Learning to Blend Rankings: a Monotonic Transformation to Blend Rankings from Heterogeneous Domains

Zhenzhen Kou
Yahoo! Labs
Sunnyvale, CA
zzkou@yahoo-inc.com

Yi Chang, Zhaohui Zheng
Yahoo! Labs
Sunnyvale, CA
{yichang,zhaohui}@yahoo-inc.com

Hongyuan Zha
College of Computing
Georgia Institute of Technology
Atlanta, GA
zha@cc.gatech.edu

ABSTRACT
There have been great needs to develop effective methods for combining multiple rankings from heterogeneous domains into one single rank list arising from many recent web search applications, such as integrating web search results from multiple engines, facets, or verticals. We define this problem as *Learning to blend rankings* from multiple domains. We propose a class of learning-to-blend methods that learn a monotonically increasing transformation for each ranking so that the rank order in each domain is preserved and the transformed values are comparable across multiple rankings. The transformation learning can be tackled by solving a quadratic programming problem. The novel machine learning method for blending multiple ranking lists is evaluated with queries sampled from a commercial search engine and a promising improvement of Discounted Cumulative Gain has been observed.

Categories and Subject Descriptors
H.3.3 [**Information Systems**]: Information Search and Retrieval – Retrieval models; H.4.m [**Information Systems**]: Miscellaneous – Machine Learning

General Terms
Algorithms, Experimentation, Theory

Keywords
Blending, ranking, monotonic transformation, quadratic programming

1. INTRODUCTION
Given a set of items $X = \{x_1,...,x_n\}$, a ranking of X is a permutation of $[n] \equiv \{1,...,n\}$. There have been tremendous amount of studies in the field of learning to rank [1, 2, 5, 8, 12]. However in many applications, we need to integrate the rankings of items from heterogeneous domains into a single ranking of all the items in all the sets, given the emergence of various vertical search engines such as video search, image search, blog search, etc. For example, one set of items can be the set of documents from the Web, and the other can be the set of documents from a vertical search engine such as Blog or News search. Merging the rank lists from heterogeneous domains is a non-trivial topic, because: 1) these heterogeneous sets can share some documents, but most likely they also have many documents that are not in common; 2) Heterogeneous domains usually have different features and feature-to-relevance correlations. Take question-answering websites (e.g. Yahoo! Answer) for one example. Although the text matching and click features developed for general web can be used in the ranking of this domain, features developed with their unique page structures and user feedback, e.g., thumbs up ratings and the total number of feedbacks in Yahoo! Answers, greatly benefit the ranking in its own domain. Even features shared by Yahoo! Answers and general web documents could have very different correlations with the relevance in the two domains. Therefore, using one universal ranking function across domains does not solve the problem well. Dedicated functions are needed in order to better rank documents within each individual domain, and new technologies that blend documents from heterogeneous domains into a single ranking list are greatly needed.

We want to emphasize that this problem is generally different from the rank aggregation [4, 9] problem where one needs to merge the different rankings on the homogeneous set of items.

We define the integration of rank lists from heterogeneous domains as a *blending* problem and formulate the problem of *learning to blend rankings* as follows:

a) We have items of heterogeneous types. The items of each type have a rank order in the corresponding domain.
b) The training data for blending is in the form of pairs of items sets and their associated rankings, the first in the pair belongs to one type of items, and the second belongs to another type of items.

For each pair of item sets, we have a combined ranking of all the items in both of the item sets, presumably indicating the correct blending of the two given rankings.

The optimal combined rankings are ground truth for learning to blend and could be generated in the following two steps: 1) assign relevance labels, e.g. Perfect, Excellent, Good, Fair and Bad (abbreviated to P/E/G/F/B) to each item in both rankings; 2) merge sort the ranking lists according to those labels. Blending in this way will maximize Discounted Cumulative Gain (DCG) [7] while preserving the ordering for both rankings.

CIKM'10, October 26 - 30, 2010, Toronto, Ontario, Canada.

Given the training data – the combined ranking and the rankings in the individual domain, we want to learn a monotonic increasing transformation (on the ranking score in the individual domain) so that when presented with a new pair of item sets and their associated rankings, we can use the transformed ranking scores to generate a combined ranking.

In this paper, we formulate the problem as a quadratic programming problem and learn a linear monotonic transformation so that the rank order in each domain is kept and the transformed scores are comparable.

The rest of the paper is organized as follows: Section 2 describes the notations and the formal formulation of the blending problem. Section 3 develops the main algorithm, where the transformation learning is formulated as a quadratic programming problem. Section 4 shows the promising experimental result, evaluated with real-world data sampled from a commercial search engine. In section 5, we make conclusions and point out directions for future research.

2. PROBLEM FORMULATION

To design a blending transformation, we assume that the training data consist of a set of pairs $\{q_i\}_{i=1}^{Q}$. In this work, we focus on the scenario where the order for each individual ranking is preserved after blending[1]. Blending with this constraint is very like merge sorting.

For simplicity, let us assume we have two rankings. Considering q_i, we will have

$$R_1^i = \{< d_{11}^i, r_{11}^i >, < d_{12}^i, r_{12}^i >, ..., < d_{1M}^i, r_{1M}^i >\}$$
$$R_2^i = \{< d_{21}^i, r_{21}^i >, < d_{22}^i, r_{22}^i >, ..., < d_{2N}^i, r_{2N}^i >\}$$

where M and N are the numbers of items in the first and second set, respectively, and d_{1m}^i and d_{2n}^i are the items.

For the rank order in each domain R_1^i and R_2^i, we consider two situations regarding the format of the rankings: 1) for a set of items, we just have the ranking of the items; and 2) for a set of items, we have a score for each of the item, and the ranking of the items are induced by the scores of the items, i.e., the ranking is obtained by sorting the scores of the items.

Given a pair of item sets and their associated rankings, we can distinguish three cases:

- Both sets are in situation 1). We need to learn a transformation that can relate the ranks in one set to those of another set.
- One set in situation 1) and the other in situation 2). We need to learn a transformation that can relate the ranks in one set to the scores of another set.
- Both sets are in situation 2). We need to learn a transformation that can calibrate the scores in the two sets.

[1] But our method is still applicable even this assumption is not satisfied.

For a ranking in situation 1) r_{1m}^i or r_{2n}^i is just the negative of its rank, and in situation 2) it is the corresponding score. Therefore, the tree cases can be addressed with one formulation.

For R_1^i and R_2^i, we would have

$$r_{11}^i \geq r_{12}^i \geq ... \geq r_{1M}^i$$
$$r_{21}^i \geq r_{22}^i \geq ... \geq r_{2N}^i$$

Correspondingly to R_1^i and R_2^i, we also have the combined ranking with totally $M+N$ items[2]:

$$R^i = \{..., d_{1m}^i, ..., d_{2n}^i, ...\}$$

As required, the order of items from either list is preserved in R^i.

Accordingly, we define two subsets of $\{(m,n)\ 1 \leq m \leq M, m \leq n \leq N\}$: S_i^+ corresponds to cases where d_{1m}^i is ranked above d_{2n}^i, and S_i^- corresponds to cases where d_{1m}^i is ranked below d_{2n}^i, and define

$$S^+ = \bigcup_{q_i \in Q} S_i^+,\ S^- = \bigcup_{q_i \in Q} S_i^- .$$

The key question is how to automatically learn a blending transformation from the training data. We propose to apply a monotonically increasing function $f(\cdot)$ to r_{2n}^i, $n = 1,...,N$ in R_2^i so that the blending would be based on r_{1m}^i and $f(r_{2n}^i)$. By doing so, the order of items from each individual ranking list are automatically preserved. $f(\cdot)$ is learned to be maximally conformed to the editorial blending ranking[3].

3. ALGORITHMS

3.1 Our algorithm

We formulate the transformation learning problem as a quadratic programming problem.

$$\min_f \sum_{k=1}^{K} \xi_k^2$$

subject to

$$r_{1m}^i \geq f(r_{2n}^i) - \xi_k \quad (m,n) \in S_i^+,$$
$$f(r_{2n}^i) \geq r_{1m}^i - \xi_k \quad (m,n) \in S_i^-,$$
$$\xi_k \geq 0$$

where K is the total number of items from both S_i^+ and S_i^- $i = 1,...,Q$.

[2] For simplicity, we assume no overlapping items between the two lists.

[3] Suppose we have X rankings, $X \geq 2$. One will be picked as the reference point and requires no transformation while the remaining $X-1$ transformations should be learned.

If one simply assume that $f(\cdot)$ is linear and in the form of $f(x) = \alpha x + \beta$, the above problem would become

$$\min_{\alpha,\beta,\xi_k} \sum_{k=1}^{K} \xi_k^2 + \lambda_1 \alpha^2 + \lambda_2 \beta^2 \qquad (1)$$

subject to

$$r_{1m}^i \geq \alpha r_{2n}^i + \beta - \xi_k \quad (m,n) \in S_i^+,$$
$$\alpha r_{2n}^i + \beta \geq r_{1m}^i - \xi_k \quad (m,n) \in S_i^-,$$
$$\xi_k \geq 0,\ \alpha \geq 0$$

By solving the above QP problem, we will obtain a (α,β) for the linear transformation (the same (α,β) will be applied to all the queries).

We could also learn a (α,β) for each query length, or each type of queries if query classification information is available. The constraints in Equation (1) are given the same weight, which can be adjusted to give higher weights to more important constraints. Other non-linear monotonic transformation should also be explored in future work.

Equation (1) demonstrates the idea with two domains. The algorithm can be easily extended to blend more than two rankings. Given ranking lists from X domains, one will be picked as the reference point and there will be $X-1$ transformations $(\alpha_1,\beta_1),...,(\alpha_{X-1},\beta_{X-1})$. The constraints to the QP problem will involve all pairs of item sets from any two domains, i.e., the problem becomes

$$\min_{\alpha_v,\beta_v,\xi_k} \sum_{k=1}^{K} \xi_k^2 + \lambda_{1,1}{\alpha_1}^2 + \lambda_{1,2}{\beta_1}^2 + \mathrm{L} + \lambda_{X-1,1}{\alpha_{X-1}}^2 + \lambda_{X-1,2}{\beta_{X-1}}^2$$

subject to

$$\alpha_u r_{um}^i + \beta_u \geq \alpha_v r_{vn}^i + \beta_v - \xi_k \quad (m,n) \in \underset{u,v\,i}{S^+},$$
$$\alpha_v r_{vn}^i + \beta_v \geq \alpha_u r_{um}^i + \beta_u - \xi_k \quad (m,n) \in \underset{u,v\,i}{S^-},$$
$$r_{1m}^i \geq \alpha_u r_{um}^i + \beta_u - \xi_k \quad (m,n) \in \underset{1,u_i}{S^+},$$
$$\alpha_u r_{um}^i + \beta_u \geq r_{1m}^i - \xi_k \quad (m,n) \in \underset{1,u_i}{S^-},$$
$$\xi_k \geq 0,\ \alpha_v \geq 0,\ \mathrm{u},v = 1,...,X-1$$

4. EXPERIMENTS

4.1 Data

We evaluated the proposed algorithm with the problem of blending web search results with vertical search results in the domain of Yahoo! Answers. 1300 queries were sampled from the query logs of a commercial search engine, and 800 queries were used for training and 500 for validation. For each query, we have two sets of documents: general web documents and Yahoo! Answers documents. Each document is labeled with one of five grades Perfect, Excellent, Good, Fair and Bad, in decreasing order of relevance. We have pre-generated ranking functions in each domain and the rank score r_{1m}^i or r_{2n}^i can be generated by applying the ranking function in each domain to the document in the corresponding domain. Given R_1^i and R_2^i, constrains for the QP problem can be constructed by applying merge-sort to the two rank lists and keeping the paired score preference between web documents and Answers documents.

4.2 Experiments

To evaluate the proposed algorithm we focus on the simple case where $f(\cdot)$ is a linear transformation, i.e., $f(x) = \alpha x + \beta$. 800 queries were used to learn the transformation and 500 queries were used for validation.

BASELINE APPROACH. The baseline we compare to is the Naïve blending method, where the scores of r_{1m}^i and r_{2n}^i are compared directly.

EVALUATION METRICS. We report the widely used relevance metric Discounted Cumulative Gain (DCG) [7]. For a ranked list of N documents (N is set to be 10 or 1 in our experiments), we use the following variation of DCG,

$$DCG_N = \sum_{i=1}^{N} \frac{G_i}{\log_2(i+1)},$$

where G_i represents the weights assigned to the label of the document at position *i*, e.g., 10 for Perfect match, 7 for Excellent match, 3 for Good match, etc. Higher degree of relevance corresponds to higher value of the DCG. We use DCG to indicate the average of DCG values over a set of testing queries.

In our application, the goal is to blend the documents from Yahoo! Answers to web rank list. We reported the DCG1 and DCG10, of web rank list and the blended list in Table 1. 1.18% DCG10 gain and 0.9% DCG1 gain were observed from our approach. Both improvements were statistically significant and highlighted with bond font in Table 1. In our application, the choice of λ_1,λ_2 does not affect the performance significantly and our experiments used $\lambda_1 = 1,\lambda_2 = 10$. The Naïve blending method did not achieve any improvement of DCG. This suggests that the rank scores from heterogeneous domains are not directly comparable and a blending algorithm is needed.

Table 1. Performance of linear transformation blending

	Web rank list	**Blended list, via Naïve blending**	**Blended list, via linear transformation**
DCG10	6.78	6.77	**6.86**
DCG1	2.31	2.31	**2.33**

Table 2. Pair-wise error of different blending methods

	Blended list, via Naïve blending	**Blended list, via linear transformation**
Error rate	46%	35%

The pair-wise error rate, i.e., the percentage of pairs of item sets that are not ranked correctly, is also calculated. In other words, this error rate measures how many constraints in the QP problem can not be satisfied. The error rate is reported in Table 2. Even the

learned linear transformation gives an error rate of 35%. Therefore we study the optimal DCG that can be obtained by the – merge-sort strategy.

UPPER BOUND FOR BLENDING. The ideal merge-sort of the two ranking lists can be considered as the best DCG10 that can be obtained via blending, i.e., the upper bound a blending algorithm could achieve. The best DCG10 of our test data is 7.06. Therefore there is room to improve for the blending algorithm. Section 5 will discuss the future research directions.

5. RELATED WORK

In recent years, the ranking problem is frequently formulated as a supervised machine learning problem [3, 6, 11]. These learning-to-rank approaches are capable of combining different kinds of features to train ranking functions. The problem of ranking can be formulated as that of learning a ranking function from pair-wise preference data. The idea is to minimize the number of contradicting pairs in training data. For example, RankSVM [8] uses support vector machines to learn a ranking function from preference data. RankNet [1] applies neural network and gradient descent to obtain a ranking function. RankBoost [5] applies the idea of boosting to construct an efficient ranking function from a set of weak ranking functions. The studies reported in [12] proposed a framework called GBRank using gradient descent in function spaces, and the weak leaner is a decision tree. Cao et al. [2] proposed the listwise approach to handle the ranking problem. Furthermore, Rank aggregation [4, 9] targets the problem of merging the different rankings on the homogeneous set of items, where items belong to the same domain.

Our algorithm of formulating a pairwise ranking problem as a quadratic programming problem was inspired by the method described in [10].

6. CONCLUSIONS AND FUTURE WORK

Emergence of various vertical search engines such as video search, image search, and blog search, motivates the development of algorithms to blend rank lists from multiple domains. Unlike the traditional learning to rank or rank aggregation problem within one domain, in this paper we study the problem of combining rank lists from heterogeneous domains to obtain one single rank list. The task of *learning to blend rankings* is defined.

The *blending* problem is a challenging task. There are few documents/features in common among heterogeneous domains. Therefore the ranking function for each type of documents needs to be learned within the domain. However the rank scores of ranking functions in heterogeneous domains are not directly comparable, which brings difficulty for blending. To determine the optimal combined list, we consider a merge-sort-like strategy to combine ranking lists based on relevance, to maximize the DCG and preserve the ranking order in each domain. To achieve such a combined ranking list with the ranking lists from multiple domains, a monotonic transformation is applied to the rank scores in each domain, such that the transformed scores become comparable.

Learning the monotonic transformation that could achieve the optimal blended list, is formulated as a quadratic programming problem. In the paper, we focus on the simple case where a linear transformation is considered.

We evaluated the novel learning-to-blend approach with real-world data sampled from a commercial search and observed promising results of 1.18% DCG10 gain.

We focused on one single linear transformation for all queries in this paper. Adapting query-type dependent monotonic transformation is a direction to explore in future work. Advanced monotonic non-linear transformation can also be explored in future work. Also exploring the application to more than two domains is of great interests.

7. REFERENCES

[1] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier,Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In ICML '05:Proceedings of the 22nd international conference on Machine learning, pages 89–96, 2005.

[2] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In ICML, pages 129–136, 2007.

[3] C. Cortes, M. Mohri, and A. Rastogi. Magnitude-preserving ranking algorithms. In Proceedings of the 24th ICML, 2007.

[4] Cynthia Dwork , Ravi Kumar , Moni Naor , D. Sivakuma. Rank Aggregation Methods for the Web , In the Proceedings of the 10th international conference on World Wide Web, pages 613-622, 2001.

[5] Y. Freund, R. Iyer, R. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In Proceedings of the Fifteenth International Conference on Machine Learning, 1998.

[6] J. Guiver and E. Snelson. Learning to rank with SoftRank and Gaussian processes. In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, 2008.

[7] K. Jarvelin and J. Kekalainen. Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems, 20:422-446, 2002.

[8] T. Joachims. Optimizing search engines using clickthrough data. In Proceedings of ACM SIGKDD, 2002.

[9] Yu-Ting Liu, Tie-Yan Liu, Tao Qin, Zhi-Ming Ma, Hang Li. Supervised rank aggregation, In Proceedings of the 16th international conference on World Wide Web, pages: 481 - 490 , 2007.

[10] Taesup Moon, Alex Smola, Yi Chang and Zhaohui Zheng. IntervalRank - Isotonic Regression with Listwise and Pairwise Constraints. In WSDM, pages 151 - 160, 2010

[11] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In Proceedings of the 30th ACM SIGIR, 2007.

[12] Zhaohui Zheng, Hongyuan Zha, and Gordon Sun. Query-level learning to rank using isotonic regression. In the 46th Annual Allerton Conference on Communication, Control and Computing, 2008.