Noise Correction in Pairwise Document Preferences for Learning to Rank

Harsh Trivedi(✉) and Prasenjit Majumder

Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar, India
harshjtrivedi94@gmail.com, prasenjit.majumder@gmail.com

Abstract. This paper proposes a way of correcting noise in the training data for Learning to Rank. It is natural to assume that some level of noise might seep in during the process of producing query-document relevance labels by human evaluators. These relevance labels, which act as gold standard training data for Learning to Rank can adversely affect the efficiency of learning algorithm if they contain errors. Hence, an automated way of reducing noise can be of great advantage. The focus in this paper is on noise correction for pairwise document preferences which are used for pairwise Learning to Rank algorithms. The approach relies on representing pairwise document preferences in an intermediate feature space on which ensemble learning based approach is applied to identify and correct the errors. Up to 90 % errors in the pairwise preferences could be corrected at statistically significant levels by using this approach, which is robust enough to even operate at high levels of noise.

1 Introduction

Learning to rank is an approach to automatically build a ranking model, based on the training data using machine learning technologies [6]. The training data for learning to rank when used for document retrieval usually consists of queries, the associated documents, and relevance labels for query-document pairs which are assigned by human judges. Several previous works have shown that human judges may not agree with each others in the task of assigning relevance labels to query-document pairs [1,9,10]. Since human annotation is costly, especially in web-search which requires large amount of training data, one can usually not afford to have several annotators to make multiple judgments. As a result, such relevance judgements are prone to be biased, unreliable and noisy. Xu et al. have shown that errors in training data can significantly degrade the performance of ranking functions trained by learning to rank algorithms [11]. So, automatic error correction for training data of learning to rank can be of great advantage.

Primarily, there are 3 types of learning to rank algorithms: pointwise, pairwise and listwise [5]. In this paper, the focus is on training data of pairwise Learning to Rank algorithms which take pairwise preferences of documents for each query as the learning instances. Using the proposed method, noise present in the pairwise preferences can be considerably reduced. To test it's efficiency

S. Ma et al. (Eds.): AIRS 2016, LNCS 9994, pp. 295–301, 2016.
DOI: 10.1007/978-3-319-48051-0_22

different levels of artificial noise are injected in the data. On this noisy data, noise reduction process is applied and the output is compared to the original human generated data, which is assumed to be correct for the sake of the evaluation. Since the effectiveness is tested on a wide range of injected noise, it also checks the robustness of the proposed process to initial noise present in the data.

There have been few attempts on improving the quality of training data for Learning to Rank. Geng et al. proposed a way of computing training data quality for Learning to Rank with a concept of "Pairwise Preference Consistency" (PPC). They have shown a way to select the most optimal subset of the initial training data which maximizes the PPC score [3]. However, because of selection of a subset there is a possibility of loosing some important examples which are discarded in this process. Hence, in this attempt an error correction, rather than error elimination approach is targeted. Xu et al. proposed a method of error correction, by leveraging the information from click-through data [11]. However, it is not natural to assume the availability of such data in all cases. To the best of our knowledge, there hasn't been any work yet, that deals with improving the quality of training data for learning to rank by error correction rather than error elimination solely on the based on training data itself.

In contrast to ranking, there has been good amount of work on improving the quality of training data for classification [2]. Ensemble learners are often used for this purpose in classification data. For example, many classifiers are learnt from different samples of training data and used to classify the data. If there is a good amount of agreement among the classifiers then only that instance is kept, otherwise discarded. A similar approach is used here to correct the highly probable error-some instances and there by reducing noise in data. However, instead of elimination, correction of the highly suspicious preference pairings is performed. Hence unlike noise elimination, there is no risk of loosing important training instances in process of noise correction.

The remaining paper is organized as follows: Sect. 2 describes the proposed approach, Sect. 3 elaborates on the experimental setup, Sect. 4 discusses the results and Sect. 5 concludes and discusses future scope of this project.

2 Approach

Learning to Rank training data contains queries, the associated documents, set of features extracted from each query-document pair and the relevance label of documents for the corresponding query. Formally, given query q, there is a set of documents $D = \{d_1, d_2..d_n\}$ and for each query-document pair (q, d_i) there exists a feature vector $\vec{f}(q, d_i)$ and relevance label $rel(q, d_i)$. First of all, transformation of this representation to pairwise preference sets is performed as following:

2.1 Pairwise Preferences Sets

We define a **partial pairwise preference set** as:

$$\{[\bar{F}(q : d_i > d_j), 1] : rel(q, d_i) > rel(q, d_j) \text{ and } d_i, d_j \in D\} \tag{1}$$

and **full pairwise preference set** as:

$$\{[\bar{F}(q : d_i > d_j), 1] \cup [\bar{F}(q : d_j > d_i), 0] : rel(q, d_i) > rel(q, d_j) \text{ and } d_i, d_j \in D\} \tag{2}$$

where,
$\bar{F}(q : d_i > d_j)$ is document preference pair vector representation, which is taken as:

$$[\bar{F}(q : d_i > d_j)] = [\bar{f}(q, d_i) - \bar{f}(q, d_j)] \tag{3}$$

The preference pair $(q : d_i > d_j)$ is represented with feature vector $[\bar{f}(q, d_i) - \bar{f}(q, d_j)]$. Its class label is 1 if $rel(q, d_i) > rel(q, d_j)$ and 0 if otherwise. This means that for a given query q, if A is set of relevant documents and B is set of irrelevant documents, then there is a set $\{(q : a > b)|a \in A, b \in B\}$ for which class label is 1. Also, at the same time, there is a set $\{(q : b > a)|a \in A, b \in B\}$ for which the class label is 0. Hence, in all there are $2 \times |A| \times |B|$ number of pairwise instances, half of which are tagged positive and other half negative. Partial and Full pairwise preference set are easily inter-convertible from each others.

2.2 Noise Injection

Once partial pairwise preference set of original noiseless data is performed, different levels of noise are injected in it. For noise level p, each pairwise document preference is reversed (≡ class label is flipped) with probability p and kept the same with probability $p - 1$. The partial preference set is then converted to full preference representation.

2.3 Two Phase Process

For each query, a 2-phase process on the full pairwise preference set is performed.

Phase 1. x-fold cross validation on the full pairwise preference set with classifier a is performed. For each of the x parts of the data, classifier a is trained on remaining $x - 1$ parts and used to label the remaining part. The preference pair is identified as faulty (error) if the predicted label doesn't match the actual label. This process is repeated for $x \in \{3, 5, 7, 10\}$ and $a \in \{\text{MultilayeredPerceptron}\}$[1]. Intersection of all preference pairs are made which are identified as faulty by

[1] Weka - machine learning software was used for classification [4] .

any combination of x and a. It is worth noting that taking such intersection highly improves the precision of fault identification. Once, these suspected faulty preference pairs are extracted, they are removed from the full pairwise preference set. A separate set is made from them, basically decomposing the initial data in 2 parts: purer and noisier sub-sample.

The choices of x and a were empirically found to be working efficiently. We do not claim that this is the best choice, but it is at least a good choice for performing this task. Also, Multilayer Perceptron classifier with default parameters was found to be giving far better results for this task than any other classifier available in weka software.

Phase 2. The purer sub-sample of full preference set is used to train the classifier b. The trained model is then used for detecting the faulty preferences in noisier sub-sample. Here $b \in$ {Multilayered Perceptron, Random Forest }. Finally a union of these faults (errors) predicted by each classifier b is taken and they are considered as the final pairwise preference faults which need to be flipped.

2.4 Noise Measurement

Once the appropriate flips are made, measurement of the noise of updated paired representation is done. It is computed as the number of incorrect document preference pairs to the total number of preference pairs. The idea of computing Document Pair noise is taken from [7] in which it is referred to as pNoise. From the study, they have concluded that document pair noise captures the true noise of ranking algorithms, and can well explain the performance degradation of ranking algorithms. Hence, it has been used to evaluate the effectiveness of the noise-correction process by the reduction in document pair noise achieved by the method[2].

3 Experimental Setup

Experiments are performed on 3 standard Learning to Rank LETOR 3.0 datasets [8]: OHSUMED, TREC-TD-2003, TREC-TD-2004. Noise levels of {0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5} are injected in each of these datasets and checked to what extent noise can be corrected depending on the initial noise present. The process is performed thrice and the average results are reported.

OHSUMED contains 106 queries with approximately 150 documents associated with each query. TREC-TD-2003 contains 50 queries with approximately 1000 documents associated with each query and TREC-TD2004 contains 75 queries with approximately 1000 documents for each query. OHSUMED represents query-documents pair by a set of 45 features, while TREC-TD 2003, 2004 use 44 features each. OHSUMED has 3 relevance levels $\{2, 1, 0\}$ while TD2003 and TD2004 have $\{1, 0\}$.

[2] document pair noise will be referred to as noise henceforth.

Table 1. Noise correction on OHSUMED

Injected noise	Post correction noise	Percentage noise reduction	Queries improved	Queries worsened
0.05	0.029	**42.00 %***	100	6
0.1	0.050	**50.00 %***	96	10
0.15	0.085	**43.33 %***	105	1
0.2	0.091	**54.50 %***	105	1
0.25	0.132	**47.20 %***	103	3
0.3	0.147	**51.00 %***	105	1
0.35	0.204	**41.71 %***	103	3
0.4	0.269	**32.75 %***	100	6
0.45	0.419	**6.80 %***	90	16
0.5	0.493	**1.40 %**	51	55

Table 2. Noise correction on TREC-TD-2003

Injected noise	Post correction noise	Percentage noise reduction	Queries improved	Queries worsened
0.05	0.002	**96.00 %***	50	0
0.1	0.006	**94.00 %***	50	0
0.15	0.019	**87.33 %***	50	0
0.2	0.013	**93.49 %***	50	0
0.25	0.023	**90.80 %***	50	0
0.3	0.030	**90.00 %***	50	0
0.35	0.064	**81.71 %***	50	0
0.4	0.108	**73.00 %***	50	0
0.45	0.393	**12.66 %***	39	11
0.5	0.483	**3.40 %**	23	27

Table 3. Noise correction on TREC-TD-2004

Injected noise	Post correction noise	Percentage noise reduction	Queries improved	Queries worsened
0.05	0.002	**96.00 %***	75	0
0.1	0.004	**96.00 %***	75	0
0.15	0.009	**94.00 %***	75	0
0.2	0.011	**94.50 %***	75	0
0.25	0.027	**89.20 %***	75	0
0.3	0.032	**89.33 %***	75	0
0.35	0.093	**73.42 %***	75	0
0.4	0.109	**72.75 %***	75	0
0.45	0.392	**12.88 %***	64	11
0.5	0.510	−2.00 %	37	38

4 Results

Tables 1, 2 and 3 show computed noise before and after applying the noise-correction process across different levels of injected noise. They also show the number of queries for which the noise decreased and the number of queries for which the noise increased after the process. To check if this reduction in noise was statistically significant, t-tests were performed using noise levels before and after the process across all the queries. Improvements marked by (*) symbol denote statistical significance with p-value < 0.05.

The 2-phase process reduces significant amount of noise up to noise level of 0.4. After this, curve takes a very steep turn and almost fails to reduce noise at statistically significant levels around noise level of 0.5. However, the process has been proved robust enough to correct errors even at high noise level of 0.45 in each of the 3 datasets.

The difference in noise reduction between OHSUMED and TREC-TD datasets is due to an inherent characteristic of the datasets. OHSUMED has 3 relevance labels $\{2, 1, 0\}$ and so it's preference set contains 3 kinds of document pairs: $(d_2, d_1), (d_1, d_0)$ & (d_2, d_0) from which anomalies are to be found. Whereas, TREC-TD datasets contain only 2 relevance labels $\{1, 0\}$ and so have only 1 kind of document pair (d_1, d_0). So the noise reduction is efficient in case of TREC-TD compared to OHSUMED in which there are mixed document pairs because of which error detection is difficult.

5 Conclusion and Future Scopes

This paper proposes a simple yet very efficient approach to correct the errors in pairwise preferences for learning to rank. The proposed approach was able to reduce up to 90 % of induced noise at statistically significant levels depending on the initial noise injected in it. The robustness of this process has also been checked by inducing different noise levels. On response to this, the process was able to correct errors at statistically significantly even at high noise level of 0.45. The proposed model has been checked on three different Learning to Rank data-sets and shown to work efficiently on each of them.

In reality some documents are difficult to assign relevance than others. All mistakes are not equally probable. So, a more realistic method for noise injection which considers this can help to better evaluate this approach. Apart of that, reduction in noise of pairwise document preferences should have direct positive impact on efficiency of pairwise learning to rank algorithms. Different Learning to Rank algorithms have different levels of robustness against noise [7]. Hence, as a future work, it would also be interesting to analyse the effect of noise correction of training data on efficiency of various pairwise learning to rank algorithms.

References

1. Bailey, P., et al.: Relevance assessment: are judges exchangeable and does it matter. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 667–674. ACM (2008)
2. Brodley, C.E., Friedl, M.A.: Identifying mislabeled training data. J. Artif. Intell. Res. **11**, 131–167 (1999)
3. Geng, X., et al.: Selecting optimal training data for learning to rank. Inf. Process. Manag. **47**(5), 730–741 (2011)
4. Hall, M., et al.: The WEKA data mining software: an update. ACM SIGKDD Explor. Newslett. **11**(1), 10–18 (2009)
5. Hang, L.I.: A short introduction to learning to rank. IEICE Trans. Inf. Syst. **94**(10), 1854–1862 (2011)

6. Liu, T.-Y.: Learning to rank for information retrieval. Found. Trends Inf. Retrieval **3**(3), 225–331 (2009)
7. Niu, S., et al.: Which noise affects algorithm robustness for learning to rank. Inf. Retrieval J. **18**(3), 215–245 (2015)
8. Qin, T., et al.: LETOR: a benchmark collection for research on learning to rank for information retrieval. Inf. Retrieval **13**(4), 346–374 (2010)
9. Voorhees, E.M.: Variations in relevance judgments, the measurement of retrieval effectiveness. Inf. Process. Manag. **36**(5), 697–716 (2000)
10. Voorhees, E., Harman, D.: Overview of the fifth text retrieval conference (TREC-5). In: NIST Special Publication SP, pp. 1–28 (1997)
11. Jingfang, X., et al.: Improving quality of training data for learning to rank using click-through data. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, pp. 171–180. ACM (2010)