Feature Extraction - Assignment 2 - Report

Petridou Elena (s2029197), Spiro Andrew (s4015355)

April 2024

1 Task Definition

The ”Home Depot Product Search Relevance” competition on Kaggle challenges participants to develop a model which predicts, for a search term, the relevance of search results (products from their catalogue). Relevance is scored between 1 (not relevant) and 3 (highly relevant) for search-product pairs. A human-created list of the search-product pairs and their relevances is provided for training the model. Additionally, a ”product_description” file provides a description of each product and an ”attributes” file provides further details about some of the products. The model must predict the relevance of product-search pairs in the test file. These predictions will be the submission for the ”Home Depot Product Search Relevance” competition, and will be used to assess the performance of the model. Due to the availability of a human-labelled dataset on which the models are trained, this is a supervised-learning task, and since we are tasked with coming up with a value from 1 to 3 for the relevance of a search result, on a continuous scale, this is a supervised regression task.

2 Data Exploration

The training data consists of 74067 product-query pairs and 53489 unique products. The most occurring products in the training data are *Lithonia Lighting All Season 4 ft. 2-Light Grey T8 Strip Fluorescent Shop Light* and *Pressure-Treated Timber #2 Southern Yellow Pine (Common: 4 in. x 4 in. x 8 ft.; Actual: 3.56 in. x 3.56 in. x 96 in.)* each occurring 21 times. The mean relevance value in the training data is 2.381634, the median is 2.330000, and the standard deviation is 0.533984. The below histogram (Figure 1) summarizes the distribution of the relevance values.

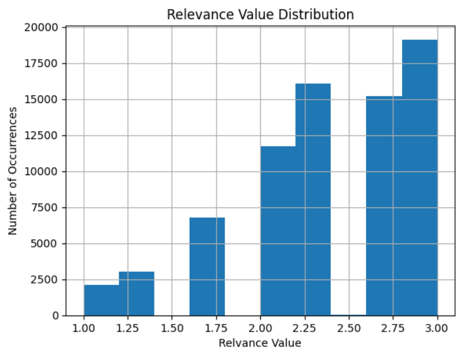

Figure 1: Frequency of product-query relevance values in the training set.

The top-5 most occurring brand names in the product attributes (listed under the attribute name "MFG Brand Name") are: Unbranded, Hampton Bay, KOHLER, Everbilt, and Home Decorators Collection.

3 Replication of Baseline Method

The method implemented by Yao-Jen Chang in the provided file works in three distinct steps: Step one is pre-processing. First, the train and test sets are imported, as are the product descriptions, which have been provided as a separate CSV file. Then, the whole dataset is combined into a single dataframe in order for the feature engineering to be carried out. To do this, the product title and description, along with the search query are 'stemmed' — a process whereby the words are reduced to their root. Then, all three of the aforementioned features are combined into a single cell for each entry. This is done in order to apply a function which counts how many times the search term appears in the product title, and the product description respectively. Next, the attributes in the dataframe that do not serve as features in the machine learning model are dropped, and the dataframe is subsequently split back into the original train and test sets.

The second step consists of training the models. Using the two subsets created previously, a Random Forest and Bagging regressor are trained.

In the third and final step, the Bagging Regressor (with the random forest regressor as its estimator, via the first argument) is used to estimate the relevance for the search query - products pair in the test set, and these are exported to the submission file.

Comparing the predictions on the test set to the true relevance values yielded a Root Mean Squared Error (RMSE) of 0.50 which agrees with the expected value 0f 0.48 up to 2%. Re-running it with the same features but un-stemmed yields an RMSE of 0.54, which is not a massive difference but shows that pre-processing of features can help make features more accurate.

4 Features and Hyperparameter Optimization

The features selected were informed by Data Exploration, mainly focusing on the availability of attributes, and the structure of search terms.

Firstly, we carried on with the method introduced in Yao-Jen Chang's code and computed how many times a word is shared between the search term and the product's attributes (in addition to the product title and product description counts, which were already part of the original code).

Moving onto new feature types, to determine which attributes might make for the best features, we first queried the top available attributes in the attributes.csv and saw that the most occuring attributes were measurements and colour/colour family. The measurements came in many different formats, which we pre-processed accordingly.

Next, we observed that in the search terms, colours and measurements were often paired with a rough product description (akin to the product name), followed by a string of 2 measurements, for example: "MDF board 4x4".

Thus, we chose to combine these attributes with the product title to create new values which resembled the search entries. In order to make these usable features for the models, we needed to calculate a suitable measure for them. As such, we computed the spacy distance measure between these new values and the search term, for each entry. The new "distance" features are made up of the spacy package's semantic distance between a search term and: (1) the product's description, (2) the product's title + assembled measurements, (3) the product's title + measurements (in the format length - width - depth), (4) the product's title + measurements (in the format depth - width - height), (5) the product's title + the product's color, (6) the product's title + the product's color family. The only new count feature was the count of words in the search term that appear in the item's attributes.

The hyperparameter optimization we ran is an Exhaustive Grid Search. Though we first tried a Randomized Parameter Optimization to extract insights on which parameters would be a good fit for the Grid Search. The selection of the hyperparameters and their ranges of values were motivated by the hyperparameters used in Yao-Jen Chang's code, official documentation of the model, and suggestions from tutorials of the models (Lee, 2023).

5 Results

5.1 Baseline Results

The table summarises the RMSE of the baseline model, found with vs without pre-processing the attributes with stemming before running the regressions. The increased accuracy of the model with stemmed attributes shows the importance of data pre-processing.

RMSE With Stemming	**RMSE Without Stemming**
0.4898566584108585	0.5201502440419029

5.2 Results for Multiple Regression Models

The following table shows the results for the regression models we trained, using the original set of features (stemmed word counts of title and product description, and length of query).

Model Name	**RMSE**
Random Forest	0.4899332372879737
ElasticNet	0.5344095090288571
LinearSVR	0.5179834633137584
Ridge	0.5096361511337741
BaggingRegressor	0.4896743447536259

5.3 Results for Different Features

The following tables report the performance of each model with a different set of features.

First, we tried applying all the models to a combination of count features (stemmed), and all the spacy distance models we computed. The results show that the new features seem to help the model:

Model Name	**RMSE**
Random Forest	0.48867330646784135
ElasticNet	0.5279045604987007
LinearSVR	0.6026315391478224
Ridge	0.5085377292033654
BaggingRegressor	0.48956667810834537

Then, we tried comparing the models using only the spacy distance measures.

Model Name	RMSE
Random Forest	0.5246753857611899
ElasticNet	0.5295861823055734
LinearSVR	0.5393891799194505
Ridge	0.5283737016859381
BaggingRegressor	0.5249022941118134

The results were disappointing, demonstrating that the original count features were a better estimator than the semantic distance ones we calculated.

5.4 Results for the Hyperparameter Optimisation

Due to the time required to run the grid search optimiser on the most efficient estimator – the Random Forest (and the Bagging Regressor using Random forest as an estimator), we were unfortunately not able to extract the information in time for submission. However, our hyperparameter optimisation tuning for the Ridge Regression yielded a decrease of the RMSE from the prior 0.50853... reported above to a 0.507064.

A greater improvement was achieved through hyperparameter optimisation of the Elastic Net regressor, dropping the RMSE to 0.507064. This is also the model we used to examine feature weights, the discussion for which follows below.

6 Feature Analysis and Conclusion

The following figure tabulates feature importance (top 5 features) after optimising a Ridge Regressor:

Feature Name	Coefficient
spacy_description_stemmed	0.237996
word_in_title_stemmed	0.109902
spacy_title_stem_and_assembled_measurement	0.095073
len_of_query	0.079315
spacy_title_and_measurement_DWH_stem	0.078462

From this it appears that the most predictive feature is the semantic distance (via spacy) between the search term and the product description. Please note: for the interpretations of the coefficients we used the suggestion by Brownlee (2020) to treat negative coefficients as positive, if a negative value has no particular interpretive meaning for our problem. In our case, therefore, it is best to interpret the coefficient as all positive – meaning if the coefficient's absolute value is higher, the and product is more likely to be relevant to the search term.

In total, three out of five of the distance measures seem to be among the most predictive for the model. As such, feature engineering informed by data exploration seems to be a good way to (somewhat) improve accuracy.

Overall, this report concludes that tuning a model and carefully extracting features informed by data exploration is essential to creating a more efficient estimator.

7 References

Lee, E. (2023, April 11). Hyperparameter optimization: Grid search vs. Random Search vs. bayesian optimization in action. Medium. https://drlee.io/hyperparameter-optimization-grid-search-vs-random-search-vs-bayesian-optimization-in-action-106f99b94e32

Brownlee, Jason. "How to Calculate Feature Importance with Python." MachineLearningMastery.com, 20 Aug. 2020, machinelearningmastery.com/calculate-feature-importance-with-python/.