Technical Report for NLP Relevance Prediction

1.Introduction

In this project in the area of text analytics has been proposed to predict the relevance of a result with respect to a query. Queries are represented by one or many search terms, resulting in a specific product. Based on how alike is the query and the answer, a relevance score is assigned. The higher the score the better the relevance of the answer. The relevance score takes a real number in the range [1,3] (in steps of 0.5), where 1 denotes the least possible contiguity and 3 denotes the most possible contiguity.

We had access to three datasets. The first one "train.csv" contains pairs of queries and answers and also contains the relevance score. The second one "product_descriptions.csv" which contains the products and for each one contained a textual description. The third dataset contains additional attributes for some of the products.

Our task is to predict the relevance score for an unknown combination between a query and a result (the evaluation measure that will be used is the Root-Mean-Square Error, RMSE)

2.Technical Part

2.1 Importing Packages

We start with importing the packages that we used to perform the task we were assigned to.

Some of the packages that we used are Numpy, Pandas for creating Arrays and Dataframes. Also we imported nltk.stem.porter, regex and random to perform stemming to words, regular expression operations and generate random numbers.

2.2 Loading the files as dataframes

After importing our packages that we used, we load our datasets as dataframes. Also we used the attribute.csv to create a new dataframe that contains the Brands of the products.

	product_uid	brand
9	100001.0	Simpson Strong-Tie
37	100002.0	BEHR Premium Textured DeckOver
69	100003.0	STERLING
93	100004.0	Grape Solar
122	100005.0	Delta

Figure 1.

2.3 Fixing Typos with Google.

We found a custom dictionary to fix any typos that may be contained in our text dataset, taken from open source https://www.kaggle.com/steubk/fixing-typos . So, we fix any typos that may be present in the search terms by mapping them with respect to the dictionary. This practice helps into normalizing the data, leading to more efficient results.

2.4 Merging the train, product descriptions and brands dataframe

Afterwards, we create a unified dataframe by merging the train dataframe, the product description dataframe and the newly created dataframe that contains the brands of the products.

	id	product_uid	product_title	search_term	relevance	product_description	brand
0	2	100001	Simpson Strong-Tie 12-Gauge Angle	angle bracket	3.00	Not only do angles make joints stronger, they ...	Simpson Strong-Tie
1	3	100001	Simpson Strong-Tie 12-Gauge Angle	l bracket	2.50	Not only do angles make joints stronger, they ...	Simpson Strong-Tie
2	9	100002	BEHR Premium Textured DeckOver 1-gal. #SC-141 ...	deck over	3.00	BEHR Premium Textured DECKOVER is an innovativ...	BEHR Premium Textured DeckOver
3	16	100005	Delta Vero 1-Handle Shower Only Faucet Trim Ki...	rain shower head	2.33	Update your bathroom with the Delta Vero Singl...	Delta
4	17	100005	Delta Vero 1-Handle Shower Only Faucet Trim Ki...	shower only faucet	2.67	Update your bathroom with the Delta Vero Singl...	Delta
5	18	100006	Whirlpool 1.9 cu. ft. Over the Range Convectio...	convection otr	3.00	Achieving delicious results is almost effortle...	Whirlpool
6	20	100006	Whirlpool 1.9 cu. ft. Over the Range Convectio...	microwave over stove	2.67	Achieving delicious results is almost effortle...	Whirlpool
7	21	100006	Whirlpool 1.9 cu. ft. Over the Range Convectio...	microwaves	3.00	Achieving delicious results is almost effortle...	Whirlpool
8	23	100007	Lithonia Lighting Quantum 2-Light Black LED Em...	emergency light	2.67	The Quantum Adjustable 2-Light LED Black Emerg...	Lithonia Lighting
9	27	100009	House of Fara 3/4 in. x 3 in. x 8 ft. MDF Flut...	mdf 3/4	3.00	Get the House of Fara 3/4 in. x 3 in. x 8 ft. ...	House of Fara

Figure 2.

2.5 Stemming

In this section we perform some preprocess to textual data in the dataframe to simplify units of measurement,as well as remove special characters, punctuations, text between parentheses/ brackets and sizes. After cleansing the text, we perform stemming to it.

	id	product_uid	product_title	search_term	relevance	product_description	brand
0	2	100001	simpson strong tie 12 gaug angl	angl bracket	3.00	not onli do angl make joint stronger they als...	simpson strong tie
1	3	100001	simpson strong tie 12 gaug angl	l bracket	2.50	not onli do angl make joint stronger they als...	simpson strong tie
2	9	100002	behr premium textur deckov 1 gal. sc 141 tugb...	deck over	3.00	behr premium textur deckov is an innov solid c...	behr premium textur deckov
3	16	100005	delta vero 1 handl shower onli faucet trim kit...	rain shower head	2.33	updat your bathroom with the delta vero singl ...	delta
4	17	100005	delta vero 1 handl shower onli faucet trim kit...	shower onli faucet	2.67	updat your bathroom with the delta vero singl ...	delta
5	18	100006	whirlpool 1.9 cu. ft. over the rang convect mi...	convect otr	3.00	achiev delici result is almost effortless with...	whirlpool
6	20	100006	whirlpool 1.9 cu. ft. over the rang convect mi...	microwav over stove	2.67	achiev delici result is almost effortless with...	whirlpool
7	21	100006	whirlpool 1.9 cu. ft. over the rang convect mi...	microwav	3.00	achiev delici result is almost effortless with...	whirlpool
8	23	100007	lithonia light quantum 2 light black led emerg...	emerg light	2.67	the quantum adjust 2 light led black emerg lig...	lithonia light
9	27	100009	hous of fara 3/4in. x 3in. x 8ft. mdf flute case	mdf 3/4	3.00	get the hous of fara 3/4in. x 3in. x 8ft. mdf ...	hous of fara

Figure 3.

2.6 Feature Generation

In order to use regression models for predicting relevance, we need to have data in a numerical form. With this in mind, we generate new features by creating functions for counting common words and whole words. We use these functions to create the numerical features and we save those features as new entities in the dataframe. Each newly generated feature corresponds to its associated id. The features that will help us with the modelization are the following:

- Length of the words in the columns "product_title", "search_term", "product_description" and "brand".
- Number of times the entire search term appears in the product title.
- Number of times the entire search term appears in the product description.
- Number of words that appear in search terms also appear in product titles.
- Number of words that appear in search terms also appear in production descriptions.
- Ratio of product title word length to search term word length.
- Ratio of product description word length to search term word length.
- Ratio of product title and search term common word count to search term word count.
- Ratio of product description and search term common word count to search term word count.
- Number of words that appear in the search term also appears in the brand. ● Ratio of search term and brand common word count to brand word count.

Finally, we drop the columns that can't be used for the training process. Those columns are the ones that contain textual data, as well as the product id and the id columns.

2.7 Modelization

In order to predict the relevance score, we make use of the scikit learn package. Firstly, we split the relevance column from the dataframe as we will use it as targets for the training and testing process of the models. Secondly, we randomly split the data into a train set (70% of the dataset) and a test set (30% of the dataset), and we feed them to each of the models that we chose. We are using Random Forest, Ridge Regression, Gradient Boosting for regression, XGBoosting for regression and Multi-layer Perceptron. As a metric, we are using RMSE.

2.7.1 Random Forest

Random Forest with 100 estimators and max depth 6.
RMSE score : 0.4781

```
[ ] rf = RandomForestRegressor(n_estimators=100, max_depth=6, random_state=0)
    rf.fit(X_train, y_train.values.ravel())

    RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                          max_depth=6, max_features='auto', max_leaf_nodes=None,
                          max_samples=None, min_impurity_decrease=0.0,
                          min_impurity_split=None, min_samples_leaf=1,
                          min_samples_split=2, min_weight_fraction_leaf=0.0,
                          n_estimators=100, n_jobs=None, oob_score=False,
                          random_state=0, verbose=0, warm_start=False)

[ ] y_pred = rf.predict(X_test)
    rf_mse = mean_squared_error(y_pred, y_test)
    rf_rmse = np.sqrt(rf_mse)
    print('RandomForest RMSE: %.4f' % rf_rmse)

    RandomForest RMSE: 0.4781
```

Figure 4.

2.7.2 Ridge Regression

Ridge Regression with alpha 0.1
RMSE score : 0.4847

```
[ ] rg= Ridge(alpha=.1)
    rg.fit(X_train, y_train.values.ravel())

    Ridge(alpha=0.1, copy_X=True, fit_intercept=True, max_iter=None,
          normalize=False, random_state=None, solver='auto', tol=0.001)

[ ] y_pred = rg.predict(X_test)
    rg_mse = mean_squared_error(y_pred, y_test)
    rg_rmse = np.sqrt(rg_mse)
    print('Ridge RMSE: %.4f' % rg_rmse)

    Ridge RMSE: 0.4847
```

Figure 5.

2.7.3 Gradient Boosting

Gradient Boosting with 100 estimators, learning rate 0.1 and, max depth 1.
RMSE score : 0.4820

```
[ ] gbr = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=1, random_state=0, loss='ls').fit(X_train, y_train.values.ravel())

[ ] y_pred = gbr.predict(X_test)
    gbr_mse = mean_squared_error(y_pred, y_test)
    gbr_rmse = np.sqrt(gbr_mse)
    print('Gradient boosting RMSE: %.4f' % gbr_rmse)

    Gradient boosting RMSE: 0.4820
```

Figure 6.

2.7.4 XGBoosting Regression

XGBoost Regression with 100 estimators, learning rate 0.08 and max depth 7.
RMSE score : 0.4768

```
[ ]  xgb = xgboost.XGBRegressor(n_estimators=100, learning_rate=0.08, gamma=0, subsample=0.75, colsample_bytree=1, max_depth=7)
     xgb.fit(X_train, y_train.values.ravel())

     [12:28:42] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
     XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                  colsample_bynode=1, colsample_bytree=1, gamma=0,
                  importance_type='gain', learning_rate=0.08, max_delta_step=0,
                  max_depth=7, min_child_weight=1, missing=None, n_estimators=100,
                  n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
                  reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
                  silent=None, subsample=0.75, verbosity=1)

[ ]  y_pred = xgb.predict(X_test)
     xgb_mse = mean_squared_error(y_pred, y_test)
     xgb_rmse = np.sqrt(xgb_mse)
     print('Xgboost RMSE: %.4f' % xgb_rmse)

     Xgboost RMSE: 0.4768
```

Figure 7.

2.7.5 Multi-Layer Perceptron

Multi-Layer Perceptron with hidden layer size 20, relu activation and 1000 iterations.
RMSE score : 04859

```
[ ]  mlpreg = MLPRegressor(hidden_layer_sizes=(20,), activation='relu',
               solver='adam', alpha=0.001, batch_size='auto',
               learning_rate='adaptive', learning_rate_init=0.01,
               power_t=0.5, max_iter=1000, shuffle=True, random_state=0,
               tol=0.0001, verbose=False, warm_start=False, momentum=0.9,
               nesterovs_momentum=True, early_stopping=False,
               validation_fraction=0.1, beta_1=0.9, beta_2=0.999,
               epsilon=1e-08)
     mlpreg.fit(X_train, y_train.values.ravel())

     MLPRegressor(activation='relu', alpha=0.001, batch_size='auto', beta_1=0.9,
                  beta_2=0.999, early_stopping=False, epsilon=1e-08,
                  hidden_layer_sizes=(20,), learning_rate='adaptive',
                  learning_rate_init=0.01, max_fun=15000, max_iter=1000,
                  momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
                  power_t=0.5, random_state=0, shuffle=True, solver='adam',
                  tol=0.0001, validation_fraction=0.1, verbose=False,
                  warm_start=False)

[ ]  y_pred = mlpreg.predict(X_test)
     mlpreg_mse = mean_squared_error(y_pred, y_test)
     mlpreg_rmse = np.sqrt(mlpreg_mse)
     print('MLPRegressor RMSE: %.4f' % mlpreg_rmse)

     MLPRegressor RMSE: 0.4859
```

Figure 8.

3. Final Results/ Conclusion

After training and testing our models, we conclude that the best model for predicting the relevance score for an unknown combination between a query and a result is XGBoost RMSE of 0.477. We can easily see that from the following barplot comparing each model's RMSE.

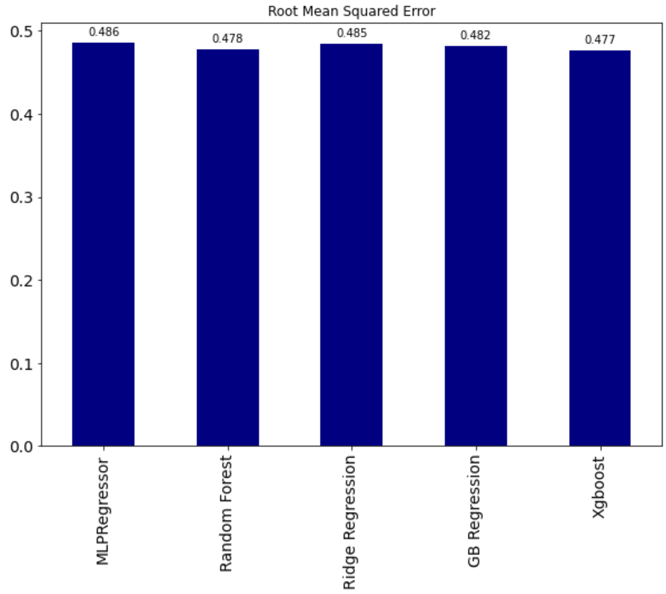

Figure 9.

Since XGBoost is the best estimator, we use a built-in function called plot_importance() to plot features ordered by their importance. We observe that the most significant features are the length of words for the description, the ratio of product description word length to search term word length and the length of words of the title.

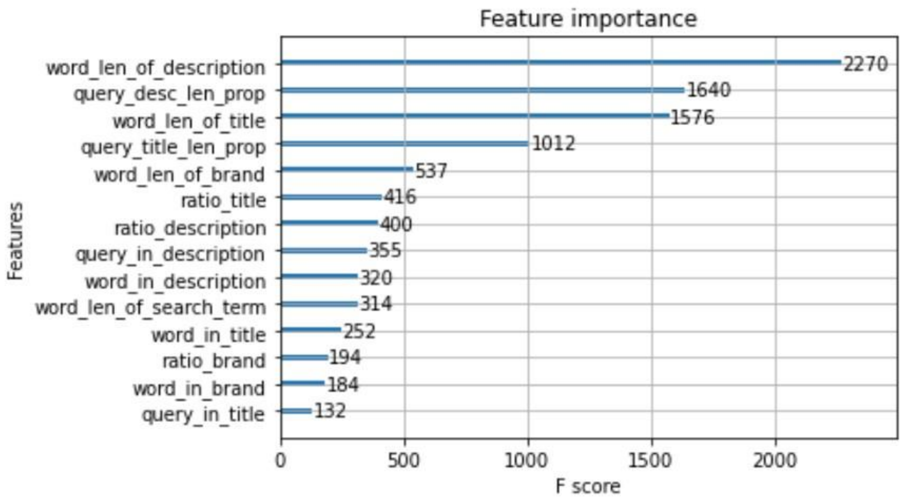

Figure 10.