Best Document Selection Based on Approximate Utility Optimization

Hungyu Henry Lin
University of California, Santa Cruz
hhlin@soe.ucsc.edu

Yi Zhang
University of California, Santa Cruz
yiz@soe.ucsc.edu

James Davis
University of California, Santa Cruz
davis@soe.ucsc.edu

ABSTRACT

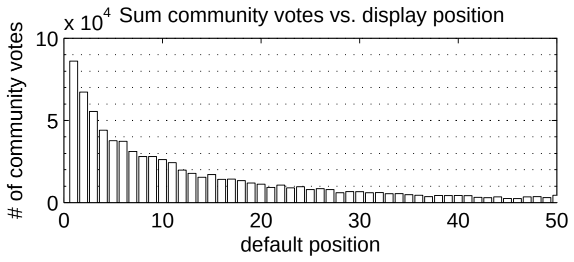
This poster describes an alternative approach to handling the best document selection problem. Best document selection is a common problem with many real world applications, but is not a well studied by itself; a simple solution would be to treat it as a ranking problem and to use existing ranking algorithms to rank all documents. We could then select only the first element of the sorted list. However, because ranking models optimize for all ranks, the model may sacrifice accuracy of the top rank for the sake of overall accuracy. This is an unnecessary trade-off.

We begin by first defining an appropriate objective function for the domain, then create a boosting algorithm that explicitly targets this function. Based on experiments on a benchmark retrieval data set and Digg.com news commenting data set, we find that even a simple algorithm built for this specific problem gives better results than baseline algorithms that were designed for the more complicated ranking tasks.

1. INTRODUCTION

Selecting the best document from a set (i.e. Best document selection) is a common problem with many real world applications. On a web search engine like Google, a widely used functionality is the "I'm feeling lucky" button that leads user directly to the highest ranked URL. On a content web site like nytimes.com, a single ad needs to be selected and displayed on the banner for each visitor. On a news commenting web site like Digg.com, one task is to promote one comment onto a highly-visible sidebar to entice visitors to participate in the discussion.

One method is through community feedback, but this is not always enough. While the community on Digg.com may vote for and against comments - ideally so the best comment will be elected as such, we observe that vote density is highly correlated with the comment's default sort position (Fig 1). Moreover, users are more likely to vote up than down and comments off the first page receive little attention (Fig. 2).



Votes aggregated for top level comments on popular stories (> 50 top level posts) on digg.com over a 6 month period.

Figure 1: Promoting content by community input is not always enough; while the site's community may vote up or down on content, vote density is heavily skewed towards the most exposed.

Best document selection is not a well studied problem by itself. One may think we can just cast it as a binary classification problem [1]. However, this approach lacks the ability to distinguish between multivariate levels of quality of the information. Another simple solution is just treat it as a ranking problem and use existing ranking algorithms to rank all documents (e.g. [5, 10], etc.). Then we can select only the first element from the sorted list. However, because ranking models optimize for all ranks, the model may sacrifice accuracy of the top rank for the sake of overall accuracy. This is an unnecessary trade-off.

We describe an alternative approach to handle the best document selection problem. We do this by first defining an appropriate objective function for the domain, then create a boosting algorithm that explicitly targets this function. Because of the comparative simplicity of the objective function and the special characteristics of the best document selection problem, we can use a stronger and tighter approximation to optimize the objective function than existing approximated ranking solutions. Based on experiments on a benchmark retrieval data set and Digg.com news commenting data set, we find that even a simple algorithm built for this specific problem gives better results than baseline algorithms that were designed for the more complicated ranking tasks.

1.1 Related Work

Reputation Systems. Prior work has been done for reputation systems on social sites, where-in *users*, not content, is graded or ranked by quality. Reputation systems can be seen as ranking models for users - indeed, many approaches are inspired by existing ranking models. Ideally, high-quality users consistently generate high-quality con-

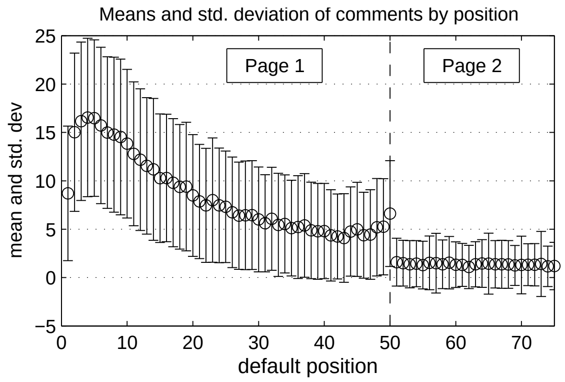


Figure 2: The community can vet for quality comments by either voting up or down on them. However, the means are correlated with the comment's default position, not necessarily with quality. Also, the second page receives little exposure or voters.

tent, but reputation systems may not be ideal for all user content-driven sites. For example, in the scenario given in the Background section, using a reputation system as a basis for selecting comments would to lead to disastrous results; the top-ranked user could simply dominate all promoted comments and over time lead to promoted comments coming from the same subset of active users.

Some algorithms treat site activity, including other forms of reputation management (*e.g.* user feedback systems), as indicative of inter-user relationships. These relationships can then be used to induce a graph. Guha et al. [7] explore trust propagation with a series of graph-based algorithms. Again, site activity is used to create a set of relationships between users - in this case, the relationship is whether an user trusts or distrusts another user. Since not all relationships can be always observed from site feedback, it is the job of each algorithm to take in an incomplete set of relationships and to infer the remaining unknown relationships. These relationships can be used to find bodies of high-quality users (such as the users trusted by most of the population pool).

Approaches inspired by PageRank [16] and HITS [14] are also used. Zhang et al. [25] use a PageRank-like algorithm to rank Java expertise amongst members of a Java Question-Answer forum, again by observing site activity to deduce relationships between users. Jurczyk and Agichtein [12] have applied the HITS algorithm [14] to find high-quality users for general Question-Answer sites. Later, Agichtein et al. [1] tackle the same domain, now casting the problem as a binary classification problem; the authors use boosting trees to classify between good answers and bad ones (and hence not a reputation system). However, the bulk of the work, which uses features applicable only to Question-Answer sites, remains domain restricted.

Machine-learned Ranking. Ranking models can also be used for content promotion. Ranking models seek to order documents by relevance, quality, or desirability. We focus here on supervised learning approaches.

RankBoost [5] is a pair-wise boosting ranker that combines a series of weak pair-wise rankers into a strong one. As such, documents are treated as pairs and the goal is to classify the better (that is, larger label) of each pair, generating a relative ordering as a result. Owing to AdaBoost's re-weighing mechanic, at the top of each iteration RankBoost increases the weights of mis-classified pairs and decreases weights of correctly ordered pairs. A weak classifier that prioritizes correctly classifying frequently mis-classified pairs is then trained. In [19], Rudin and Schapire prove that AdaBoost is as good as RankBoost in ranking tasks.

FRank [4] (Fidelity Rank) is based on RankNet [2] and uses a loss function called Fidelity loss. The name derives itself from the quantum fidelity metric used in physics defining the distance between two quantum states. If fidelity of two states is close to one, then the states are close and near zero means the states are far apart. The concept is used here as the "fidelity" of two distributions; near one means the two distributions are nearly identical and so on. With this, the authors induce a distribution model over the relative ordering of documents (a pair-wise model) using training labels and the goal of the algorithm is to build a classifier, via boosting, whose induced distribution is close to distribution induced from training labels. FRank, as such, does not optimize over NDCG.

AdaRank [8] is a list-wise boosting algorithm which embeds ranking measures (NDCG, MAP, etc.) in its update computations. Any arbitrary measure may be used by AdaRank, including NDCG@1, yet the algorithm does not directly optimize any measure but rather uses them heuristically.

NDCG_boost [23] is another list-wise boosting algorithm which makes the point of basing its objective function off of the NDCG measure itself. This results in an objective function that is a relaxed and approximated lower bound of NDCG. To smooth out the step-wise nature of NDCG, the authors map predictions to a probability model based, roughly, on the pair-wise difference of document scores. At each stage the algorithm generates a new set of not only weights but binary labels as well; as such, the label of a document may change from iteration to iteration and weak learners are restricted to binary classifiers. We use a similar re-labeling approach here, although we do not use the same relaxation and approximation approach.

Ranking algorithms seek to optimize the overall ordering of queries, not the top-ranked item. Because we seek to optimize the first rank of the ordering, we should instead make the trade-off of higher first rank accuracy but poorer *overall* ranking performance. We also note some ranking models can be tuned for the top rank; for example, BM25 is often tuned over NDCG@1 [22]. Other examples lead to the same limitation of merely tuning to give best possible scores for the given algorithm as opposed to explicitly optimizing for it.

Lexical Tests. Finally, lexical ranking functions, such as tf-idf [21] and similar schemes (*e.g.* BM25 [18]) can be used. Other readability tests such as Flesch-Kincaid [13], SMOG [15]) give a rough assessment of readability. These simple tests can be used as features for higher level machine learning algorithms, such as in [1] as well as in this paper.

2. PRELIMINARIES

This section is used to lay out relevant measures used in our work.

Precision at n (**precision@n** or p@n) measures the ratio of relevant documents in a sorted list up to a truncation point. It takes in a set of sorted documents and a binary judgement on the relevance (e.g. relevant' or 'not relevant') of each document and outputs:

$$\text{P@}n = \frac{\textit{\# of relevant documents in top n ranks}}{n} \tag{1}$$

A special case of P@n, **Winner Takes All** (WTA), is sometimes used. WTA is equivalent to p@1 - WTA returns 1 if the top ranked document is relevant, 0 otherwise.

Average Precision extends p@n to emphasize the importance of placing relevant documents at the top of the sorted list. Its equation is:

$$\text{AveP} = \frac{\sum_n \text{P@}n * r_n}{\textit{\# of relevant documents}} \tag{2}$$

where r_n is the (real or integer) relevance label of the nth document in the sorted list of documents. For a set of queries, **Mean Average Precision** (MAP) is usually reported - as the name suggests it is simply the mean AveP scores for all queries.

Precision-based measures suffers the issue of only supporting binary labels. In practice, there are different grades of relevance. **Normalized Discounted Cumulative Gain** (NDCG) [9], in contrast to MAP, uses integer or real valued labels. It is defined as:

$$\text{NDCG} = Z \sum_n \frac{2^{r_n} - 1}{\log_2(1+n)} \tag{3}$$

where Z is a normalization term chosen such that the optimal ranking nets score of 1. Like MAP, NDCG emphasizes the importance of placing highly relevant documents at the front of the list. Also like MAP, the mean NDCG score is usually reported for a set of queries. For the truncated variant of NDCG, NDCG@n, we take the first n documents in the sorted list in the same fashion as precision@n.

3. OVERVIEW OF THE APPROACH

We now introduce our new measure, Utility (U), and an algorithm which optimizes for our measure.

3.1 Utility Function

For the task of selecting the best document(s), we measure the utility as the average ratio between the selected document's relevance score and the best one:

$$U = \frac{1}{|Q|} \sum_{k=1}^{|Q|} \frac{r_{k,s}}{r_{k,o}} \tag{4}$$

$$= \frac{1}{|Q|} \sum_{k=1}^{|Q|} \frac{1}{r_{k,o}} \sum_i r_{k,i} [d_{k,i} \textit{ is selected}] \tag{5}$$

where Q is the set of queries, and $r_{k,s}$ and $r_{k,o}$ are the relevance scores of the $s$elected and $o$ptimal documents for the kth query, respectively. $d_{k,i}$ is a ith candidate document for the kth query, and $r_{k,i}$ is its relevance score. [] is the Iverson bracket that returns 1 for when the condition in the bracket is true, 0 otherwise. This measure is equivalent to WTA, NDCG@1, and p@1 for binary labels. Otherwise, U can be seen as a multivariate variant of p@1, MAP@1, and WTA as well as a non-exponentiated version of NDCG@1.

3.2 Approximate Objective Function

The utility measure in Equation 4 can not be optimized easily using gradient descent methods, since it is not differentiable. Similar to what people has done for optimizing ranking measures such as MAP or NDCG, we find an approximate solution by constructing a new approximate objective function that is differentiable. To do so, we approximate the Iverson bracket [] with a softmax function, which is commonly used in machine learning and statistics, for mathematical convenience. Thus the approximated objective function is:

$$\mathcal{M}(\vec{H}) = \frac{1}{|Q|} \sum_{k=1}^{|Q|} \frac{1}{r_{k,o}} \sum_i r_{k,i} \overbrace{\frac{\exp(\beta * H_{k,i})}{\sum_j \exp(\beta * H_{k,j})}}^{\approx [d_{k,i} \textit{ is selected}]}$$
$$+ \lambda * \underbrace{\sum_i \exp(\beta * H_{k,i})^2 / (\sum_j \exp(\beta * H_{k,j}))^2}_{= l_2^2(\cdot)} \tag{6}$$

where H is the model (e.g. learner or hypothesis) to be learned, which can predict a relevance score for each document query pair, j is the index for candidate documents, $H_{k,i}$ is the retrieval score of document i for query k estimated by hypothesis H. β is a coefficient that controls the tightness of the approximation. Note that the function approaches the indicator as β approaches infinite. $l_2^2(\cdot)$ is introduced as a regularizer to control model complexity and avoid the overfitting problem, and λ is a pre-set parameter that controls the degree of regularization.

3.3 Algorithm: CommentBoost

The approximated objective function in Equation 6 can be optimized via gradient boosting. We employ a simple stage-wise gradient technique à la AdaBoost [6]. We call this algorithm CommentBoost[1] or CBoost@1. It is a fast, list-wise boosting algorithm for best document selection. At each stage t, CBoost@1 chooses the direction $\vec{w}_t$ of steepest ascent at the current hypotheses vector $\vec{H}$. This direction becomes the basis for target weights and labels when we train a weak learner.

After weak hypothesis $\vec{h}$ is obtained from the newly-trained learner, the coefficient α is chosen to weigh the learner. At the final step of the stage, the current hypothesis is updated as $\vec{H} \leftarrow \vec{H} + \alpha\vec{h}$ and the next stage begins.

The final algorithm makes real-valued score predictions as:

$$H(d_{k,i}) = \sum_{t=1}^{T} \alpha_t h_t(d_{k,i}) \tag{7}$$

Sorting each document by highest-score first produces the final total ordering of documents.

3.4 Algorithm Derivation

The first step at the start of stage t is to find binary target labels and weights over the training documents. Given the

[1]Because one major motivation for this algorithm is to select the best comment for users to discuss at Digg.com

```
Initialize: $H_{k,i} = 0$ for all document-query pairs.
            $\beta \geq 0$ as a temperature parameter.
            $\lambda \geq 0$ as a regularization parameter.

For $t = 1, ..., T$:
    • For each document-query pair, compute
          $w_i^k = \frac{\partial}{\partial H_{k,i}} \mathcal{M}(\vec{H})$
    • Train a weak learner $h_t(d)$ that correlates well with
      $\vec{w}$ and get a weak ranking $h : d_{k,i} \rightarrow h_{k,i} \in \mathbb{R}$
    • Compute:
          $\alpha_t = \frac{d}{d\gamma} \mathcal{M}(\vec{H} + \gamma \vec{h})\big|_{\gamma=0}$
    • Update hypotheses:
          $\vec{H} \leftarrow \vec{H} + \alpha_t \vec{h}$

Output the final hypothesis: $H(d_{k,i}) = \sum_{t=1}^{T} \alpha_t h_t(d_{k,i})$
```

Figure 3: The CBoost@1 algorithm

current hypothesis H, the gradient vector at $\vec{H}$ is

$$\begin{aligned}\frac{\partial}{\partial H_{k,i}} \mathcal{M}(\vec{H}) = &\frac{\beta}{r_{\max}^k} \frac{\exp(\beta H_{k,i})}{\sum_{j=1} \exp(\beta H_{k,j})} \\ &* \left(r_i^k - \frac{\sum_{j=1} r_j^k \exp(\beta H_{k,j}}{\sum_{j=1} \exp(\beta H_{k,j})} \right) \\ &+ 2\lambda\beta \left(\frac{\exp(\beta H_{k,i})^2}{\left(\sum_{j=1} \exp(\beta H_{k,j})\right)^2} \right. \\ &\left. - \frac{\exp(\beta H_{k,i}) \sum_{j=1} \exp(\beta H_{k,j})^2}{\left(\sum_{j=1} \exp(\beta H_{k,j})\right)^3} \right) \qquad (8)\end{aligned}$$

A weak learner that outputs this vector gives us the greatest ascent at our objective function. If binary classifiers are used as weak learners, then this results in setting each label as $y_i^k = \texttt{sign}(\frac{\partial}{\partial H_{k,i}} \mathcal{M}(\vec{H}))$ and weights as $w_i^k = \texttt{abs}(\frac{\partial}{\partial H_{k,i}} \mathcal{M}(\vec{H}))$. However, any weak learners that may output real values (*e.g.* point-wise rankers) can be accepted as long as it is well correlated with the gradient. We then choose an α such that $\mathcal{M}(\vec{H} + \alpha\vec{h}) \geq \mathcal{M}(\vec{H})$.

Finding a closed-form solution for α is done by plugging our update scheme $\vec{H} + \gamma\vec{h}$ into $\mathcal{M}$, solving the derivative for γ, then setting $\gamma = 0$.[2]

[2]Note that by taking $\frac{d}{d\alpha}\Psi(\alpha, \vec{\delta}, \vec{\Theta}) = -\alpha + \frac{\partial}{\partial\gamma}\mathcal{M}(\vec{\Theta} + \gamma * \vec{\delta})\big|_{\gamma=0}$, $\frac{d}{d\alpha}\Psi(\alpha, \vec{h}, \vec{H})$ is less than $\frac{d}{d\alpha}\mathcal{M}(\vec{H} + \alpha * \vec{h})$ when $\alpha < 0$ and greater than $\frac{d}{d\alpha}\mathcal{M}(\vec{H} + \alpha * \vec{h})$ when $\alpha > 0$. Thus there $\exists c \in \mathbb{R}$ s.t. $\Psi(0, \vec{h}, \vec{H}) + c = \mathcal{M}(\vec{H})$ and $\Psi(\alpha, \vec{h}, \vec{H}) + c \leq \mathcal{M}(\vec{H} + \alpha\vec{h})$ for all $\alpha \in \mathbb{R}$, satisfying properties for a lower bound whose optimal solution improves $\mathcal{M}$ (See [20]). Setting $\frac{d}{d\alpha}\Psi(\alpha, \vec{h}, \vec{H}) = 0$ and solving for α gives us our closed form solution.

$$\begin{aligned}\frac{d}{d\gamma}\mathcal{M}(\vec{H} + \gamma\vec{h})\big|_{\gamma=0} = &\sum_{k=1}^{|Q|} \sum_{i=1} \frac{\beta}{r_{\max}^k} \frac{r_i^k \exp(\beta H_i^k)}{\sum_{j=1} \exp(\beta H_{k,j})} \\ &* \left(H_{k,i} - \frac{\sum_{j=1} H_{k,j} * \exp(\beta H_{k,j})}{\sum_{j=1} \exp(\beta H_{k,j})} \right) \\ &+ 2\lambda\beta \left(\frac{\sum_{j=1} h_j \exp(\beta H_j)^2}{\left(\sum_{j=1} \exp(\beta H_{k,j})\right)^2} \right. \\ &\left. - \left(\sum_{j=1} \exp(\beta H_{k,j})^2 \right) \frac{\sum_{j=1} h_j \exp(\beta H_{k,j})}{\left(\sum_{j=1} \exp(\beta H_{k,j})\right)^3} \right) \\ &= \alpha \qquad (9)\end{aligned}$$

We now have a complete build of CBoost@1. Its overview is presented in Fig. 3.

4. EXPERIMENTS

The algorithm is tested on a user comments data set collected from Digg.com, where the best document selection problem is well motivated. Labels are gathered as the community's vote on each comment, and features are gathered from various lexical tests (e.g. word count, SMOG scores) as well as user profiles (account age, friend counts, etc.). Features are then query-level normalized. We also compared our algorithm against previously published baselines on the LETOR3.0 data set. For CBoost@1, we use decision stumps as our weak learners and selected the best parameters as rated against the validation set. For the purpose of comparison, we report NDCG figures instead of U on the Letor3.0 data set, and we report U on the Digg.com data set.

4.1 Datasets

We use the publicly available LETOR datasets [17]. Not all datasets are included due to time and space constraints. A summary of datasets used are as follows:

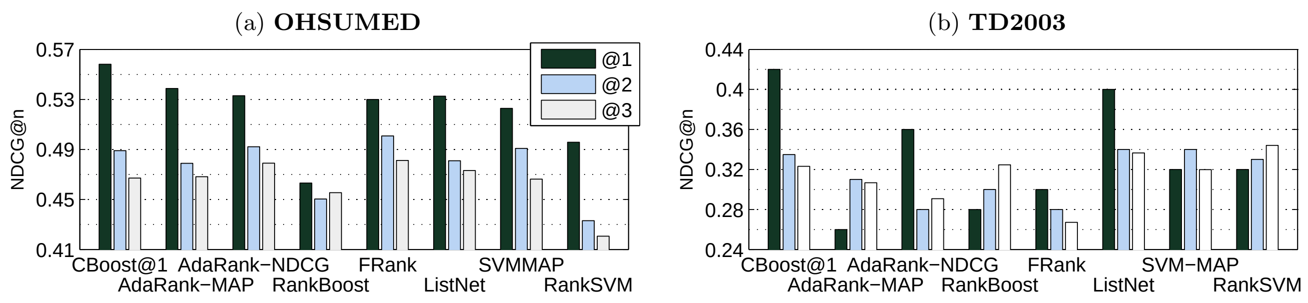
OHSUMED: The OHSUMED dataset is one of the sets included in the LETOR 3.0 package. It contains the relevance labels, pre-computed feature vectors, and content of medical publication titles and abstracts. In total, there are 348,556 documents and 106 queries. Each abstract is labeled as either 'not relevant', 'possibly relevant', and 'definitely relevant'. A mix of low and high level features, such as tf-idf and BM251, make up the 25-dimensional feature vector.

TD2003: We use one of the two topic distillation datasets included in the LETOR package. Contrary to OHSUMED, documents are only labeled as either 'relevant' or 'not relevant' - a binary classification. This makes their use in ranking limited, but still applicable.

TD2003 is arranged into 5 fold and consist of 49,171 documents total across 50 queries. Each document has a total of 44 features ranging from tf-idf to ones proposed in recent papers. It is, therefore, also a smaller set with only 10 queries reserved for testing.

4.2 Baselines

The Letor packages include verified test results for several algorithms. We give a summary of them here.



Parameters we used are; OHSUMED: $\beta = 1$, $\lambda = 0.4$, $T = 140$; TD2003: $\beta = 1$, $\lambda = 0.4$, $T = 100$

Figure 4: Experiment results on benchmark datasets. As desired, our approach (CBoost@1) beats all baselines at NDCG@1. At higher @n truncation points, Cboost@1 is comparable, but hardly superior, to the baselines.

CBoost@1: The algorithm presented in this paper, tuned for NDCG@1.

AdaRank-MAP, NDCG: AdaRank incorporates ranking measures in its optimization. We show results using both MAP (Mean Reciprocal Rank) and NDCG in AdaRank. [8]

RankBoost: A pair-wise boosting algorithm introduced by Freund et al. in 2003. [5]

FRank: FRank uses a loss function called fidelty loss to relate the similarity of two distributions; one generated using predicted pair-wise ordering, the second using the training set. The result is an additive, pair-wise boosting algorithm. [4]

ListNet: Listwise approach by Microsoft that uses a cross-entropy loss function. [3]

RankSVM: RankSVM-Struct is a pairwise approach using Support Vector Machines. [11]

SVMMAP: SVM-MAP is a list-wise support vector machine algorithm for optimizing Mean Average Precision (MAP). MAP is another common ranking measure that is limited to binary labels. For multivariate label sets, all non-zero labels are assumed to be 1. [24]

On both LETOR data sets, we see across-the-board improvements for NDCG@1 with our method. For NDCG@2 and beyond, our algorithm is comparable to the baselines, however, rarely improving upon them. On the Digg.com data set, our approach yields a utility score of 0.393, much better than a tuned svm_rank [10] (0.365).

5. CONCLUSIONS AND FUTURE WORK

This paper studies the problem of best document selection using a machine learning algorithm. Instead of using existing ranking algorithms, we propose a boosting algorithm that optimizes explicitly for the top rank. The strength of this approach is that it greatly simplifies the required objective function and allows us to use a tighter, more accurate approximation than before. We also demonstrate that our algorithm does out-performs a number of baselines on a benchmark ranking data set LETOR3.0 and a new Digg.com data set where best document selection problem is well motivated. As part of future work, we plan on investigating scalable, distributed best document selection algorithms.

6. REFERENCES

[1] Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. Finding high-quality content in social media. In *Proceedings of the international conference on Web search and web data mining*, WSDM '08, pages 183–194, New York, NY, USA, 2008. ACM.

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

[3] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 129–136, New York, NY, USA, 2007. ACM.

[4] Ming feng Tsai, Tie yan Liu, Tao Qin, Hsin hsi Chen, and Wei ying Ma. Frank: A ranking method with fidelity loss. In *Proceedings of the 30th Annual International ACM SIGIR Conference*, 2007.

[5] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, 2003.

[6] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting, 1995.

[7] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of Trust and Distrust, 2004.

[8] Chiao-Fang Hsu, E. Khabiri, and J. Caverlee. Ranking comments on the social web, August 2009.

[9] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20:422–446, October 2002.

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

[11] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 217–226, New York, NY, USA, 2006. ACM.

[12] Pawel Jurczyk. Hits on question answer portals: an exploration of link analysis for author ranking. In *In SIGIR (posters). ACM, 2007. Research Report No*, pages 845–846. ACM Press, 2007.

[13] J. Peter Kincaid, Robert P. Fishburne, Richard L. Rogers, and Brad S. Chissom. Derivation of New Readability Formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy Enlisted Personnel. Technical report, February 1975.

[14] Jon M. Kleinberg. Hubs, authorities, and communities. *ACM Comput. Surv.*, 31, December 1999.

[15] G Harry McLaughlin. Smog grading —– a new readability formula. *Journal Of Reading*, 12(May):639–646, 1969.

[16] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[17] Microsoft Research. Letor: A benchmark collection for research on learning to rank for information retrieval. `http://research.microsoft.com/en-us/um/beijing/projects/letor/`, 2009.

[18] S.E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. pages 109–126, 1996.

[19] Cynthia Rudin and Robert E. Schapire. Margin-based ranking and an equivalence between adaboost and rankboost. *J. Mach. Learn. Res.*, 10:2193–2232, December 2009.

[20] Ruslan Salakhutdinov, Sam Roweis, and Zoubin Ghahramani. On the convergence of bound optimization algorithms. In *in: Proc. 19th Conference in Uncertainty in Artificial Intelligence (UAI '03*, pages 509–516. Morgan Kaufmann, 2003.

[21] Karen Spärck Jones. *A statistical interpretation of term specificity and its application in retrieval*, pages 132–142. Taylor Graham Publishing, London, UK, UK, 1988.

[22] Michael Taylor, Hugo Zaragoza, Nick Craswell, Stephen Robertson, and Chris Burges. Optimisation methods for ranking functions with multiple parameters. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, CIKM '06, pages 585–593, New York, NY, USA, 2006. ACM.

[23] Hamed Valizadegan, Rong Jin, Ruofei Zhang, and Jianchang Mao. Learning to rank by optimizing ndcg measure. Advances in Neural Information Processing Systems, 2009.

[24] Yisong Yue and Thomas Finley. A support vector method for optimizing average precision. In *In Proceedings of SIGIR'07*, pages 271–278. ACM, 2007.

[25] Jun Zhang, Mark S. Ackerman, and Lada Adamic. Expertise networks in online communities: structure and algorithms. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 221–230, New York, NY, USA, 2007. ACM.