Team Member : Sipeng Wang, Jiafang Liu

Project Title : **Search Relevance**

1. Introduction

Our project is from Kaggle competition. Shoppers rely on Home Depot's product authority to find and buy the latest products and to get timely solutions to their home improvement 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 our project, we suppose to use NLP technology and network to help Home Depot improve their customers' shopping experience by developing a model that can accurately predict the relevance of search results.

Our major goal is to predict a relevance score for the provided combinations of search terms and products. Search relevancy is an implicit measure Home Depot uses to gauge how quickly they can get customers to the right products. Currently, human raters evaluate the impact of potential changes to their search algorithms, which is a slow and subjective process. By removing or minimizing human input in search relevance evaluation, Home Depot hopes to increase the number of iterations their team can perform on the current search algorithms.

2. Data set analysis

2.1. View Data

When we initially get raw data from the Home Depot Competition. The are four files of interest for exploration (train.csv, test.csv, product_description.csv, attributes.csv) . We need to see the content and architecture the theses data.

	id	product_uid	product_title	search_term	relevance
0	2	100001	Simpson Strong-Tie 12-Gauge Angle	angle bracket	3.00
1	3	100001	Simpson Strong-Tie 12-Gauge Angle	l bracket	2.50
2	9	100002	BEHR Premium Textured DeckOver 1-gal. #SC-141 ...	deck over	3.00
3	16	100005	Delta Vero 1-Handle Shower Only Faucet Trim Ki...	rain shower head	2.33
4	17	100005	Delta Vero 1-Handle Shower Only Faucet Trim Ki...	shower only faucet	2.67

train.csv

	product_uid	product_description
0	100001	Not only do angles make joints stronger, they ...
1	100002	BEHR Premium Textured DECKOVER is an innovativ...
2	100003	Classic architecture meets contemporary design...
3	100004	The Grape Solar 265-Watt Polycrystalline PV So...
4	100005	Update your bathroom with the Delta Vero Singl...

product_description.csv

For easily clean all the data, we concatenated all the data together

	id	product_title	product_uid	relevance	search_term	product_description
0	2	Simpson Strong-Tie 12-Gauge Angle	100001	3.00	angle bracket	Not only do angles make joints stronger, they ...
1	3	Simpson Strong-Tie 12-Gauge Angle	100001	2.50	l bracket	Not only do angles make joints stronger, they ...
2	9	BEHR Premium Textured DeckOver 1-gal. #SC-141 ...	100002	3.00	deck over	BEHR Premium Textured DECKOVER is an innovativ...
3	16	Delta Vero 1-Handle Shower Only Faucet Trim Ki...	100005	2.33	rain shower head	Update your bathroom with the Delta Vero Singl...
4	17	Delta Vero 1-Handle Shower Only Faucet Trim Ki...	100005	2.67	shower only faucet	Update your bathroom with the Delta Vero Singl...

concat.csv

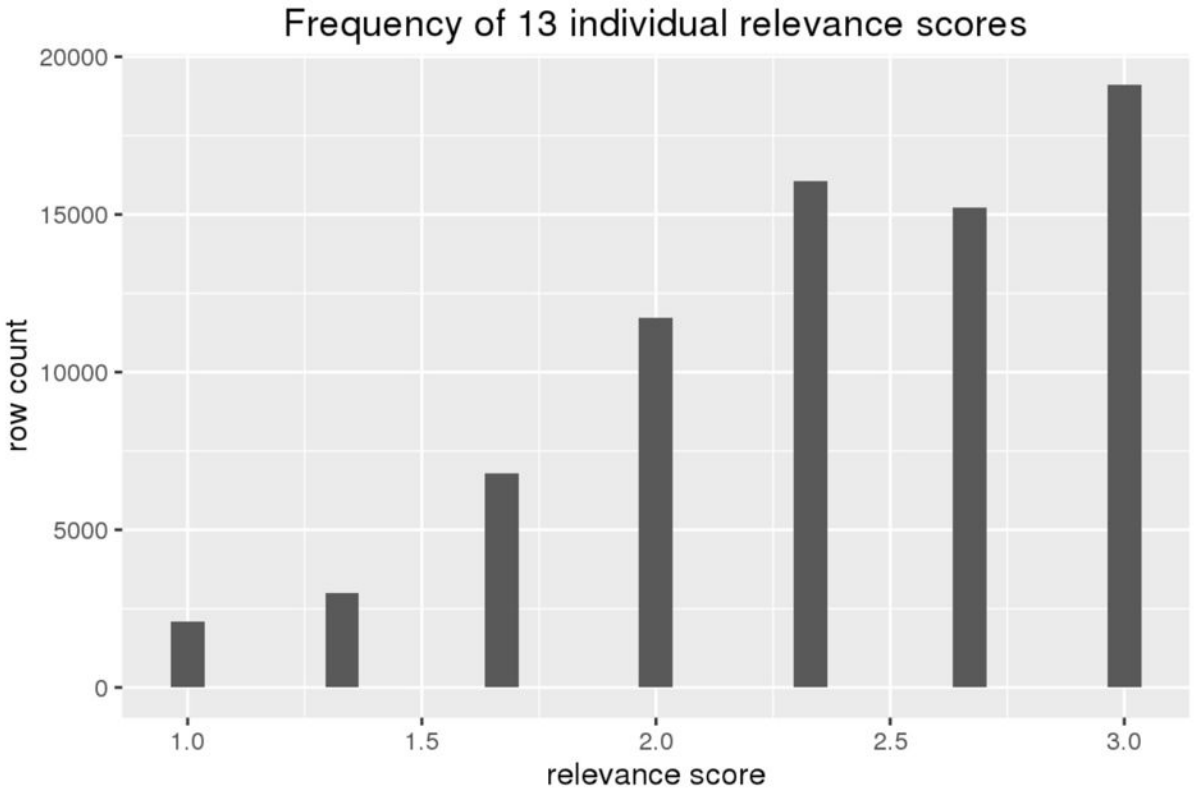
overlap of train data and test data: There are 74067 rows in the train.csv data and 166693 in the test.csv data. Both have the same columns [id, product_uid, product_title, search_term, relevance] bar the relevance score which is not included in test.csv data. After viewing the column product_id, we found that there are 54667 unique values in the train.csv data and 97460 in the test.csv data. We also can see that the intersect data is 27667 product_uid values. Following is the intersection .

We can find that there are only 27699 product_uid in common between the total of 54667 product_uid in train.csv and 97460 in test.csv data. So the total number of unique product_uid is equal the number of product_uid in the product_descriptions.csv file.

The target variable is the relevance score. In the supplied relevance_instructions.cocx file supplied to the human graders it instructs to provide a relevance grade of either:

- Irrelevant (1)
- Partially or somewhat relevant (2)
- Perfect match (3).

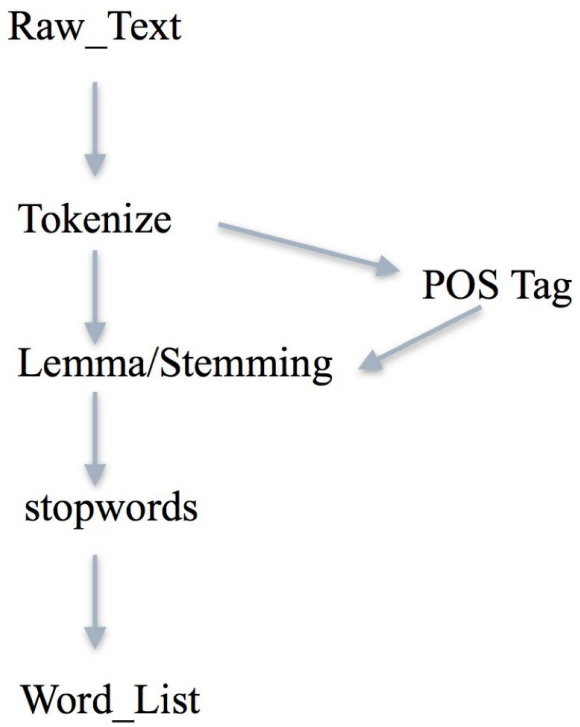
Furthermore in the competition data it states that *Each pair [search_term,product] was evaluated by at least three human raters. The provided relevance scores are the average value of the ratings*. There are a total of 13 unique values for **relevance** in the *train* data. A bar-chart of their frequency is displayed below.



Note there are a set of **relevance** scores that occur with low frequencies (1.25,1.5,1.75,2.25,2.5,2.75), thus indicating that perhaps more than 3 or less than 3 human evaluators appraised that row.

2.2. Data Preprocessed

Because Machine can not read all the text and train raw data together, we need to do some proprocesses for the raw data we download from kaggle official website. How to choose a data cleaning process which has significant influence to our final results. We use following typical data cleaning process flow(tokenize the raw data, attach pos tag, Lemmanize/ stem, remove stop words and meaningless words) to generate a cleaning word_list.



Data Clean Flow

3. Implementation and Training

3.1. Add features.

- Simple features

At first we thought and implemented some very simple features which will improve the training accuracy. First one is the length of search term that inputted by users. Second one is how many common words between search term and product title. Last one we can get easily is how many common words between search term and product description. Following the result we generate simple features.

	id	product_title	product_uid	relevance	search_term	product_description	len_of_query	commons_in_title	commons_in_desc
0	2	simpson strong-ti 12-gaug angl	100001	3.00	angl bracket	not on do angl make joint stronger, they also ...	2	1	1
1	3	simpson strong-ti 12-gaug angl	100001	2.50	l bracket	not on do angl make joint stronger, they also ...	2	1	1
2	9	behr premium textur deckov 1-gal. #sc-141 tugb...	100002	3.00	deck over	behr premium textur deckov is an innov solid c...	2	1	1
3	16	delta vero 1-handl shower on faucet trim kit i...	100005	2.33	rain shower head	updat your bathroom with the delta vero single...	3	1	1
4	17	delta vero 1-handl shower on faucet trim kit i...	100005	2.67	shower onli faucet	updat your bathroom with the delta vero single...	3	2	2

Simple features

- Levenshtein

Levenshtein distance (LD) is a measure of the similarity between two strings, which we will refer to as the source string (s) and the target string (t). The distance is the number of deletions, insertions, or substitutions required to transform s into t. For example:

1. If s is "test" and t is "test", then LD(s,t) = 0, because no transformations are needed. The strings are already identical.
2. If s is "test" and t is "tent", then LD(s,t) = 1, because one substitution (change "s" to "n") is sufficient to transform s into t.

In other word, the greater the Levenshtein distance, the more different the strings are.

In our project, we compared the users' research terms with product titles and product descriptions. Here we use levenshtein ratio to represent the similarity. The levenshtein ratio is computed as (lensum - ldist) / lensum where lensum is the sum of two strings' length and ldist is the levenshtein distance. By using normalized levenshtein ratio, it's better to reflect the similarity between two strings.

	search_term	product_uid	product_title	product_description	dist_in_title	dist_in_desc
0	angl bracket	100001	simpson strong-ti 12-gaug angl	not onli do angl make joint stronger, they als...	0.190476	0.030418
1	l bracket	100001	simpson strong-ti 12-gaug angl	not onli do angl make joint stronger, they als...	0.153846	0.022901
2	deck over	100002	behr premium textur deckov 1-gal. #sc-141 tugb...	behr premium textur deckov is an innov solid c...	0.175000	0.017875
3	rain shower head	100005	delta vero 1-handl shower on faucet trim kit i...	updat your bathroom with the delta vero single...	0.333333	0.048632
4	shower onli faucet	100005	delta vero 1-handl shower on faucet trim kit i...	updat your bathroom with the delta vero single...	0.347826	0.054545

Default data and levenshtein ratios

- TF-IDF

TF: Term Frequency, which measures how frequently a term occurs in a document. TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

IDF: Inverse Document Frequency, which measures how important a term is. IDF(t) = log_e(Total number of documents / Number of documents with term t in it).

TF-IDF can be computed as TF * IDF. Here is an example:

Consider a document containing 100 words wherein the word cat appears 3 times. The term frequency (i.e., tf) for cat is then (3 / 100) = 0.03. Now, assume we have 10 million documents and the word cat appears in one thousand of these. Then, the inverse document frequency (i.e., idf) is calculated as log(10,000,000 / 1,000) = 4. Thus, the Tf-idf weight is the product of these quantities: 0.03 * 4 = 0.12.

The advantage of TF-iDF is that this model focus on the less frequent words shown in corpus. Because the common words in corpus contain less information than unique words, TF-iDF model reduced weight of common words in search term and product title/description.

	search_term	product_uid	product_title	product_description	tfidf_cos_sim_in_title	tfidf_cos_sim_in_desc
0	angl bracket	100001	simpson strong-ti 12-gaug angl	not onli do angl make joint stronger, they als...	0.274539	0.182825
1	l bracket	100001	simpson strong-ti 12-gaug angl	not onli do angl make joint stronger, they als...	0.000000	0.000000
2	deck over	100002	behr premium textur deckov 1-gal. #sc-141 tugb...	behr premium textur deckov is an innov solid c...	0.000000	0.053480
3	rain shower head	100005	delta vero 1-handl shower on faucet trim kit i...	updat your bathroom with the delta vero single...	0.135885	0.043676
4	shower onli faucet	100005	delta vero 1-handl shower on faucet trim kit i...	updat your bathroom with the delta vero single...	0.304622	0.097910

Default data and cosine similarities of TF-iDF

- Word2Vec

Word2vec (w2v) is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the

corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space.

In the best condition, we can use well-trained w2v models, like Google's pre-trained model. For doing more exercise, we trained our own model by this dataset. The w2v is different from TF-iDF, it need word-level tokenization rather than sentence-level tokenization. So we have to tokenize our corpus into words. After tokenization, we feed corput to the w2v model and train it. The model can convert a word to a multi-dimensions vector.

After we get the word vector, we use cosine similarity to calculate similarity between search term and product title or description.

	search_term	product_uid	product_title	product_description	w2v_cos_sim_in_title	w2v_cos_sim_in_desc
0	angl bracket	100001	simpson strong-ti 12-gaug angl	not onli do angl make joint stronger, they als...	0.462966	0.389653
1	l bracket	100001	simpson strong-ti 12-gaug angl	not onli do angl make joint stronger, they als...	0.305487	0.089469
2	deck over	100002	behr premium textur deckov 1-gal. #sc-141 tugb...	behr premium textur deckov is an innov solid c...	0.338075	0.464262
3	rain shower head	100005	delta vero 1-handl shower on faucet trim kit i...	updat your bathroom with the delta vero single...	0.550397	0.471239
4	shower onli faucet	100005	delta vero 1-handl shower on faucet trim kit i...	updat your bathroom with the delta vero single...	0.677524	0.482417

Default data and cosine similarities of Word2Vec

- Removing text and rebuilding train/test dataset

Before we start to train, we have to remove all text content. Then we also need to separate our labels (relevance) from dataset.

After implementing all above operation, we generate following new training data.

	Unnamed: 0	product_uid	dist_in_title	dist_in_desc	tfidf_cos_sim_in_title	tfidf_cos_sim_in_desc	w2v_cos_sim_in_title	w2v_cos_sim_in_desc
1	1	100001	0.153846	0.022901	0.000000	0.000000	0.305487	0.089469
2	2	100002	0.175000	0.017875	0.000000	0.053480	0.338075	0.464262
3	3	100005	0.333333	0.048632	0.135885	0.043676	0.550397	0.471239
4	4	100005	0.347826	0.054545	0.304622	0.097910	0.677524	0.482417
5	5	100006	0.222222	0.007542	0.212003	0.081471	0.529997	0.316039

All Features

3.2. Training algorithm

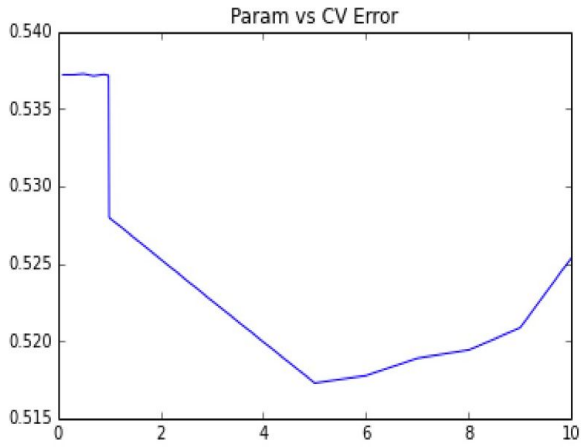
We choose the three training model to train our data. **Bagging(Breiman, 1996)** Fit many large trees to booststrap-resampled versions of the training data, and classify by majority vote. Random Forest **(Freund & shapire, 1996)** Fancier version of bagging, add one more randomness in variable choosing. **Boostring(Breiman, 19999)** Fit many large or small trees to re-weighted version of the training data. Classify by weighted majority vote. After choosing algorithm, we need to test different configs for training step to generate a parameters- following error chart. Finally we compare different config parameters to get an optimal training model.

4. Results

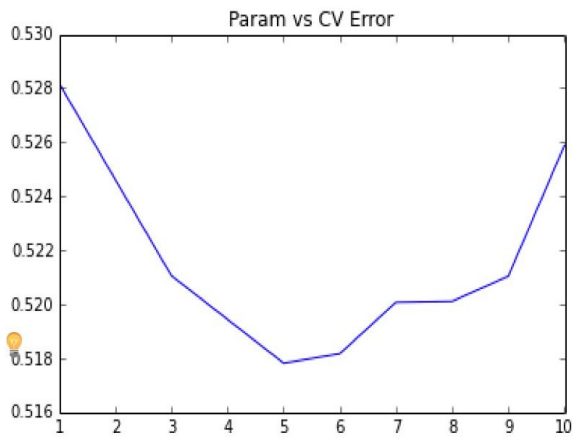
- Parameters of models

Here we test different models with various parameters of model.
XGBoosting and RandomForest we tested different max depth of the tree. And for Ridge regressor, we test which alpha better. This part will help us to find the best regressor for our project.

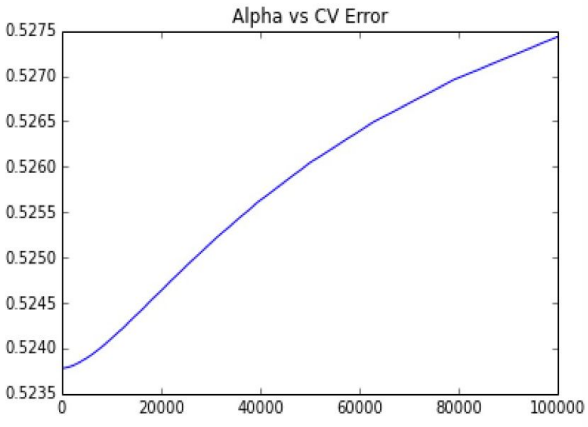
Simple features Data training result:



XGBoost

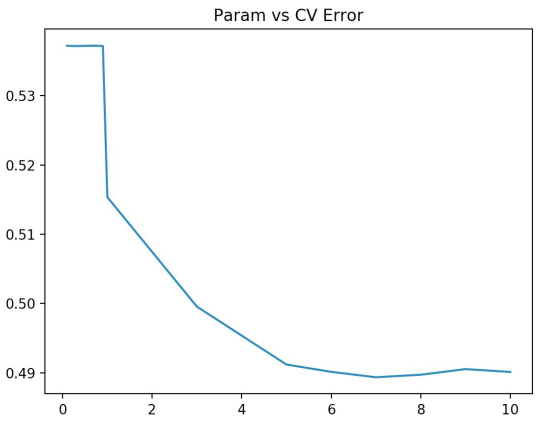


RandomForest

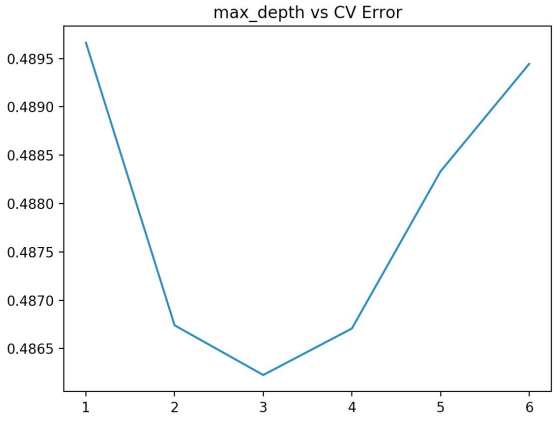


Ridge

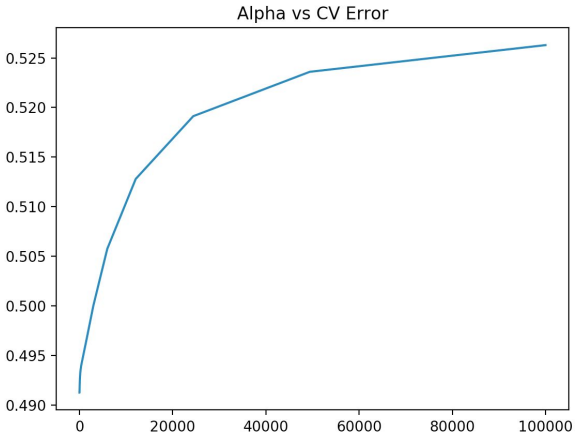
Concat all features as training data：



RandomForest



Xgbooster



Ridge

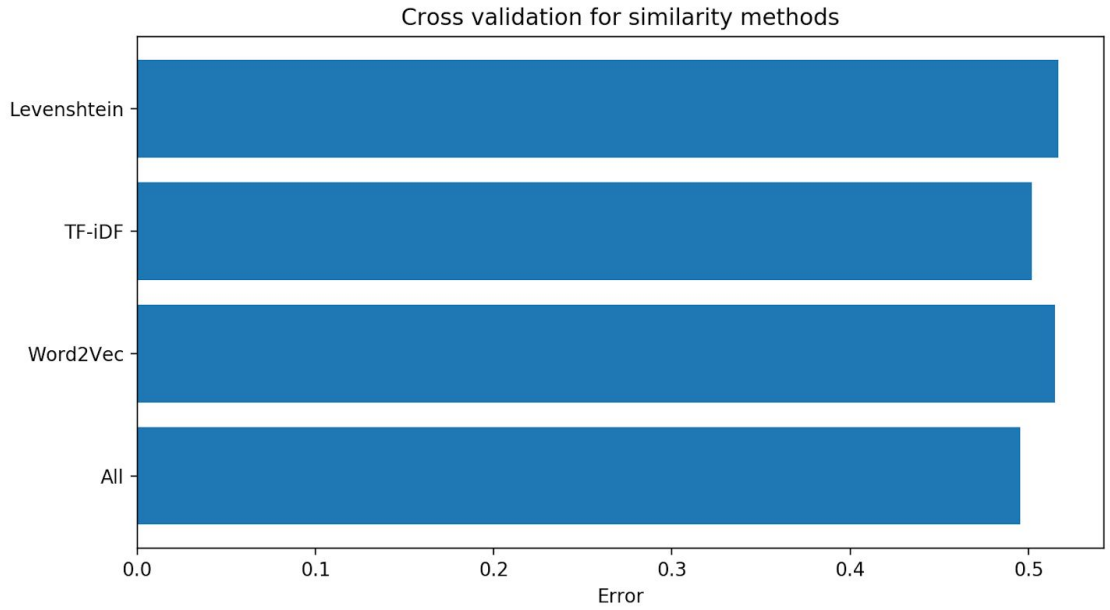
Now we list the lowest error for each regressor with the best parameters. The result is as following:

Training Algorithm	RandomForest	Ridge Regression	XGBoost
Error(simple)	0.518	0.523	0.524
Error(all)	0.488	0.491	0.486

Different regressors to train simple/all model

- Performance of each methods

Here we tested the effect of each similarity model. We used each method independently to train the network. The error is shown as following:



XGBoost regressor to train different similarity methods

As we can see Tf-iDF method is better than Levenshtein and Word2Vec, but they don't have much difference. In my expectation, Word2Vec should be the best method in all three methods. I think the reason why it doesn't have a good performance is the corpus. We trained Word2Vec by using our project dataset. This corpus (Home-depot dataset) is not comprehensive and small.

- The accuracy of best model

What we did is predictions, so we have to define a range that the result dropping in this range is

correct. So we defined two range:

1. the difference of predicted relevance and true relevance is in 0.5.
2. the difference of predicted relevance and true relevance is in 0.1.

We compute the proportion of the correct relevance. The formula is as following:

$$accuracy(threshold) = \frac{number\ of\ abs(y\_true - y\_pred) < threshold}{number\ of\ samples}$$

The accuracy of our best model:

accuracy(0.5) = 65.8%
accuracy(0.1) = 12.7%

Above all test, our model's accuracy is not very high, but it still can predict an approximate relevance between search term and the product system returns.