Learning to Rank from Pseudo Labeled Examples

Harsh Trivedi
DA-IICT, Gandhinagar
harshjtrivedi94@gmail.com
Supervisor
Prof. Prasenjit Majumder

***Abstract*— This paper reports the description, results and analysis of the experiments on Learning to Rank (LTR) undertaken as part of my final year project. The baseline setup of LTR is explained and compared with the best state-of-the art methods of standard retrieval. With this, a motivation of using pseudo labels for LTR is explained. Subsequently, each of our novel experiments on how far we can go with LTR without the human judged query relevance judgements have been elaborated. This attempts to answer that whether we can eliminate or at least reduce the human intervention in the process of learning to rank. We conclude that the answer is affirmative. In the end, few experiments on feature selection for LTR have also been carried out that can help improvise LTR for each of the previous experiments.**

***Index Terms*— Information Retrieval, Learning to Rank, Feature Selection, Psuedo Labeling**

I. INTRODUCTION

Learning to Rank (LTR) can be applied to any ranking problem - a typical use-case is document retrieval. Given the set of Documents and a query, the task of returning the top k most relevant documents is called Document Retrieval. The standard retrieval model does this task by computing score $f(q,d)$ for a query q and a document d, sorting the documents by score and returning the top k documents. Whereas LTR does this job by modeling this to a learning problem. For each query different models give different ranking of documents. An LTR algorithm tries to learn from these ranking models considering them as a feature set and based on this, it generates a final ranking of documents for a given query. The following images demonstrate the difference between standard retrieval models and LTR for document retrieval [1].

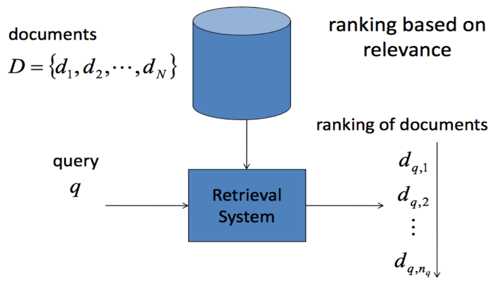

Fig. 1: Document Retrieval

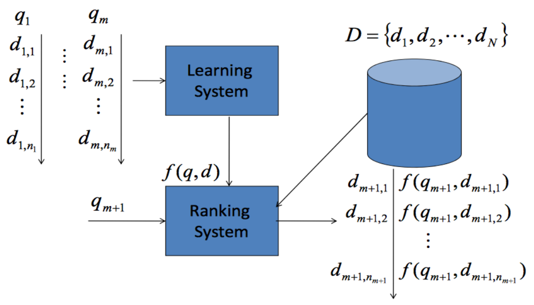
q_1
$d_{1,1}$
$d_{1,2}$
d_{1,n_1}
q_m
$d_{m,1}$
$d_{m,2}$
d_{m,n_m}
Learning System
$D=\{d_1,d_2,\cdots,d_N\}$
$f(q,d)$
q_{m+1}
Ranking System
$d_{m+1,1}$ $f(q_{m+1},d_{m+1,1})$
$d_{m+1,2}$ $f(q_{m+1},d_{m+1,2})$
$d_{m+1,n_{m+1}}$ $f(q_{m+1},d_{m+1,n_{m+1}})$

Fig. 2: Learning to Rank for Document Retrieval

There are 3 main types of LTR methods: (1) Point-wise (2) Pair-wise (3) List-wise [1]. Point wise methods assign to each query-document pair, a score $f(q,d)$ based on the learnt model and then sort the results. Pair-wise method learns for each pair of documents, the order of documents and the corresponding features. Based on these pair-wise computations it generated the final ranking list. The list-wise approach takes the whole rank list generated by individual retrieval models and translates it to the final rank list. In this case study, we have limited my scope of research to a well-known pair-wise LTR method called SVM-Rank [2].

LTR bases it's learning from the gold standard dataset which are human judged query relevance judgements. Certain documents are pulled and for each of them, corresponding to the query, a relevance class is given ie. 2, 1, 0 etc by a human evaluator. However, this process is extremely tedious, time consuming and costly. If we can eliminate or reduce the human intervention learning process of LTR we can benefit and speed up the process a lot. Hence comes the idea of using pseudo labels for learning. Here we are trying to answer the research question that can we automatically come up with a suitable training data for LTR and yet deliver the comparable or even better results than the LTR that is trained on the human judged relevance judgements. The final objective is to boost the performance of LTR and make it even better than best retrieval model like BM25 by using pseudo learning.

II. DATA-SET

A. Corpus

In recent years, lot of research has gone into proposing more and more efficient methods of Learning to Rank. Specifically, the launch of benchmark dataset like LETOR 3.0, LETOR 4.0, Yandex, Yahoo LTR Challenge etc which were prepared particularly for LTR research have provided a standard experimental setting across which different LTR algorithms can be tested. Abstracting out the feature extraction and representing dataset only as query-document level feature vector keeps only the LTR algorithm a variable in experimental setting. This has definitely benefited the community to directly compare different algorithms, however it has also led to exclusive usage of these datasets of LTR purpose. For this reason, the effect of LTR algorithms on datsets like TREC CDS, WSJ, ROBUST etc. have not been explored much. At the same time, due to usage of query-document level features without any knowledge of what query or document text was (as the released dataset provided) has prevented the exploration of how other techniques of boosting retrieval effectiveness like query-processing (eg. Query Expansion) and Document pre-processing would behave with LTR. Hence, in this research we perform our experiments on TREC Clinical Decision support track (CDS-2014), and keeping the LTR algorithm same (SVM-Rank) explore a novel method of boosting LTR performance.

The CDS track investigates the techniques for linking medical records to information relevant for patient care. It contains 728 thousand full length medical research papers as documents and 30 queries each of which has 2 versions of it: summary and description. The queries describe the medical situation of a patient in mostly layman language corresponding to which the system is expected to retrieve most relevant research papers. We have used summary part of the queries for retrieval.

Due to the large size of dataset, performing experiments had become increasing difficult since LTR is inherently slower than standard retrieval models. Hence a smaller pool of 108 thousand documents was sampled from full dataset and all experiments were carried out on it. This pool was the set of unique documents present in top 5000 documents in retrieved list of state-of-the-art models: BM25, Lemur-TFIDF, Hiemstra-LM.

B. Feature Extraction

We started with the standard feature set as explained in Letor-Ohsumed dataset, which have been identified as important for LTR [3]. These constitute 15 features which were extracted using Terrier for each query-document pair. The rest of the 25 features have been taken as the already implemented retrieval model in Terrier framework [4], [5], [6], [7].

III. Base-line Results

After verifying the initial implementation of feature extraction, the following results were obtained. For LTR (SVM-Rank), 5-fold cross validation was used to compute final result. The standard three models shown here are the 3 best performing models from all 35 features extracted for LTR.

	BM25	**Lemur-TFIDF**	**Hiemstra-LM**	**Baseline LTR**
InfNDCG	0.1955	0.2043	0.1781	0.18348

TABLE I: BaseLine results

IV. Generating Pseudo Query Relevance Judgements

The first attempt was made by splitting the top K documents from retrieved list of BM25 in 3 equal parts, labeling first as of relevance class 2, second as relevance class 1, and last as relevance class 0. To check the effect of results on K, its was taken from 1000, 2000, 3000.

	Baseline - LTR	**Pseudo LTR**
K = 1000	0.18348	0.17868
K = 2000	0.18348	0.18344
K = 3000	0.18348	0.18048

TABLE II: BaseLine results

The first naive attempt itself lead to interesting results. It can be seen that results from pseudo learning are very comparable to the results of human judged learning. Interestingly there is no statistical difference between them ($p >> 0.05$). Improvising upon this attempt based on some novel ideas, lead yet better results.

The following 3 directions of improvisation were experimented:

1) **Effective Partitioning:** Division of top K list in 3 parts in more statistical way convincing way.
2) **Training QRel Purification:** Removing the noisy training data from this partitioning.
3) **QRel Augmentation:** Augmenting the pseudo qrels (Query Relevance Judgements) with human-judged qrels.

A. Effective Partitioning

1) Clustering: Considering some value of K (say, K = 1000), a typical plot of the scores of top K documents given by some retrieval model (say, BM25) versus the rank of of document is as shown in Fig 3.

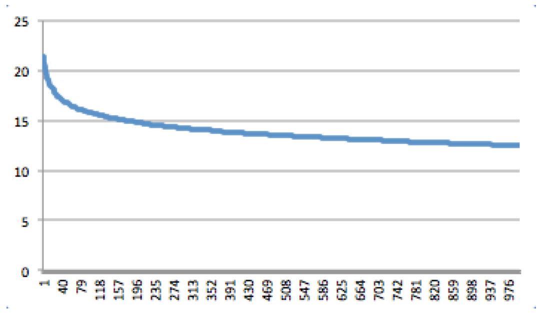

Fig. 3: Score of document by BM25 vs ranking

This lead to an intuition that rather than uniform equal partition, partitions should be made according to the clustering defined by the document scores. Three different clustering methods were tried, as implemented in Weka toolkit [ref]: Simple KMeans with euclidean distance, Simple KMeans with manhattan distance and EM algorithm. EM algorithm turned out to give best results, which are documented in the results section.

2) Classification: Other way of partitioning was devised by leveraging the knowledge of human judged qrels. Using the human judged qrels and corresponding feature vectors from LTR, a classification model by trained. Using this model, the tok K documents from the ranking were classified in relevance score of 2, 1, 0 and then were used for training LTR system. This approach is not strictly pseudo learning in nature, but it can help in qrel augmentation as it will be discussed later in this paper. We tried using almost all classifiers provided in Weka [8] and choose the one which gave best 10-fold AUC for human judged qrels. Random Forest and Logistic turned to be very good classifiers in this context. We used Random Forest for the reported experiment.

3) Partition with most cross-relevance consensus: Given a base ranklist of top K documents, there can be $K2$ partitions in 3 pseudo relevance classes. If we pose an optimization problem of choosing such a parition which has maximum number of pairs among cross pseudo relevance classes for which the order is complete consensus among all the rank lists. Although, a brute force method seems quite a costly approach [$O(n^5)$], it can be easily reduced to time complexity of [$O(n^3)$]. However, this has not been implemented yet and so we do not elaborate on this approach in a greater detail.

The effects on partitioning by clustering and classification compared to uniform partitioning is shown in table III

	Uniform Partitioning	**Partitioning by clustering**	**Partitioning by classification**
K = 1000	0.17868	0.2010 *	**0.2011** *
K = 2000	0.18344	0.18134 *	**0.2025** *
K = 3000	0.18048	0.18212 *	**0.1970** *

TABLE III: Effects of Partitioning of pseudo LTR (Inf-NDCG results)
[(*) denotes the improvement over baseline LTR (human judged query-relevance judgements)]

The results clearly show an improvement. This means that in absence of any human judged labeling, we are able to yield same or better results on the dataset. However, the improvement was found to be not statistically significant ($p > 0.05$) which led to further attempts to improvise the approach.

B. Training QRel Purification

As mentioned earlier we have limited the scope of this research project on SVM-Rank, a pairwise Learning to Rank approach. Hence for each query-document pair (q, d_a) in relevance class A, is compared to each query-document pair (q, d_b) in relevance class B for same query q. If $A > B$, the SVM-Rank will learn $f_v(q, d_a) > f_v(q, d_b)$, where f_v is the feature vector function defined for each query document pair. Hence, if there is pair in our pseudo relevance classes for which pseudo relevance $A > B$, but their actual relevance is the reverse, then it induces a wrong training example to our pair-wise SVM-Rank learner. However, there is not real method of knowing what actual relevance order is unless we have human judgement. Hence, we are proposing voting based method to remove such pairs whose order of relevance is susceptible.

Given 3 rank list of top K documents given by 3 different retrieval model $r_1(d_{r_1}1, d_{r_1}2, d_{r_1}3...d_{r_1}k)$, $r_2(d_{r_2}1, d_{r_2}2, d_{r_2}3...d_{r_2}k)$ and $r_3(d_{r_3}1, d_{r_3}2, d_{r_3}3...d_{r_3}k)$, we use any one rank list (say, r_1) as the base ranking model. We partition this base ranking model in 3 pseudo relevance classes according to any partitioning method as explained above. If there are n_1, n_2 and n_3 documents in these 3 psuedo relevance classes, then there will be $n_1 \times n_2 \times n_3$ pairs for which SVM-Rank will generate training example to learn. Our target is to eliminate documents from the partitioning of r_1 such that maximum number of pairs of $n_1 \times n_2 \times n_3$ are in actual ordering of relevance. Since, we do not have knowledge of actual ordering, we use other 2 rank list for voting the appropriateness of the ordering. For each pair (d_a, d_b) in $n_1 \times n_2 \times n_3$ pairs, we check if the ordering of (d_a, d_b) is same in other 2 rank lists r_2 and r_3. If not, then we need to remove either one from the pair $\{d_a, d_b\}$ to remove this conflicting order between the rank lists. This process is carried out for all such combinations and final set of pairs is prepared for each of which at least one document needs to be eliminated to remove conflict. We choose such documents from this set of pairs that minimizes the documents needed to be removed to bring final partition in complete cross pseudo relevance consensus.

Finally we are left with partitioning of r_1, for which ordering between any document of one psuedo relevance class A and any document of another psuedo relevance class B is same for each of the three rank lists r_1, r_2 and r_3. We define process of eliminating documents from relevance classes as purification of training data. To check the effectiveness of this process, we prepare 3 versions of 3-part paritioning of base ranking r_1:

1) **All** : Partitioning with no purification
2) **With Consensus**: Purification performed after partitioning
3) **Without Consensus**: Keeping only the documents eliminated during purification

The resulting effects of purification on each type of partitioning is shown in tables IV, V, VI

	ALL	**With Consensus**	**Without Consensus**
K = 1000	0.17868	**0.18518** *	0.1311
K = 2000	0.18344	**0.18608** *	0.12816
K = 3000	0.18048	**0.18748** *	0.12598

TABLE IV: Effects of Purification on Uniform Partitioning (Inf-NDCG results)

	ALL	**With Consensus**	**Without Consensus**
K = 1000	**0.2011** *	0.16782	0.16914
K = 2000	0.18134	**0.1815**	0.1702
K = 3000	0.18212	**0.18226**	0.16936

TABLE V: Effects of Purification on Partitioning by Clustering (Inf-NDCG results)

	ALL	**With Consensus**	**Without Consensus**
K = 1000	**0.2011** *	0.1580	0.1929 *
K = 2000	**0.2025** *	0.1568	0.2033 *
K = 3000	**0.1970** *	0.1879 *	0.19632 *

TABLE VI: Effects of Purification on Partitioning by Classification (Inf-NDCG results)

The proposed method of purification doesn't always necessarily improve the overall performance. For example, with uniform partitioning the *'with consensus'* consistently worked better than *'all'* and *'without consensus'* ones. However, the same doesn't apply to clustered and classified partitionings. The purification process is proposed to eliminate the suspected noise from the initial training data, however we believe this elimination tends to render too strictly on classified and clustered partitionings. This leads to loss of important training examples and hence leads to worsened result.

C. Pseudo Qrel Augmentation

This process attempts to check whether the generated training relevance judgements can be used alongside with the human judged relevance judgements. In each of the above experiments, we add the human judged relevance judgements and see the effect. In this process, if there is a query-document pair for which the relevance class is conflicting between human judgement and pseudo judgement, then human judgement is given preference. This approach doesn't actually work in complete absence of human labels, but it can be used to systematically augment the human labels and achieve better results.

The resulting effects of purification on each type of partitioning is shown in tables VII, VIII, IX

	ALL	**With Consensus**	**Without Consensus**
K = 1000	0.16182	**0.18674**	0.12344
K = 2000	0.14786	**0.16412**	0.08253
K = 3000	**0.14744**	0.146959	0.05373

TABLE VII: Effects of Augmentation on Uniform Partitioning (Inf-NDCG results)

	ALL	**With Consensus**	**Without Consensus**
K = 1000	0.18114	**0.22549** *	0.1639
K = 2000	0.17614	**0.18974** *	0.1497
K = 3000	0.17024	**0.19066** *	0.1366

TABLE VIII: Effects of Augmentation on Partitioning by Clustering (Inf-NDCG results)

	ALL	**With Consensus**	**Without Consensus**
K = 1000	0.1578	0.1692	**0.1736**
K = 2000	0.1562	0.1651	**0.1821**
K = 3000	0.1655	0.1674	**0.1882**

TABLE IX: Effects of Augmentation on Partitioning by Classification (Inf-NDCG results)

The results are much counter intutive here aswell. The augmentation of pseudo labels with the human labels doesn't necessarily improve the results. Infact most of the times pseudo labels and human labels individually alone give better results than the combined version. We yet need to investigate the reason for the same. However, we believe that with proper augmentation schemes improvement can be made.

V. FEATURE SELECTION

It can be clearly seen that the baseline LTR system compared to best retrieval model BM25 works marginally poorly. Hence, even after the proposed methods to boost the performance of LTR, the performance is not significantly better than BM25. Since whole LTR depends on the features extracted, first direction of scrutiny had to be features. We implemented 3 methods of feature selection devised particularly for LTR, namely Maximal Marginal Relevance (MMR), Maximum Sum Dispersion (MSD), Modern Portfolio Theory (MPT) [9]. Each of these methods try to select a feature set by maximizing two quantities: the importance of each individual feature and the diversity among the selected feature set. Out of 35 features, number of selected features were varied from 10, 20, 30 and checked for improvement on baseline.

The resulting effects of feature selection of baseline LTR is provided in table X

	K = 10	**K = 20**	**K = 30**	**K = 35 (all)**
MMR	**0.19852**	0.19448	0.19474	0.18348
MPT	0.19262	**0.20618**	0.18284	0.18348
MSD	0.19184	**0.19796**	0.1915	0.18348

TABLE X: Feature Selection on BaseLine LTR (Inf-NDCG results)

VI. CONCLUSION AND CURRENT WORK

In this paper, we explore a novel way of improving the performance of LTR and at the same time explore how LTR can yeild results comparable or even better in the absence of human judged relevance judgements which is very counter intuitive finding. We also propose statistical methods of tuning the pseudo learning data to yield improvement. We have yet used very intutive methods of partitioning and augmentation. However, we believe a more thorough research in this direction can definitely yeild good improvements.

Also as the results show, feature selection also helps improve the performance of LTR. Hence if we perform feature selection before the pseudo qrel experiments then an improvement can be expected. However, this work has not been completed yet.

Lastly, we have performed the current experiments on a pooled dataset of CDS-14. However, to make a claim we need to test these approaches on other standard datasets also.

References

[1] LI Hang. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, 94(10):1854–1862, 2011.

[2] Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM.

[3] T Qin, TY Liu, J Xu, and H Li. Letor: A benchmark collection for learning to rank for information retrieval. In *SIGIR 2007 Workshop on Learning to rank for information retrieval*, 2008.

[4] Craig Macdonald, Richard McCreadie, Rodrygo LT Santos, and Iadh Ounis. From puppy to maturity: Experiences in developing terrier. *Proc. of OSIR at SIGIR*, pages 60–63, 2012.

[5] C.;Macdonald C.;Plachouras V. Ounis, I.;Lioma. Research directions in terrier. *Novatica/UPGRADE Special Issue on Web Information Access, Ricardo Baeza-Yates et al. (Eds), Invited Paper*, 2007.

[6] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma. Terrier: A High Performance and Scalable Information Retrieval Platform. In *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)*, 2006.

[7] I. Ounis, G. Amati, Plachouras V., B. He, C. Macdonald, and Johnson. Terrier Information Retrieval Platform. In *Proceedings of the 27th European Conference on IR Research (ECIR 2005)*, volume 3408 of *Lecture Notes in Computer Science*, pages 517–519. Springer, 2005.

[8] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[9] Kaweh Djafari Naini and Ismail Sengor Altingovde. Exploiting result diversification methods for feature selection in learning to rank. In *Advances in Information Retrieval*, pages 455–461. Springer, 2014.