve score less than 0.1, it is decided to be used in the next stages.

V. Model Performance

After preprocessing and feature engineering, the following models are chosen in order to decrease the RMSE.

- *Random Forest Regression*

Random Forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual tree [4].

- *XGBoost Regression*

Boosting is a sequential technique which works on the principle of an ensemble. It combines a set of weak learners and delivers improved prediction accuracy. At any instant, the model outcomes are weighed based on the outcomes of previous instant. The outcomes predicted correctly are given a lower weight and the ones miss-classified are weighed higher. Note that a weak learner is one which is slightly better than random guessing [5].

- *Gradient Boost Regression*

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting. Gradient boosting models become popular because of their effectiveness at classifying complex datasets [6].

- *Ridge Regression*

Tikhonov regularization, as known ridge regression, is particularly useful to mitigate the problem of multicollinearity in linear regression, which commonly occurs in models with large numbers of parameters. In general, the method provides improved efficiency in parameter estimation problems in exchange for a tolerable amount of bias [7].

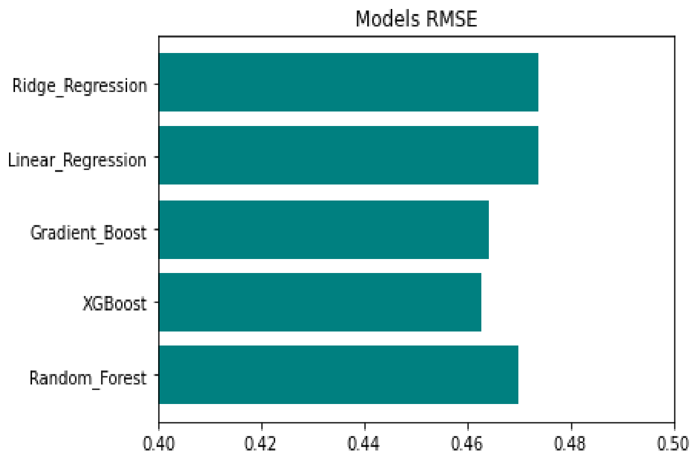
- *Linear Regression*

In statistics, linear regression is a linear approach to modeling the relationship between a scalar response (and dependent variables) and one or more explanatory variables (or independent variables) [8].

VI. Results and Comparative Experiments

The RMSE scores are obtained for the models mentioned above using 20-fold cross validation. The purpose of K-fold cross-validation [9] is to test how well the model is trained upon a given data set and test it on unseen data. The following charts display the results.

	Statistical Metric
Models	*Root Mean Square Error*
Random Forest	0.4698
XGBoost	0.4627
Gradient Boost	0.4643
Ridge Regression	0.4739
Linear Regression	0.4739



As it shown, Gradient Boost and XGBoost models score the lowers RMSE. For this reason, they will be used in Hyperparameter optimization procedure. Hyperparameter optimization or Grid-Searching is the problem of choosing a set of optimal hyperparameters for a learning process. Several iterations take place in order to find the best combination. Since XGBoost Regressor provides the lowest RMSE it is decided to test this model with various parameters in contrast with Gradient Boosting where tuning takes place only for max depth of the tree.

- XGBoost Grid-Searching

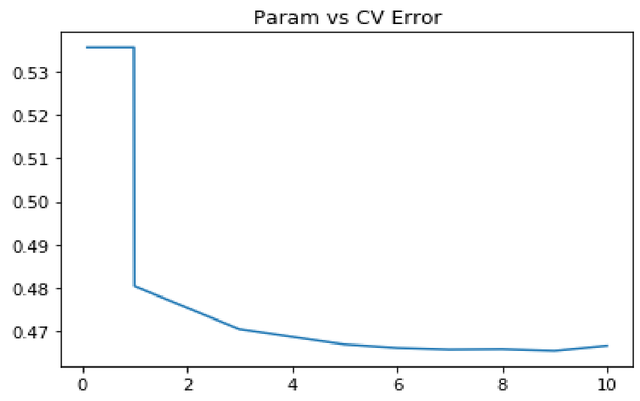
The combinations of parameters that the model is tested are up to 1920 fits and the best parameters displayed below:

◊ *Colsample_bytree*: 1
◊ *Gamma*: 0.5
◊ *Learning_rate*: 0.08
◊ *Max_depth*: 7
◊ *Min_child_weight*: 1
◊ *N_estimators*: 100
◊ *Subsample*: 0.75

It is found that the best parameters are the same used before Hyperparameter optimization. Hence, it is not necessary to reconstruct the model and test the RMSE score again with new parameter combination.

- Gradient Boosting Grid-Searching

Regarding this model tuning only takes place for the max depth of the tree. The graph below illustrates the results.

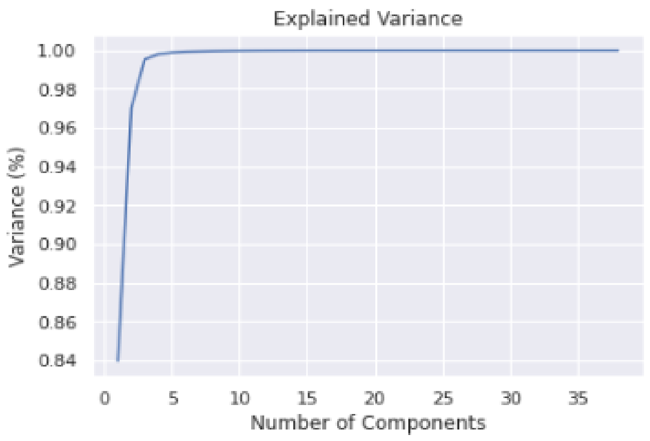


It is observed that by the time max depth is assigned with the value of 6, RMSE score starts to be stabilized under 0.47. The initial value of max depth, before grid-searching is 6. Due to the consolidation of RMSE above 6, no further testing is necessary regarding this parameter.

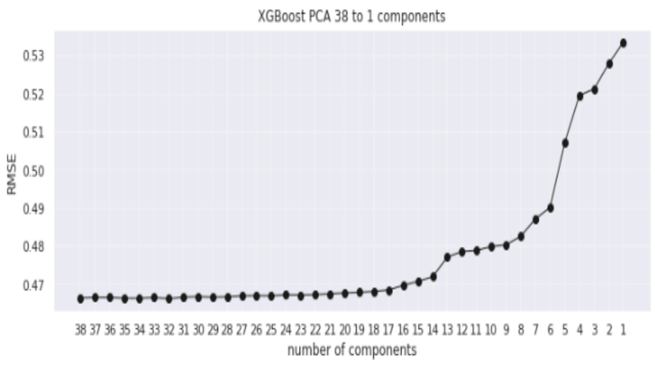
- Principle Component Analysis

Another approach targeting to the decrease of RMSE is dimensionality reduction. Dimensionality reduction is the notion that reducing the number of random variables under consideration has an array of positive effects on the model performance.

According to the explained variance of each feature it is found that 99, 8% of the original data information is preserved using only 5 components.



However, the XGBoost model is tested using all components separately starting from 38 to 1 component. This procedure results in the fact that using 32 components scores 0,4662 which is the lowest RMSE. Considering the score of XGBoost before PCA, there is no significant improvement.



VII. Conclusion

During this project, many techniques are surveyed and applied for predicting the relevance between a query and a product. In detail, 38 features are extracted from the original dataset and 5 regression models are used (Random Forest, Gradient Boost, XGBoost Regression, Ridge Regression and Linear Regression). In addition, further techniques such as cross-validation, hyperparameter optimization and principal component analysis are used in order to decrease the Root Mean Square Error. Finally, the results reveal that in this particular approach XGBoost Regression model would score the lowest RMSE. Through this analysis a thorough understanding of the existing natural language processing techniques is achieved and a better insight in designing time efficient and accurate models for search relevance prediction.

VIII. Future Work

The intention of this part is to discuss about future work in Search Relevance prediction for Home Depot products. So far, the main body of the experiment is focused on the feature extraction and the training several models in order to achieve the lowest RMSE score. Due to the fact that feature extraction stage gave many features, they can be grouped into different combinations. Furthermore, a wider spectrum of models could be tested like SVM and Neural Networks. Also, instead of principal component analysis, singular value decomposition could be tested. Another thing worth mentioning is that this problem could be approached as classification problem by labeling the results as relevant or irrelevant. Finally, different approaches word embeddings techniques could be applied.

IX. References

[1] Google Colab Tutorial: colab.research.google.com

[2] Fuzzy String Matching: towardsdatascience.com/fuzzy-string-matching-in-python-68f240d910fe

[3] Tomas Mikolov, Ilya Sutskever, Kai Chen, Kai Chen, Jeffrey Dean (2013). Distributed Representations of Words and Phrases and their Compositionality. Neural Information Processing System

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

[5] XGBoost, Wikipedia: en.wikipedia.org/wiki/XGBoost

[6] Gradient Boosting, Wikipedia: en.wikipedia.org/wiki/Gradient_boosting

[7] Ridge Regression, Wikipedia: en.wikipedia.org/wiki/Tikhonov_regularization

[8] Linear Regression, Wikipedia: en.wikipedia.org/wiki/Linear_regression

[9] Cross Validation: towardsdatascience.com/cross-validation-a-beginners-duide

[10] Fixing Typos: www.kaggle.com/steubk/fixing-typos