Distributed Intelligent
Systems and Technologies
8-10 June 2009 • St. Petersburg, Russia

COMPARISON OF MACHINE LEARNING TECHNIQUES FOR DOCUMENT RANKING PROBLEM

Mikhail Kalinkin[1], Juliana Kiseleva[2], Nikolay Vyahhi[2], Bernhard Lang[1]

(1) OOO Siemens, Corporate Technology, Monitoring and Preventive Control,
(2) St. Petersburg State University
mkalinkin@yandex.ru

Abstract

The document ranking problem is a well-known problem which appears for example in search engines. Once a search engine has identified a set of potentially relevant documents it faces the problem to determine which of them are more relevant and which are less, it means to range this documents by relevancy. This is typically done by assigning a numerical score to each document based on a ranking function, which incorporates features of the document, the query, and the overall document collection. Main issues of determining ranking function are high dimensionality of feature space and large amount of data. We present applications of several machine learning techniques (including median statistics, k-nearest neighbor, linear regression, neural networks) to obtaining ranking function from the training dataset, and then we estimate quality of this function using test set. We discuss advantages and disadvantages of each technique and compare them by efficiency and accuracy. Both learning and testing data sets are real-world data provided by Yandex assessors.

INTRODUCTION

Information Retrieval is a field of Computer Science that deals with the automated storage and retrieval of documents [10, 18]. Web search engines, tools designed to search for relevant information in internet, are very important users of Information Retrieval theory and algorithms. Search results are usually presented as a ranked (sorted) list, which is constructed using special ranking function. If the document is more appropriate to the particular query then it gets higher relevance value from the ranking function for this query and will be higher in the list.

Web search queries can have up to thousands or millions results, but users definitely will not look on all of them to find what they are interested in. Actually, most internet users do not read more than one page of search results. Therefore, it is much more important for the search engines to output relevant results within the few top positions, then to range pages correctly on positions on thousandth page.

The contribution of this study is to compare variety of machine learning techniques for the problem of finding document ranking function with a good predictive accuracy. Formal problem description and detailed description of score function will be given in Section 2. We compare such techniques as average and median statistics (Section 4.1), k-nearest neighbor (Section 4.2), linear and quadratic regression (Section 4.3) and feed-forward neural networks (Section 4.4) providing their short description and experimental results with different learning parameters.

DOCUMENT RANKING PROBLEM AND DATA DESCRIPTION

The document ranking problem is to find a ranking function from learning set with known relevancies. Learning set usually contains feature values for query-document pairs and resulting relevance made by real assessors. This features can be any property of current query or current document or even both, like for example PageRank, tf*idf and query length, but this meta information is usually unknown to the algorithm.

We use sets made by Yandex assessors which contain 245 features for each query-document pair. All values of features are floats from 0 to 1 inclusive and relevancies are floats from 0 to 4 inclusive. Learning set has 9124 different queries and 97290 pairs total.

Scoring function

The resulting ranking function can be excellent or worthless, or something in the middle. There are a lot of different ways to estimate its accuracy using a test set. Most of them use test set which have the same format as the learning set. We use the following scoring algorithm which is pretty common:

1. We rank documents inside each query in test set using new ranking function. If some documents have the same resulting relevance, then documents with better real assessor’s relevance will be lower in rank.
2. For each query we calculate Discounted Cumulative Gain (DCG) value by the following formula:

$$DCG = \sum_i \frac{rel_i}{\log_2 i + 1} \tag{1}$$

where rel_i is a relevance from real assessors for i-th document in the resulting ranked list.

3. After all, we calculate the average value of DCGs for all queries in test set.

As much such average DCG is, as close the new ranking function results are to real assessor's results. Using machine learning techniques we try to maximize this DCG value and get the best ranking function for given learning set.

Test set that we use has 2026 queries and 21103 pairs total.

RELATED WORK

Formally the problem of finding ranking function can be considered as regression problem, classification problem or ordinal regression problem. The latter is a type of learning ranking (ordering) on instances and has properties of both classification and regression [15]. Several methods for this types of problems were investigated during past decade. They include perceptron [4], neural network with gradient descent [3], support vector machine [9], boosting algorithm [7], regression trees [11] and others.

There are good fundamental works about machine learning techniques such as [2, 14]. Detailed descriptions of most common approaches can be found there, but every technique has weak and strong sides depending on the problem. Therefore it is needed to have practical comparison of machine learning techniques for every particular problem or set of related problems. There are some papers with such comparison like [1] for phishing detection, [19] for IP traffic flow classification, [12] for spam e-mail categorization or [8] for link completion.

MACHINE LEARNING TECHNIQUES FOR RANKING PROBLEM

The current section is the main contribution of this paper, where we shortly describe used techniques and compare their results for particular document ranking problem. This techniques are average and median statistics (Section 4.1), k-nearest neighbor (Section 4.2), linear and quadratic regression (Section 4.3) and feed-forward neural networks (Section 4.4).

Average and median statistics

These two techniques are quite naive and simple. Main idea of the average method is that for every relevance value we calculate a characterizing vector as average among all feature-vectors with the same relevance. Then every query-document pair from the test set obtains relevance value equal to the relevance of closest characterizing vector from the learning set.

With Euclidean metric results of proposed method is quite low (DCG ~ 3.6), but after we changed it to cosine metrics, where closeness of two vectors measures as cosine value between them, results became better (DCG ~ 3.8).

Instead of taking average of feature-vectors, we can take median values for every feature and construct characterizing vector from them. With cosine metrics it gives even better results than previous two tries: DCG ~ 4.0, but even this is not acceptable as a good enough result.

K-nearest neighbour

K-nearest neighbour algorithm belongs to instance-based learning methods family [14]. The main idea behind such methods is simple storing of all samples from learning set and then, when new sample from test set is met, examining its relationship with stored ones and making classification of new sample. Since generalizing beyond test set is postponed, such algorithms are often called “lazy”. In contrast to methods that create full representation of target function, only its local approximation is constructed each time for a new test sample. A big advantage here is that even if target function is very complex, it still may be quite well represented by combination of less complex functions. However, there are two general disadvantages as well. First, time of processing can be large, because almost all computations are done after new sample is met. And second, which is important especially for k-nearest algorithm, is that usually all features are considered during process of retrieving ‘similar’ samples from the memory. If only few out of all features actually play role, then two very ‘similar‘ samples may be still very far away from each other. More details and theory about k-nearest neighbour algorithm can be found in [2, 14].

Basically there are three parameters which affect DCG and time of computation: number of nearest neighbours *k*, number of randomly chosen samples from training set *n* and number of features used. We have tried several combinations of these parameters (Table 1).

* Intel Core2 CPU 1.86 GHz, 3 GB of RAM

DCG	*k*	*n*	Features	Time* (hh:mm)
3.820	1	97290	5 most frequently met	03:24
3.937	5	6000	245	04:16
4.045	50	2000	245	01:20
4.129	200	80000	245	60:15

Table 1. K-nearest neighbour results.

As you can see from the first two rows in Table 1, it is better to use larger number of features and smaller number of samples than vice-versa. Increasing number of neighbors also improves the result. However the problem of finding optimal parameters for k-nearest neighbor using exhaustive search through all promising combinations seems not possible because of the large computational time. Under experiments we also decreased weight function and achieved slightly improved DCG (about 3%), but we got bigger computational time as was expected.

Therefore k-nearest method is not well appropriate for the document ranking problem because of many dimensions, large number of samples and hard tuning, but still it yields better result than median statistics.

Linear and quadratic regression

Within linear regression approach we assume that target function can be represented as linear combination of instance attributes. The problem here is to find optimal values for adjustable parameters of model, weights, by minimizing error function E. This is done by iteratively moving to the direction of steepest descend of E with some learning speed η. More theory about regressions can be found in [2, 14].

Linear regression is quite simple and fast algorithm. Only after 100 steps with learning speed η=0.001 it gives DCG = 4.187. This 2-minutes result is significantly better than 60-hours result of k-nearest. Adding momentum does not improve this result much.

We also tried extension of the linear regression algorithm, which use also quadratic term from Taylor series – a quadratic regression (polynomial with order 2). Number of weights in this case is much bigger – about 30000 versus 246 for linear regression, so computational time is much bigger (3 hours for 40 epochs), however obtained DCG is much better - 4.229. Further increasing of polynomial order seems impossible, because for cubic regression (order 3) number of weights will be about 2.5 millions.

For document ranking problem linear regression seems much more effective than k-nearest with big advantage as much smaller computational time. However best ranking function definitely is not so simple to be presented as linear combinations of features and therefore this method has strong limitations. Expanding to the quadratic regression improves result but algorithm is not fast anymore.

Feed-forward neural networks

Feed-forward neural networks (also known as Multilayer Perceptrons - MLP) are biologically inspired function approximation algorithms with successive applications [17] in numerous fields spanning from learning to recognize written character to driving a car.

The basic MLP model consists of series of functional transformations [2]. For a network with *H* sigmoid units in first hidden layer, *L* sigmoid units in second hidden layer and single linear output unit it can be expressed as

$$\hat{f}(x) = w_b + \sum_{l=1}^{L} w_l \sigma(w_{b,l} + \sum_{h=1}^{H} w_{l,h} \sigma(w_{b,h} + \sum_{j=1}^{n} w_{h,j} a_j(x))) \qquad (2)$$

Here w is set of network adjustable parameters (weights) and σ is sigmoid activation function:

$$\sigma(x) = \frac{1}{1+e^{-x}} \qquad (3)$$

MLPs are known as universal approximators – it has been proven [6] that three-layer neural network as described above is capable to approximate any function with arbitrary accuracy given sufficient number of units in hidden layer.

For training of such network one of possible choices is backpropagation algorithm [16]. It is based on gradient descent and tries to minimize sum squared error between network outputs and target values for these outputs. In contrast to linear regression, here error function can contain several local minima, and always there is a danger to fall into one of them instead of find optimal solution.

To overcome strong limitations of the linear regression algorithm we tried multiperceptron approach. We suggest that due to its universal approximation property it would be capable to reproduce complex behavior of ranking function better than any linear and quadratic regression.

At first we made a network architecture with one hidden sigmoid layer and single linear output unit (two-layer network). We tried various number of neurons in hidden layer *nh*. All networks were trained using backpropagation algorithm with learning speed η=0.01 during 30 epochs (corresponding running time varied from 5 minutes to 45 minutes depending on *nh*). Initial weights were initialized with small random values. As in case of linear regression, adding momentum did not give significant improvement. Final root-mean-square (RMS) error for training set was calculated for each case and presented in Figure 1.

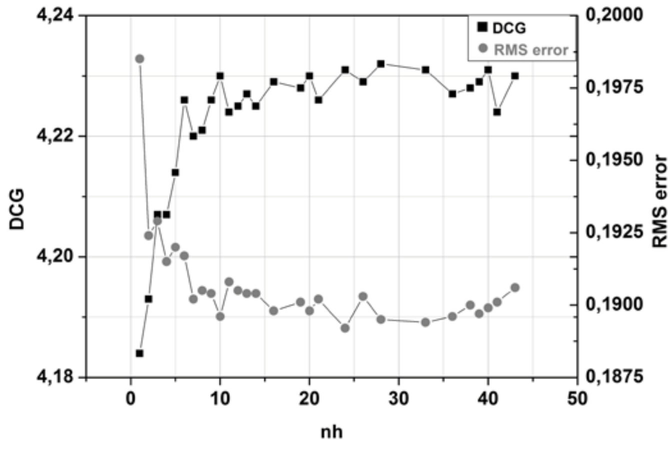

Figure 1. Dependence of DCG and final RMS error for training set on number of units in hidden layer for two-layer MLP

As you can see in Figure 1, DCG increases and RMS error decreases quite fast with small *nh*. But after reaching *nh*=10 both DCG and RMS error remain almost constants. The reason of their small oscillations is different local minima caused by different values of initial weights. We tried to make an average of different results from networks with different initial weights, but it did not improve DCG. Best DCG value, 4.232 (*nh*=28), is slightly better than in case of quadratic regression.

It is known that two-layer network is capable to approximate well only continuous functions [14], what is probably not true for document ranking function. Therefore we have implemented three-layer network with two hidden layers containing nh_1 and nh_2 sigmoid units. Again we used backpropagation algorithm with $\eta=0.01$. Number of epochs needed to achieve best DCG was determined empirically when DCG started to decrease. In Table 2 results are presented for different values of nh_1 and nh_2.

DCG	nh_1	nh_2	Epochs	Final RMS error	Time (hh:mm)
4.234	40	40	350	0.1762	06:51
4.235	50	30	200	0.1790	05:15
4.235	60	60	150	0.1819	04:35
4.235	200	50	200	0.1855	17:32
4.237	150	150	300	0.1825	27:33
4.239	10	10	1000	0.1792	04:34
4.239	30	30	200	0.1814	02:59
4.239	400	1	150	0.1839	26:58
4.241	100	3	300	0.1768	12:18
4.241	150	50	290	0.1819	20:38
4.245	40	20	450	0.1761	16:12
4.245	100	100	730	0.1742	42:20
4.247	70	70	200	0.1793	07:12
4.250	50	50	300	0.1797	07:31

Table 2. Three-layer MLP results.

As you can see, DCG is nonmonotonic function from nh_1 and nh_2. It is small for very small and very big nh_1 and nh_2 values and have maximum somewhere in between. The reason can be that small networks do not have enough adjustable parameters (weights) to capture all details of ranking function behavior while big networks due to big number of parameters simply 'learn' examples from learning set and therefore have poor generalization capability. The best DCG = 4.250 has been achieved in case of 50 units in both hidden layers after 300 epochs.

Then we decreased η in 100 times to $\eta = 10^{-4}$ and continued network learning. It gives even better DCG = 4.254 after 30 additional iterations. By decreasing η again in 100 times to $\eta = 10^{-6}$ we achieved DCG = 4.255, which is the best result over all our experiments.

Besides these experiments we made classification among 79 classes (corresponds to 79 different probabilities met in the learning set) with MLP and also pairwise probabilistic cost function as described in [3]. The former approach did not give any good results and latter gives only DCG = 4.239.

CONCLUSIONS AND FUTURE RESEARCH

In this paper we compared machine learning algorithms with application for particular document ranking problem. We used average and median statistics, k-nearest neighbor, linear and quadratic regression and feed-forward neural networks with different numbers of units and layers.

It was shown by experiments that three-layer neural network gives the best results over all techniques, but also require lots of tuning and few hours to compute. But even untuned neural network with random parameters can give fair results, slightly better than quadratic regression and much better than others approaches.

In the future research we plan to consider other machine learning techniques closer, such as for example SVM, regression trees and other types of neural networks. Also we are going to mix resulting functions from different methods to construct more complex and relevant ranking function which can provide better results.

REFERENCES

[1] Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair. A Comparison of Machine Learning Techniques for Phishing Detection. Proceedings of APWG eCrime Researchers Summit, 2007.

[2] Christopher M. Bishop (2006) Pattern recognition and Machine Learning, Springer

[3] Burges C.J. C., Shaked T., Renshaw E., Lazier A., Deeds M., Hamilton N., Hullender G. (2005). Learning to rank using gradient descent. . ICML '05, vol. **119**. p, 89-97. ACM.

[4] Crammer K., Singer, Y. (2002). Pranking with ranking. In Advances in neural information processing A Neural Network Approach to Ordinal Regression systems (nips) **14**, 641-647. Cambridge, MA: MIT press.

[5] Nello Cristianini and John Shawe-Taylor. An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press, 2000.

[6] Cyhenko, G. (1988). Continuous valued neural networks with two hidden layers are sufficient (Technical Report). Department of Computer Science, Tufts University, Medford, MA.

[7] Freund Y., Iyer R., Schapire R., Singer Y. (2003). An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research, **4**, 933-969.

[8] Anna Goldenberg, Jeremy Kubica, Paul Komarek, Andrew Moore, Jeff Schneider. A Comparison of Statistical and Machine Learning Algorithms on the Task of Link Completion. KDD Workshop on Link Analysis for Detecting Complex Behavior, 2003.

[9] Herbrich R., Graepel T., Obermayer K. (2000). Large margin rank boundaries for ordinal regression. Advances in large margin classifiers, 115-132. Cambridge, MA: MIT Press.

[10]Jiawei Han and Micheline Kamber. Data Mining. 2005.

[11]Kramer S., Widmer G., Pfahringer, B., DeGroeve M. (2001). Prediction of ordinal classes using regression trees. Fundamenta Informaticae, **47**, 1-13.

[12]Chih-Chin Lai; Ming-Chi Tsai. An empirical performance comparison of machine learning methods for spam e-mail categorization. Hybrid Intelligent Systems. Volume , Issue , 5-8 Dec. 2004, p. 44 – 48

[13]David Meyer, Friedrich Leisch, and Kurt Hornik. The support vector machine under test. Neurocomputing **55(1-2)**: 169-186, 2003

[14]Tom M. Mitchell (1997), Machine Learning, McGraw Hill

[15]Shyamsundar Rajaram, Ashutosh Garg, Xiang Sean Zhou and Thomas S. Huang (2003). In Machine learning: Ecml 2003, vol. **2837**, 301-312, Springer-Verlag.

[16]Rumelhart D. E., McClelland J. L. (1986). Parallel distributed processing: exploration in the microstructure of cognition (Vols. **1,2**). Cambridge, MA: MIT Press.

[17]Rumelhart D., Widrow B., Lehr M. (1994). The basic ideas in neural networks. Communications of the ACM, **37(3)**, 87-92.

[18]Soumen Chakrabarti. Mining the Web, 2007.

[19]Nigel Williams, Sebastian Zander, Grenville Armitage. ACM SIGCOMM Computer Communication Review. Volume **36**, Number 5, October 2006.