Boosting for Document Routing

Raj D. Iyer David D. Lewis
Robert E. Schapire Yoram Singer
Amit Singhal

報告者：黃立德

Outline

- Introduction
- The concept of boosting for ranking
- The details of boosting
  - Weak learner
  - The number of rounds
- Term weighting
- Experiment results
  - Study 1: Boosting with real-valued weak hypothesis
  - Study 2: Learning term-weighting functions from raw data
- Conclusion and future work

Introduction

- RankBoost is a recently proposed algorithm for learning ranking functions.
- It is simple to implement and has strong justifications from computational learning theory.
- RankBoost archieves comparable performance to the state-of-the-art algorithm when combined with feature or example selection heuristics.

Introduction (cont.)

- Most IR techniques for ranking have two facets
  - Model fitting : the selection and weighting of terms for a particular information need
  - Term weighting : the conversion of textual documents first into tokens for index terms
- The paper reports an attempt to bring machine learning approaches to bear on both of these facets of ranking.

The concept of boosting for ranking

- RankBoost is based on Freund and Schapire's AdaBoost algorithm.
- The goal of RankBoost is to produce a statistical model that, when applied to a set of documents, orders them in a fashion that approximates their true order, that is, an ordering according to relevant.
- RankBoost attempts to minimize one possible measure that is called pair-wise disagreement.

$$disagree_D(h) = \sum_{d_0, d_1 : h_s(d_0) \geq h_s(d_1)} D_s(d_0, d_1)$$

The concept of boosting for ranking (cont.)

- The idea of boosting is to generate and combine many different simple rankings in a principled manner to produce a single highly accurate ordering.
  - The simple rankings are real-valued function called weak hypotheses.
  - An algorithm or subroutine for generating these rankings is called weak learner.
  - These weak hypotheses are finally combined into a single ordering called the final hypothesis.

The concept of boosting for ranking (cont.)

- Example
  - Given a ranking h and document d, we refer to $h(d)$ as the *score* that h assigns d.
  - h orders d above d' if $h(d) > h(d')$.
- The boosting algorithm proceeds in rounds. During the course of its execution, it assigns different importance weights to different pairs of training documents.
  - These weights are not maintained for all possible document pairs.
  - Pairs of documents that are hard to differentiate correctly get higher weights.

$$disagree_D(h) = \sum_{d_0, d_1 : h_s(d_0) \geq h_s(d_1)} D_s(d_0, d_1)$$

The concept of boosting for ranking (cont.)

- The final hypothesis orders a set of new documents by assigning to each a real-valued score which is a weighted combination of the score assigned to that document by the weak hypotheses.

$$H(d) = \sum_{s=1}^{T} \alpha_s h_s(d)$$

Where d is a document in dataset, H is the final hypothesis

h_s is the weak hypothesis on round s , $\alpha_s \in R$

The concept of boosting for ranking (cont.)

- How to choose α_s?
- Freund prove the pair-wise disagreement of final hypothesis H is bounded above by the product of the normalization factors.

$$\prod_{s=1}^{T} Z_s$$

- For general weak learners whose hypotheses assign documents arbitrary real numbers, the authors suggest finding α_s via numerical search.

The concept of boosting for ranking (cont.)

- If the hypotheses generated by the weak learner have the range {0,1} , Freund provide a direct calculation of α_s :

$$\alpha_s = \frac{1}{2}\ln\left(\frac{1+r_s}{1-r_s}\right)$$

where

$$r_s = \sum_{d_0,d_1} D(d_0 - d_1)\left(h_s(d_1) - h_s(d_0)\right)$$

The concept of boosting for ranking (cont.)

Input a set X of documents separated into disjoint subsets X_1 of relevant documents and X_0 of non-relevant documents;
Initialize $\forall (d_0, d_1) \in X_0 \times X_1 : D_1(d_0, d_1) = 1/(|X_0|\,|X_1|)$.
Do for $s = 1, \ldots, T$:

1. Train WeakLearn using distribution D_s over $X_0 \times X_1$.
2. WeakLearn returns a weak hypothesis $h_s : X \to \mathbb{R}$.
3. Compute $\alpha_s \in \mathbb{R}$.
4. Update: $\forall (d_0, d_1) \in X_0 \times X_1$:

$$D_{s+1}(d_0, d_1) = \frac{D_s(d_0, d_1) \exp\left(-\alpha_s \left(h_s(d_0) - h_s(d_1)\right)\right)}{Z_s}$$

where Z_s is the normalization factor:

$$Z_s = \sum_{d_0, d_1} D_s(d_0, d_1) \exp\left(-\alpha_s \left(h_s(d_0) - h_s(d_1)\right)\right) .$$

Output the final hypothesis: $H(d) = \sum_{s=1}^{T} \alpha_s h_s(d)$.

Figure 1: RankBoost algorithm for document routing

The details of boosting - Weak learner

- A weak learner takes as input a distribution D over the document pairs and a set of N *ranking features*.
  - A ranking feature is a function that assigns a real-valued score to a document and also allowed to leave some documents unranked.
- The weak learner attempts to find one weak hypothesis which minimizes Z instead of minimizing the pair-wise disagreement.

The details of boosting - Weak learner (cont.)

- Two types of weak learner
  - WeakReal
    - Simply select one of the available ranking features to be the weak hypothesis.
    - The algorithm calculates the α which minimizes Z by running Newton's method.
  - WeakThreshold
    - Select a ranking feature and convert it into a binary (0-1) function by choosing a threshold score.
    - Search for h to maximize $\left|r\right|$ rather than minimize Z.

$$Z \le \sqrt{1-r^2} \quad \text{(proved by Freund)}$$

The details of boosting - Weak learner (cont.)

– A set of ranking features $\{f_i\}$

$$h(d) = \begin{cases} 1 & \text{if } f_i(d) > \theta \\ 0 & \text{if } f_i(d) \leq \theta \text{ or } f_i(d) = \perp \end{cases}$$

The details of boosting – the rounds (cont.)

- Although the algorithm runs for more rounds, the final hypothesis predicts more accurately on the training data, but it may overfit and not generalize well to future data.
- The author use heuristics to estimate a good number of rounds of boosting.
- When running on a particular topic, they looked at the number of features in the feature set and the number of positive examples in the training set, and set T to be the smaller of the two.

Term weighting

- Applying a standard term weighting function seems peculiar with WeakThreshold
  - real-valued feature values simply assign to 0/1
- The rawtf representation
  - Define one ranking feature, f_t ,for each term t in one document is equal to the *raw tf*.

$$h(d) = \begin{cases} 1 & \text{if } tf \geq \theta \\ 0 & \text{if } tf < \theta \end{cases}$$

Term weighting (cont.)

– RankBoost may choose several weak hypotheses based on the same indexing term.

$$g(d) = \begin{cases} w_n & \text{if } tf \geq \theta_n \\ w_{n-1} & \text{if } tf \geq \theta_{n-1} \text{ and } tf < \theta_n \\ w_{n-2} & \text{if } tf \geq \theta_{n-2} \text{ and } tf < \theta_{n-1} \\ \cdots & \\ 0 & \text{if } tf < \theta_1 \end{cases}$$

– Larger value of tf cause a term to have a large contribution to a document's score.

Term weighting (cont.)

- *raw-tf&length* representation
  - It is common in term weighting to use document length normalization.

$$h(d) = \begin{cases} 1 & \text{if } length \leq \theta \\ 0 & \text{if } length > \theta \end{cases}$$

  - Shorter documents get a boost to their score.

Term weighting (cont.)

- Q (quadrant) representation
  - We can also define features that allow a document length to affect each term differently.
  - Define multiple ranking features for each term.

$$h(d) = \begin{cases} 1 & \text{if } tf \geq u \text{ and } length \leq \theta \\ 0 & \text{otherwise.} \end{cases}$$

Experiment results

- Datasets
  - TREC-3 dataset
    - Use 50 TREC topics
  - Reuters-21578
    - Use 90 categories that have at least one positive training instance and at least one positive test instance .
- Both datasets were indexed using standard SMART raw *tf* and *Lnu* (normalization)
- Use Rocchio's relevant feedback formula (Rocchio-QZ-DFO) to compare with it.

Experiment results (cont.)

- Study 1: Boosting with real-valued weak hypothesis
  - Focus on issues of model fitting.
  - Use WeakReal as a weak learner applied in a standard IR context.
  - Each ranking feature is associated with a term whose value is given by SMART.

Experiment results (cont.)

	Rocchio-QZ-DFO	RankBoost with WeakReal	RankBoost with WeakThreshold
Reuters-21578			
all topics	0.6786	0.6284	0.5836
≥ 5 rel. test	0.8078	0.7501	0.7446

Table 1: Non-interpolated average precision results for Reuters-21578 and TREC-3. RankBoost uses all terms that occur in at least two positive training documents as features. The WeakThreshold version uses the Q representation (Section 4.3) based on the same terms. The second Reuters line gives results restricted to the 59 topics with at least five relevant test documents.

Experiment results (cont.)

	Rocchio-QZ-DFO	RankBoost with WeakReal	RankBoost with WeakThreshold
Reuters-21578			
all topics	0.6786	0.6956	0.5817
≥ 5 rel. test	0.8078	0.8455	0.7483
TREC-3			
all documents	0.4669	0.3974	0.4276
top 1,000 + rel.	0.4669	0.4638	–

Table 2: Non-interpolated average precision results for Reuters-21578 and TREC-3. RankBoost here is restricted to using features given a nonzero weight by Rocchio-QZ-DFO. The WeakThreshold version uses the Q representation (Section 4.3). The second Reuters line gives results only on topics with at least five relevant test documents. The second TREC line gives results for the top 1,000 test documents as ranked by Rocchio-QZ-DFO, plus any remaining relevant test documents.

Experiment results (cont.)

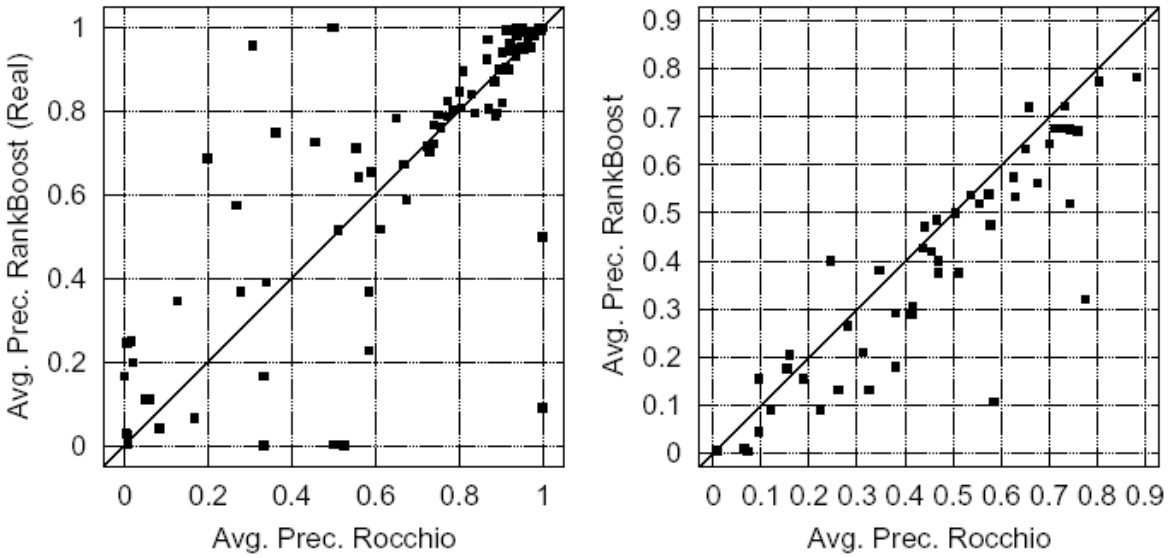

Figure 2: Comparison of RankBoost with WeakReal and Rocchio-QZ-DFO on Reuters-21578 (left) and TREC-3 (right) text collections. Each point in each scatterplot shows the test average precision of the two competing algorithms on a single topic. The x- and y-coordinates of each point give the test average precision of Rocchio-QZ-DFO and RankBoost (respectively) on the given topic.

Experiment results (cont.)

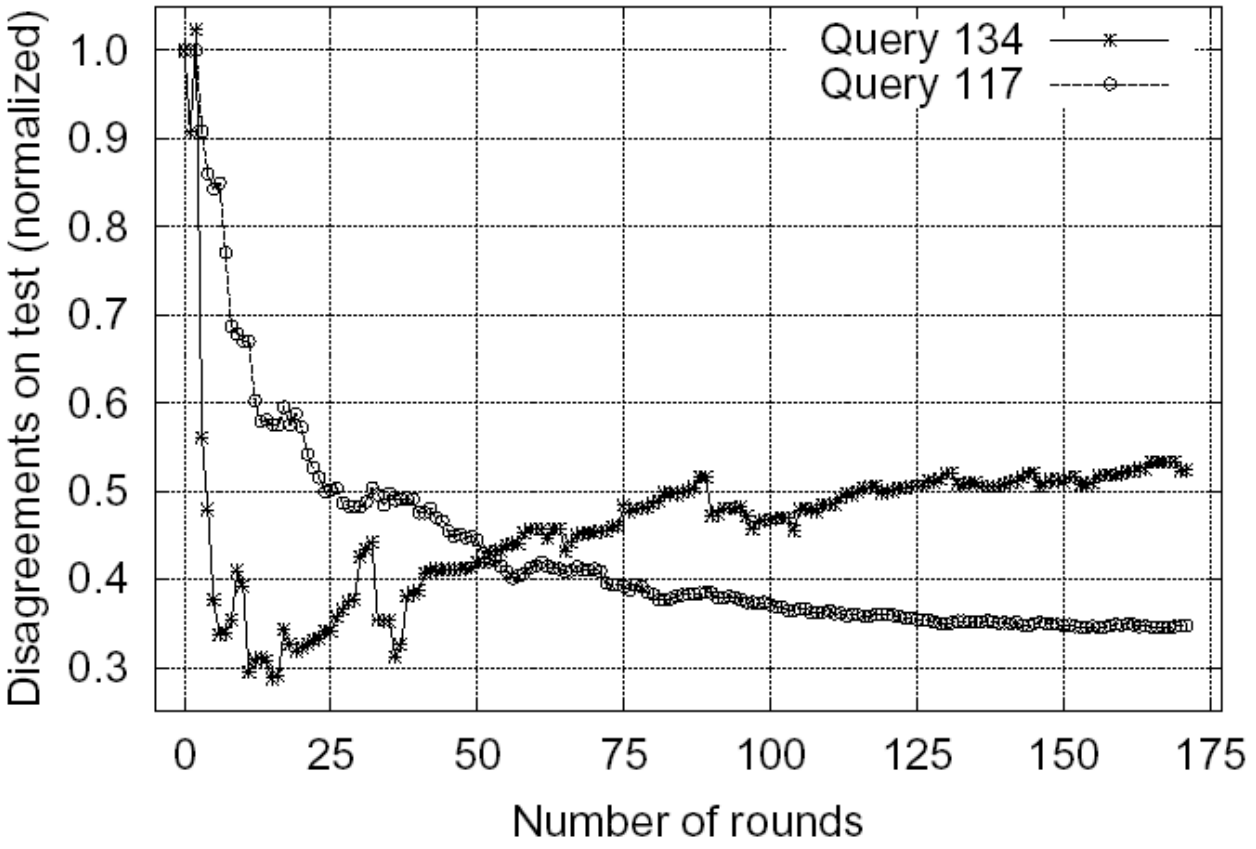

Figure 4: For two TREC-3 topics, the pairwise disagreement on a test set for RankBoost with WeakReal, measured as a function of the number of rounds of boosting.

Experiment results (cont.)

- Study 2: Learning term-weighting functions from raw data
  - Focus on the character of learned term weighting functions.
  - Use WeakThreshold as a weak learner applied in each of these form.
    - rawtf , rawtf&length ,and Q representation

Experiment results (cont.)

	rawtf	*rawtf&length*	*Q*
Reuters-21578			
all features	0.6420	0.6557	0.5836
Rocchio features	0.5834	0.5952	0.5638
TREC-3			
all features	0.3797	–	0.4233
Rocchio features	0.3909	–	0.4276

Table 3: Non-interpolated average precision results for Reuters-21578 and TREC-3 using RankBoost with Weak-Threshold. Six text representations are compared: *rawtf*, *rawtf&length*, and *Q* (Section 4.3), plus the versions of these restricted to features selected by Rocchio ($rawtf_r$, $rawtf\&length_r$, and Q_r). TREC-3 data for the *rawtf&length* representations was lost due to a bug.

Experiment results (cont.)

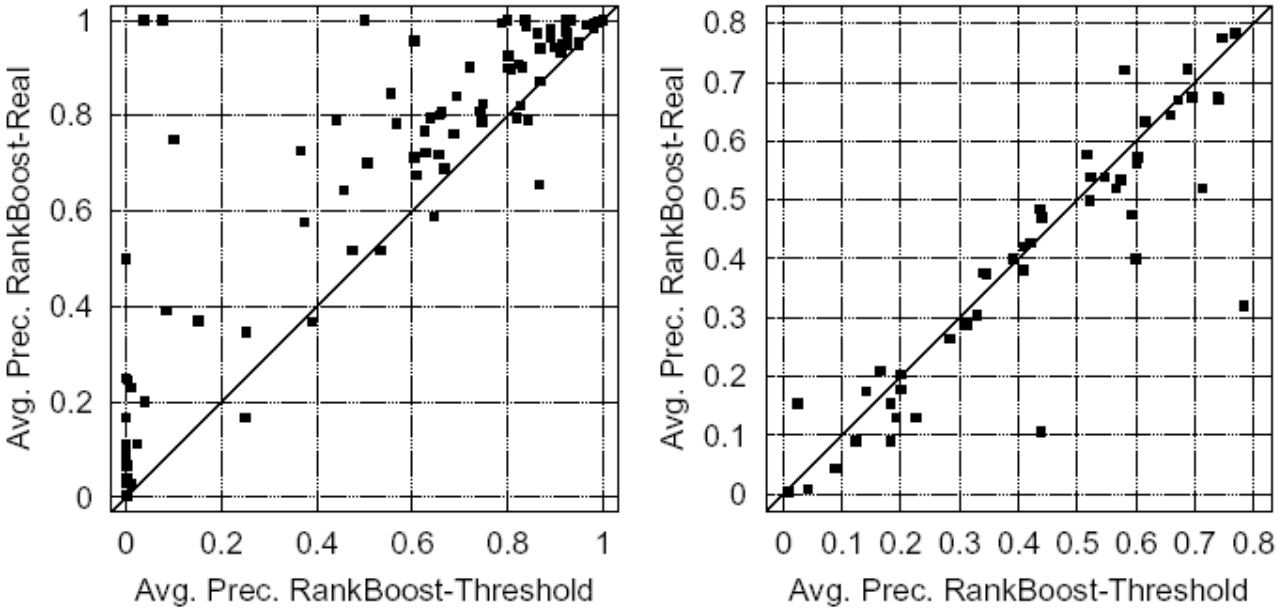

Figure 3: Comparison of RankBoost with two weak learners, WeakReal and WeakThreshold on Reuters-21578 (left) and TREC-3 (right) text collections. WeakThreshold uses the Q_r representation. (See Figure 2.)

Conclusion and future work

- The author have presented preliminary evidence that RankBoost,a simple boosting algorithm with strong theoretical foundations.
- Our results show that RankBoost tends to overfit in the absence of some additional feature selection mechanism
- On a restricted feature set however, results are quite competitive, particularly when substantial training data is available.
- Strengthen the case for RankBoost by replacing the use of Rocchio-QZ-DFO for feature selection.
- produce a more efficient implementation in the future.