Home Depot Product Search Relevance

Ehsan RahimiNasab, Yu Wen, Shifu Xu

Northeastern University
Machine Learning Project Proposal
Faculty Supervisor: Lu Wang

Abstract

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 this project we want to help Home Depot to improve their customers' shopping experience by developing a model that can accurately predict the relevance of search results. Then this model can be used in order to rank the search results according to their relevancy score, and thus it will help Home Depot's search engine to have a better performance.

I. Introduction

Learning to rank or machine-learned ranking (MLR) is the application of machine learning, typically supervised, semi-supervised or reinforcement learning, in the construction of ranking models for information retrieval systems. Training data consists of lists of items with some partial order specified between items in each list. This order is typically induced by giving a numerical or ordinal score or a binary judgment (e.g. "relevant" or "not relevant") for each item. The ranking model's purpose is to rank, i.e. produce a permutation of items in new, unseen lists in a way which is "similar" to rankings in the training data in some sense.

Ranking is a central part of many information retrieval problems, such as document retrieval, collaborative filtering, sentiment analysis, and online advertising. One of them which is the underlying problem of our project is designing a model for query-document pairs where each pair has a relevancy score. In this problem training data consists of queries and documents matching them together with relevance degree of each match. It may be prepared manually by human assessors (Like Home Depot), who check results for some queries and determine relevance of each result. It is not feasible to check relevance of all documents, and so typically a technique called pooling is used, only the top few documents, retrieved by some existing ranking models are checked. Alternatively, training data may be derived automatically by analyzing clickthrough logs (i.e. search results which got clicks from users), query chains, or such search engines' features as Google's SearchWiki.

The details of the problem that we want to tackle are as follows:
In Kaggle website, there is a problem about helping the Home Depot company to improve the relevancy between their search key words and search results. In this problem, data is given in the form of (q, p, s), where q is the query word, p is the product returned by the search algorithm and s is the relevance score of q and p rated by a human rater. Clearly, given any q, the search algorithm can return many p and each pair of (q, p) will assign a relevance score. If according to the quality of q, p could be the best search result, in this case, s should have a high value in the tuple

(q, p, s). On other hand, if p is not relevant to the true search query due to some reason, in this case, s should have a low value. With the help of these data, the search engine can quickly return the relevant results to users even though their query word contains some typos or low-detailed descriptions.
The goal of this project is to build a model which can perfectly predict the relevance score between q and p. The training data is given by the Kaggle which contains the tuples (q, p, s) mentioned above, where s is a relevance score given by an human rater and is viewed as the ground truth. After a model have been successfully trained, the input of this model would be some pair like (q, p) and the output will be s which is considered as an automatically generated relevance score.

There are several approaches toward this problem, the one that is relevant to our goal is called **Pointwise approach** in which it is assumed that each query-document pair in the training data has a numerical or ordinal score. Then learning-to-rank problem can be approximated by a regression problem –given a single query-document pair, predict its score. A number of existing supervised machine learning algorithms can be readily used for this purpose. Ordinal regression and classification algorithms can also be used in pointwise approach when they are used to predict score of a single query-document pair, and it takes a small, finite number of values.

Other approaches toward this problem are **Pairwise approach** and **Listwise approach** which in the first one, learning-to-rank problem is approximated by a classification problem – learning a binary classifier that can tell which document is better in a given pair of documents, and the other one tries to directly optimize the value of one of the evaluation measures, averaged over all queries in the training data. This is difficult because most evaluation measures are not continuous functions with respect to ranking model's parameters, and so continuous approximations or bounds on evaluation measures have to be used.

In this project we pursue the Pointwise approach.

II. Related Work

A partial list of published learning-to-rank algorithms with poitwise approach is shown below with years of first publication of each method:

Table 1: *Pointwise learning-to-rank algorithms*

Year	Name	Notes
1989	OPRF	Polynomial regression
1992	SLR	Staged logistic regression
2002	Pranking	Ordinal regression.
2007	McRank	
2010	CRR	

In **OPRF** they show that any approach to developing optimum retrieval functions is based on two kinds of assumptions: first, a certain form of representation for documents and requests, and second, additional simplifying assumptions that predefine the type of the retrieval function. Then they describe an approach for the development of optimum polynomial retrieval functions: request-document pairs (fl, dm) are mapped onto description vectors $x(fl, dm)$, and a polynomial function $e(x)$ is developed such that it yields estimates of the probability of relevance $P(R|x(fl, dm))$ with minimum square errors.

In **SLR**, they claim that previously explored methods for computing a ranking had involved the use of statistical independence assumptions and multiple regression analysis on a learning sample. They state that in their work those techniques are recombined in a new way to achieve greater accuracy of probabilistic estimate without undue additional computational complexity. The novel element of the proposed design is that the regression analysis be carried out in two or more levels or stages. Such an approach allows composite or grouped retrieval clues to be analyzed in an orderly manner – first within groups, and then between. It compensates automatically for systematic biases introduced by the statistical simplifying assumptions, and gives rise to search algorithms of reasonable computational efficiency.

Pranking, discusses the problem of ranking instances. In their framework each instance is associated with a rank or a rating, which is an integer from 1 to k. Their goal is to find a rank-prediction rule that assigns each instance a rank which is as close as possible to the instance's true rank. They describe a simple and efficient online algorithm, analyze its performance in the mistake bound model, and prove its correctness. They describe two sets of experiments, with synthetic data and with the EachMovie dataset for collaborative filtering. In the experiments they performed, their algorithm outperforms online algorithms for regression and classification applied to ranking.

In **McRank**, they propose using the Expected Relevance to convert the class probabilities into ranking scores. The class probabilities are learned using a gradient boosting tree algorithm. Evaluations on large-scale datasets show that their approach can improve LambdaRank and the regressions-based ranker, in terms of the (normalized) DCG scores.

In **CRR**, they have presented a combined regression and ranking method, that gives strong performance on both regression and ranking metrics. The use of stochastic gradient descent makes the algorithm easy to implement, and efficient for use on large-scale data sets. They have found that CRR is especially effective on minority class distributions, and have demonstrated its applicability to the problem of CTR prediction in sponsored search.

III. Datasets

The dataset is from a Kaggle competition, which contains a number of products and real customer search terms from Home Depot's website. The challenge is to predict a relevance score for the provided combinations of search terms and products. To create the ground truth labels, Home Depot has crowdsourced the search/product pairs to multiple human raters.

The relevance is a number between 1 (not relevant) to 3 (highly relevant). For example, a search for "AA battery" would be considered highly relevant to a pack of size AA batteries (relevance = 3), mildly relevant to a cordless drill battery (relevance = 2), and not relevant to a snow shovel (relevance = 1).

Each pair was evaluated by at least three human raters. The provided relevance scores are the average value of the ratings. There are three additional things to know about the ratings:

- The specific instructions given to the raters is provided in relevance_instructions.docx.
- Raters did not have access to the attributes.
- Raters had access to product images, while the competition does not include images.

Our task is to predict the relevance for each pair listed in the test set. Note that the test set contains both seen and unseen search terms.

File descriptions

- **train.csv** - the training set, contains products, searches, and relevance scores. This file contains **74,067** training examples.
- **test.csv** - the test set, contains products and searches. You must predict the relevance for these pairs. This file contains **166,693** test examples.
- **product_descriptions.csv** - contains a text description of each product. You may join this table to the training or test set via the product_uid. There are **124,428** records in this dataset.

- **attributes.csv** - provides extended information about a subset of the products (typically representing detailed technical specifications). Not every product will have attributes.
- **sample_submission.csv** - a file showing the correct submission format.
- **relevance_instructions.docx** - the instructions provided to human raters.

Data fields

- **id** - a unique Id field which represents a (search_term, product_uid) pair.
- **product_uid** -an id for the products.
- **product_title** - he product title.
- **product_description** - the text description of the product (may contain HTML content)
- **search_term** - the search query.
- **relevance** - the average of the relevance ratings for a given id.
- **name** - an attribute name.
- **value** - the attribute's value.

IV. Methodology

In this project we want to use a supervised learning algorithm which classifies under the regression category. For first attempt, we want to use **ElasticNet** regression model. ElasticNet is hybrid of Lasso and Ridge Regression techniques. It is trained with L_1 and L_2 prior as regularizer. Elastic-net is useful when there are multiple features which are correlated.

In ElasticNet model we have:

$$\hat{\beta} = \arg\min_{\beta}(||y - X\beta||^2 + \lambda_2||\beta||^2 + \lambda_1||\beta||_1) \tag{1}$$

V. Evaluation

For each id in the test set, you must predict a relevance. This is a real number in [1,3]. The file should contain a header and have the following format:

Table 2: *Output format*

id	relevance
1	1
4	2
5	3
etc	

Submissions are evaluated on the root mean squared error (RMSE). The use of RMSE is very common and it makes an excellent general purpose error metric for numerical predictions. Compared to the similar Mean Absolute Error, RMSE amplifies and severely punishes large errors.

$$RSME = \sqrt{1/n \sum_{i}^{n}(y_i - \hat{y}_i)^2} \tag{2}$$

Again, in this project, we can't use the data in the test set to evaluate the result by ourselves since the true label is hidden by the host, but we can divide the training set into two subsets, using the smaller subset to be our test set. Moreover, we can submit our result and see the final score which Kaggle calculates for us.

References

[Kaggle Competition, 2016] Predict the relevance of search results on homedepot.com. Kaggle Home Depot Competition.

[Fuhr (1989)] Fuhr, Norbert (1989). Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Transactions on Information Systems 7*, (3): 183âĂŞ204.

[Cooper; Frederic; Dabney (1992)] Cooper, William S.; Gey, Frederic C.; Dabney, Daniel P. (1992). Probabilistic retrieval based on staged logistic regression. *SIGIR '92 Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, 98âĂŞ210.

[Crammer , Singer(2001)] Koby Crammer , Yoram Singer (2001). Pranking with Ranking. *Advances in Neural Information Processing Systems 14*, 208 - 5.

[Li, Burges, Wu(2008)] P. Li, C.J.C. Burges, and Q. Wu (2008). Learning to Rank Using Classification and Gradient Boosting. *Advances in Neural Information Processing Systems 20*, Microsoft Research, MSR-TR-2007-74.

[Sculley(2010)] D. Sculley (2010). Combined Regression and Ranking. *Google, Inc.*.

[Wikipedia, 2016] Learning to rank. Wikipedia.