

Ranking Mechanisms in Twitter-like Forums

Anish Das Sarma*
Yahoo Research
Santa Clara, CA, USA
anishdas@yahoo-
inc.com

Atish Das Sarma
Georgia Institute of
Technology
Atlanta, GA, USA
atish@cc.gatech.edu

Sreenivas Gollapudi
Microsoft Research
Mountain View, CA, USA
sreenig@microsoft.com

Rina Panigrahy
Microsoft Research
Mountain View, CA, USA
rina@microsoft.com

ABSTRACT

We study the problem of designing a mechanism to rank *items* in forums by making use of the user reviews such as thumb and star ratings. We compare mechanisms where forum users rate individual posts and also mechanisms where the user is asked to perform a pairwise comparison and state which one is better. The main metric used to evaluate a mechanism is the ranking accuracy vs the cost of reviews, where the cost is measured as the average number of reviews used per post. We show that for many reasonable probability models, there is no thumb (or star) based ranking mechanism that can produce approximately accurate rankings with bounded number of reviews per item. On the other hand we provide a review mechanism based on pairwise comparisons which achieves approximate rankings with bounded cost. We have implemented a system, shoutvelocity [5], which is a twitter-like forum but items (i.e., tweets in Twitter) are rated by using comparisons. For each new item the user who posts the item is required to compare two previous entries. This ensures that over a sequence of n posts, we get at least n comparisons requiring one review per item on average. Our mechanism uses this sequence of comparisons to obtain a ranking estimate. It ensures that every item is reviewed at least once and winning entries are reviewed more often to obtain better estimates of top items.

Categories and Subject Descriptors

H.3.3 Information Search and Retrieval [Ranking]

General Terms

Algorithms, Design, Experimentation, Human Factors

*Work done while a student at Stanford University

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WSDM'10, February 4–6, 2010, New York City, New York, USA.
Copyright 2010 ACM 978-1-60558-889-6/10/02 ...\$10.00.

Keywords

Ranking mechanisms, comparisons, thumb-based ranking

1. INTRODUCTION

Ranking has become an important issue not just in web search but also in forums, blogs and social networks such as twitter. While there has been a large body of research on ranking for web search, there is little systematic study of ranking in the aforementioned forums. This paper presents a study of mechanisms for ranking in twitter-like forums. We use *item* as a generic term for posts (in forums), tweets (on twitter), and messages (in social networks).

The popular methods for rankings in forums include “star ratings”, “thumbs up-down ratings”, and “reputation points”. These ranking methods suffer from a few drawbacks. (1) Generally, they aid in the “rich gets richer phenomena” – the items that get rated a few times often end up being displayed on top thereby receiving more impressions and ratings; because of the large volume in most sites, other potentially good items often never receive any ratings. (2) Giving an item a score (thumbs up or down), independent of other items results in unnormalized scores; We shall show that such an independent scoring usually needs a lot of feedback before converging to accurate rankings; and finally, (3) In the absence of any incentives, it is impractical to expect all users to participate in the feedback process; At the same time, absence of any feedback results in user dissatisfaction, as items largely go without any rating. Therefore, appropriate incentive/reward systems coupled with user feedback are key to the efficiency and effectiveness of a rating mechanism.

In this paper, we advocate a comparison-based ranking scheme, where feedback from users are sought in the form of comparisons. Users are shown a pair of items, and they express their opinion of which item they prefer. We theoretically show that such a comparison-based ranking scheme converges to accurate rankings faster than independent ranking schemes described above. Moreover, we show that by using comparison-based ranking, each item needs on average a bounded (constant) feedback bandwidth for our ranking to get very close to the accurate ranking. We have built a system, shoutvelocity¹ (<http://shoutvelocity.com>) [5], that fully implements the comparison-based ranking scheme on

¹Our system is called shoutvelocity where users can “shout” anything they want. When a user ‘shouts’ she is required to compare two previous shouts giving n comparisons over

top of a Web-based forum. Both theoretical and empirical results show that such a system achieves good rankings with very little feedback from users, i.e., for each submitted item, on average we need a constant amount of feedback to ensure a good ranking. We note that, as a consequence, a comparison-based ranking mechanism mitigates drawbacks (1) and (2) mentioned above.

To handle drawback (3), what is really required is a self-correcting method that automatically estimates the score of each item. In our system, we align the incentive mechanism with user satisfaction, i.e., user’s providing feedback are more likely to see the rank of their items faster than other users. Thus, we are able to make optimal use of the “feedback bandwidth” from the users who are inclined to themselves make use of the feedback mechanisms, without burdening other viewers.

The core techniques presented in this paper, and adopted by shoutvelocity, are useful in a number of applications. Obviously, as described above, our techniques are ideally suited for ranking posts in public forums such as digg, twitter, jokes sites, and ranking messages on social networks such as status messages on Facebook. Comparison-based ranking could also be used for ranking blog posts, providing scores and ranking items on online shopping sites such as Amazon and eBay, as well as providing generic or personalized movie recommendations from sites like IMDb. Our algorithm is also appropriate for ranking multimedia such as photos on Flickr², and videos on YouTube (with the caveat that user feedback in the case of videos may take long and be burdensome). Finally, a far-fetched idea is to use comparisons for academic peer reviewing of papers, instead of independent feedback on each paper.

Contributions and Outline

The main contributions of this paper are:

1. Review popular approaches for ranking items through a generic taxonomy (Section 2), and present the considerations that must be taken into account while choosing a particular ranking mechanism (Section 3).
2. In Section 5, we propose a comparison-based ranking mechanism. We analytically study and compare thumbs-based and comparison-based ranking mechanisms, and theoretically show that comparison-based ranking is generally superior.

Preliminaries for our ranking mechanism appear in Section 4, and rigorous proofs for our results from Section 5 appear in Section 7.

3. In Section 6, we present shoutvelocity, a full-fledged forum that implements the comparison-based ranking method. shoutvelocity has been deployed on the Web for over a year, and we provide general statistics and screen shots from our system.
4. In Section 8, we present a detailed experimental evaluation of thumb-based and comparison-based ranking mechanisms based on synthetically generated data with various distributions of item scores as well as real

¹ n shouts. All shouts converge to their merited score/rank very quickly.

²Such ranking has been implemented in practice, as we describe in Section 9

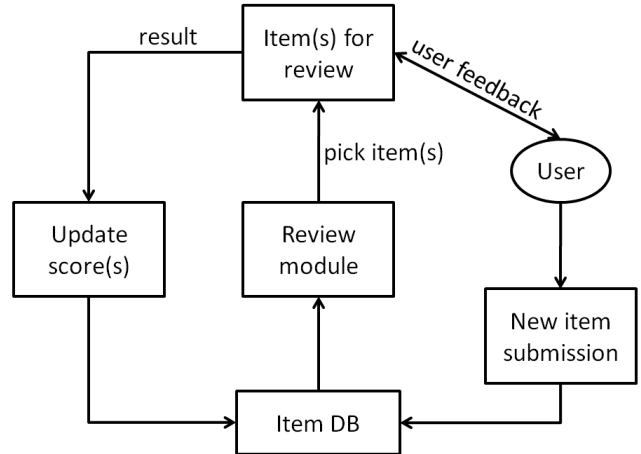


Figure 1: Generic Architecture for Item Ranking System.

data from the shoutvelocity system. We show that comparison-based ranking converges to the right ranking with much less feedback.

Related work is presented in Section 9, and we conclude in Section 10.

2. A TAXONOMY OF APPROACHES

In this section we provide a generic architecture of a system for ranking items and survey possible alternatives for its review module. Suppose we have n items. Our goal is to obtain the best possible *ranking* of the n items with as little *effort* as possible. Intuitively, the amount of effort is captured by the total work done by the reviewers who provide rating feedback.

We assume that any system for ranking items maintains a running *score estimate* for each item. These scores are refined based on user feedback, and the final ranking is given by ordering items in descending order of scores.

Figure 1 presents a generic framework for systems that score items. Items are added to (and maintained in) a database when users submit them. At any time, the *Review Module* picks a set of one or more items, and seeks feedback about them from the user. Based on the feedback provided, the items’ scores are revised and the database is updated. The choice of which items are shown for reviewing may be either implicit or explicit; we say it is explicit if there is a separate ‘rate’ link that when a reader clicks is shown one or more specific item to rate. An implicit system is one where the reader rates as she is browsing through the entries. This is essentially like an explicit system where the reader is asked to rate a specific item that is picked with a probability distribution that depends on the order in which the item entries are displayed; clearly lower items are examined with lower probability.

Obviously, the main challenges here are in determining how the review module works, and how scores are updated based on user feedback.

Broadly, there are two approaches to reviewing items:

1. **Independent Scoring:** Each item is independently

shown to a user, and s/he scores the item based on how much they like the item. The most common such reviewing mechanism is the *thumb-based* scheme, where each user merely gives a thumbs-up or thumbs-down to any item based on whether they like it or not respectively. Thumb-based ranking schemes are widely used on the Web, such as in Digg [1], Twitter [8] and Facebook status message ranking [2], and movie streaming websites such as tv-links [7], and sidereel [6].

A natural generalization of the thumb-based approach is a *star-rating* scheme where each user gives a score (typically between 0 and 5 stars), based on the extent to which they like the item. Star-rating schemes are also popular on the Web such as in movie rating sites like IMDb [3] and Netflix [4].

In this paper, we primarily consider the thumb-based independent scoring mechanism.

2. **Comparison-based Scoring:** A pair of items is shown to a user, and s/he responds by only telling the system which item they find better (irrespective of whether they like/dislike any or both items). While a comparison-based ranking scheme is not as popular as a thumb-based scheme, similar approaches have been used in various different contexts such as ranking photos [13], Elo chess ratings [12], and ranking sports teams. Our system adopts the comparison-based scoring mechanism; in the following we shall exhibit various benefits of it over independent scoring. In the rest of the paper, we study various technical challenges involved with applying comparison-based ranking in a fair and efficient manner.

A further generalization of a comparison-based ranking scheme consists of presenting a user with a set of k items ($k > 2$), and asking the user to provide a ranking among these k items in terms of a partial/total ordering. We do not consider this generalization for most of the paper.

Pairwise comparisons have benefits over thumb-based schemes. First, consider a case where we have n items, a bandwidth of n feedbacks from users. If a user gives a thumbs-up or down on each of the n items, we do not get much information about the relative ranking of the items. On the other hand, by comparing pairs of items, as in any knockout tournament, we are able to get better rankings of the items. Second, to ensure fairness, an item needs to receive comparisons as long as it wins (or as long as there are other items to compare against). However, such a continued thumb-based reviewing may not be practical from the bandwidth point of view. A more rigorous theoretical comparison of the two approaches appears in Section 5.

3. DESIRABLE PROPERTIES

In general, we would like the review module to satisfy certain desirable properties.

1. **Ranking Accuracy:** The system should be able to rank the items as accurately as possible within the review budget. While it may not be critical (or even feasible) to obtain the exact ranking, getting it approximately is desirable. A reasonable goal is to approximate the rank r of an item within a multiplicative

error $r(1 \pm \epsilon)$ with high confidence, for some parameter ϵ . This will ensure that we get the top item right and allows more tolerance for the lower ranked ones.

2. **Review Feedback Bandwidth:** The ranking should converge to the correct one within the desired level of accuracy quickly with a small amount of feedback per item. An important measure here is the average amount of feedback μ_f used per item as the number of items n goes to ∞ in the steady state. μ_f must be bounded as $n \rightarrow \infty$ as otherwise the system is unstable.
3. **Low Latency:** Users should not have to wait long before receiving an estimate on their score/rank. This is distinct from feedback bandwidth as a system can need low bandwidth but have high latency. A specific latency measure that is important is the amount of time the user has to wait after posting an item till s/he receives the first review from the system say in the form of a thumb rating or a result of a pair wise comparison. The latter is important to retain the interest and enthusiasm of the forum writers.
4. **Fairness:** Items should be treated equally with respect to ranking and allocation of review bandwidth. Items that perform equally (thinking of the review system as a game) should be treated equally. Thus, if say in the thumb based rating scheme, two items receive identical ratings in a sequence of reviews, it should not be the case that one is scheduled for more reviews and another is not. Further, preferably, an item should be reviewed till it ‘loses’ at least once; as long as an item is winning it should not be removed from the review system. However, achieving this depends on the available review bandwidth.

We show in Section 5 that for reasonable score distributions, there is no thumb-based algorithm that approximates the rank within a multiplicative error using bounded feedback bandwidth. However, we give a comparison-based algorithm that can achieve a multiplicative error with bounded feedback.

4. PRELIMINARIES

Next we present background necessary for the rest of the paper. Section 4.1 states our assumptions on how scores of items are distributed, and presents a model of how users rate items. Section 4.2 describes how thumb-based and comparison-based algorithms update scores of items that received feedback from users.

4.1 Probability Models

The feedback bandwidth and the ranking accuracy of an algorithm depends on two factors: (1) The distribution of scores of the items; (2) Probability model of how items are rated by the algorithm. Next we describe the models used for these factors.

Score Distribution: We will assume that each item has a score that is given by a real number and the scores come from a certain distribution (say the normal distribution). Let $g(x)$ denote the probability density function of the scores, and $g_c(x)$ denote the cumulative distribution function.

Rating Items: When an item with score x is rated using thumb-based algorithm, it gets a thumbs up with probability $f(x)$ and gets a thumbs down with probability $1 - f(x)$. A typical distribution that has been used for $f(x)$ is the $erf(\alpha x)$ function [12], which is the probability that a normal random variable with mean 0 and variance $\alpha/\sqrt{2}$ exceeds x . Another popular choice is the logistic function $\frac{1}{(1+e^{-\alpha x})}$ [10, 15].

The function $f(x)$ for modeling results of thumb ratings can be generalized to results of pairwise comparisons as follows. We may assume that the probability distribution of the outcome of comparing two items with scores x and y respectively depends on the difference of their scores. We say that the item with score x wins against the one with score y with probability $f(x - y)$. Thus, the thumbs rating probability is given by $f(x - 0)$, which is essentially like comparing an item with score x to an item which score 0 which can be interpreted as a median item.

This is consistent with one of the earliest models for two player games suggested by Arpad Elo [12] for chess. Elo’s central assumption was that the performance of each player is a normally distributed random variable with mean equal to the true score of the player. In any one game the players performance is given by a random number that is normally distributed around his true score. He also made the assumption that all players have the same variance. Observe that the difference between two normally distributed variables with the same variance and means x and y is also a normally distributed variable with mean $x - y$. So this gives that the probability that player with true score x loses to another with true score y is equal to the probability that this variable exceeds $x - y$ which is given by $f(x - y) = erf(\alpha x)$, where α depends on the variance.

4.2 Estimating scores from reviews

Any ranking algorithm makes use of the reviews to produce score estimates for each item. These score estimates are then used to rank the items. The algorithm’s goal is to obtain score estimates such that ranking based on the score estimates is close to the ranking based on the inherent scores of the items. Since the goal of the algorithm is get a near-accurate ranking, the accuracy of the actual score estimates to the actual score is less important than the ranking induced by them.

Thumbs: For thumb-based algorithms, a natural way of estimating the score x of an item is the fraction of reviews in which the item received a thumbs-up rating.

Comparisons: For algorithms based on pairwise-comparisons, a popular choice for revising score estimates is the Elo rating system [12], which was originally invented by Arpad Elo for Chess but is now widely used for many other games.

Consider an item A with score estimate x . Under a given probability model, suppose A was expected to receive E_A points in a comparison, but actually received S_A points. Then, A ’s score is revised to x' based on the following formula: $x' = x + K(S_A - E_A)$. K is a parameter that decays with the number of times the item has been reviewed. Intuitively, K is high initially so as to get close to an items actual score quickly. Based on our probability model, when item A is compared with item B having score estimate y ,

the expected points A receives is $E_A = f(x - y)$. The actual points received by A is $S_A = 1$ if A wins the comparison against B and $S_A = 0$ if A loses the comparison. Originally the Elo system assumed that $f(x)$ is given by the $erf(x)$ function. Later, it was simplified to the logistic function because the resulting update formulae are simpler. Therefore, the final update formula becomes: $x' = x + K(S_A - \frac{1}{1+e^{y-x}})$ (assuming $\alpha = 1$). Intuitively, if an item wins against another item with a high score estimate, it’s score gets a more significant boost than it would by defeating an item with low score estimate.

5. SCHEDULING ITEMS FOR REVIEW

Clearly the main component in any (thumb-based or comparison-based) ranking algorithm is the module that picks the item(s) for review. Even in systems such as Digg and twitter, if the reader rates items while browsing through the site, the reviewing algorithm can be modeled as picking an item for review with a probability depending on its rank in the display order. We will therefore consider explicit systems in our work.

In this section we analyze thumb-based and comparison-based algorithms based on the desirable properties from Section 3. We will look at the ratings bandwidth required by both the thumbs based and the comparison based algorithms. If there are n items with different qualities, we can order them by decreasing quality which results in a ranking of the items. We will compute the number of comparisons/thumbs required to estimate the rank of an item. We are interested in estimating the rank r within a multiplicative error $r(1 \pm \epsilon)$; thus we are interested in identifying the top ranked items with small error in their rank and allow more error as the rank increases. We will compute the average number of ratings required per item with both methods. We will assume that the qualities of the n items are normally distributed.

First, Section 5.1 shows a negative result for thumb-based algorithms, showing that no thumb-based reviewing module can achieve a good ranking with bounded feedback bandwidth. We will show that a comparison-based algorithm can indeed achieve such an approximation with bounded feedback bandwidth.

Fundamentally, comparison-based algorithms allow you to distinguish between items of similar score by directly comparing them against each other, whereas a thumb-based algorithm intuitively compares an item with a median item. Section 5.2 theoretically crystallizes the advantage of being able to compare items with similar score.

Therefore, we focus on comparison-based algorithms for most of the paper. Section 5.3 studies in detail review mechanisms based on pairwise comparisons, which is the model adopted by our system.

5.1 Thumb-based algorithm

We ask the following question: Is there any algorithm for scheduling items for review that can approximate the ranks of items within some multiplicative error with bounded feedback? It turns out that under typical probability models, for thumb-based systems, it can be shown theoretically that there is no such reviewing mechanism. The following theorem formalizes this result. To maintain the flow of the paper, the proof of this theorem and all other results in the paper are deferred to Section 7.

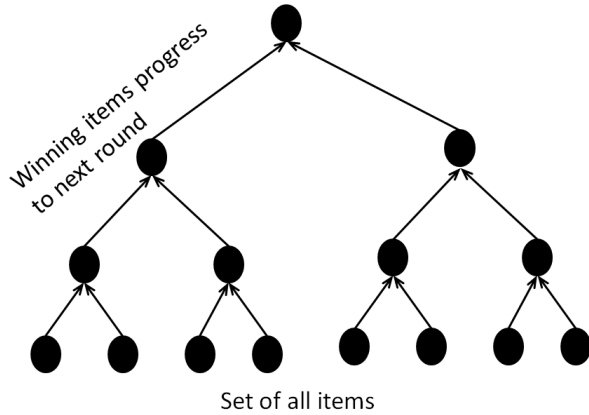


Figure 2: Static tournament to obtain a score estimate for each item.

THEOREM 5.1. *With thumbs rating, if $f = \text{erf}(\sqrt{2}\alpha x)$ and g is the normal distribution with variance 1 and $\alpha > 1$ it is impossible to estimate the ranks r within $r(1 \pm \epsilon)$ with a bounded review bandwidth per item. More generally, if $\int_0^1 \frac{1}{1-f(g_c^{-1}(x))} dx$ diverges and tends to ∞ then there is no such review system based on thumbs, where g_c is the cumulative distribution function corresponding to g , i.e., $g_c(x) = \int_{-\infty}^x g(s) ds$.* \square

In the proof, we show that with thumbs rating if an item is to be rated till it gets a thumbs-down at least once, the average number of ratings μ_f tends to infinity as n tends to infinity when $\alpha > 1$. This means that, for $\alpha > 1$, it is impossible to estimate the ranks r within $r(1 \pm \epsilon)$.

5.2 Comparing items with similar rank

Intuitively, the main advantage of the pairwise comparison over thumbs is that it allows an item to be compared to other items that may be closer in score rather than always comparing it to an average item (assuming thumb rating is like comparing to an average item). This finer comparison allows more accurate estimates of the score. The following theorem, proved in Section 7, confirms the fact that under general conditions, comparing an item with another item of similar score is better than comparing it to an average item.

THEOREM 5.2. *If f is chosen to be the logistic or the erf function, the number of comparisons required to estimate the score of an item decreases as the difference from the score of the item to which it is compared decreases. More generally this is true as long as $\frac{f(x)(1-f(x))}{f'(x)^2}$ is a decreasing function in x (which holds for the logistic and erf functions).* \square

5.3 Comparison-based ranking algorithm

5.3.1 Reviewing Mechanism

Tournament (static case): For simplicity, consider a static collection of n items. In this case it is possible to produce a score estimate and output a unique top candidate in n pairwise comparisons. Each item is compared at most $\log n$

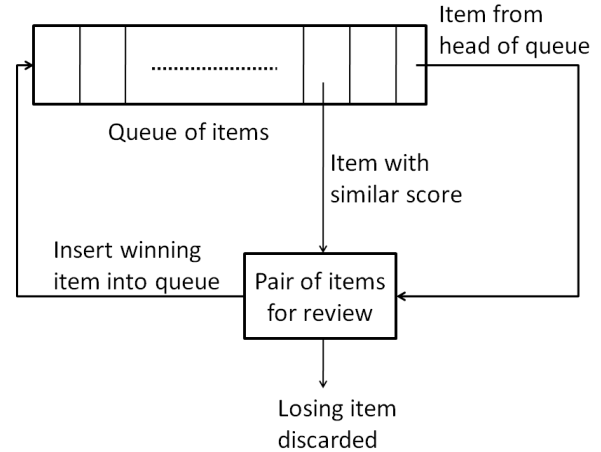


Figure 3: Dynamic tournament using a queue.

times. Each item is compared till it loses, as shown in Figure 2; the height at which it loses gives an estimated score and approximate rank. The average number of comparisons per item is 1.

Simulating Tournament using a queue (dynamic case): In the dynamic case we can simulate the tournament using a queue, as illustrated in Figure 3. An item is enqueued at the tail when it is created. In each comparison we pick an item at the head of the queue and send it to be compared with another item in the queue with the closest score estimate. The losing item is deleted from the queue and the winning item is inserted back into the tail. Thus an item is compared till it loses once. In every review the queue size decreases by one. Thus if the number of reviews is equal or more than the number of items, the queue size is always bounded. Thus average bandwidth used by the system is 1. If the queue size is too small, there will be few items in it which means we will be comparing items with very different score estimates; in this case we may compare the item at the head with another item with similar score outside the queue from the item database. Whenever the queue size exceeds a certain threshold we revert back to comparing only with items from the queue.

The above mechanism utilizes a feedback bandwidth of 1 per item. If more bandwidth is available then there will be times when there is at most one item in the queue. Then we can use it to improve the ranking among the top candidate items. Again one can pick an item randomly biased by the current (possibly time discounted) score estimates, and send it for comparison with another item with near by score. For instance, we may rank all items by the time discounted score and pick an item with rank r with probability proportional to $1/r^\gamma$, where γ is a parameter, and compare it with the next item in the ranking. If $\gamma = 0$, we are sampling uniformly; if $\gamma > 0$ we are sampling biased towards the items with higher score. Thus γ should be chosen in the range $[0, 1]$ and for $\gamma > 1$ the bias is large and most of the probability is concentrated in the top few ranks. Later in the simulations we will see how the convergence of the score estimates depends on γ for the standard f and g functions.

Recall, when we get feedback on a pair of items, the up-

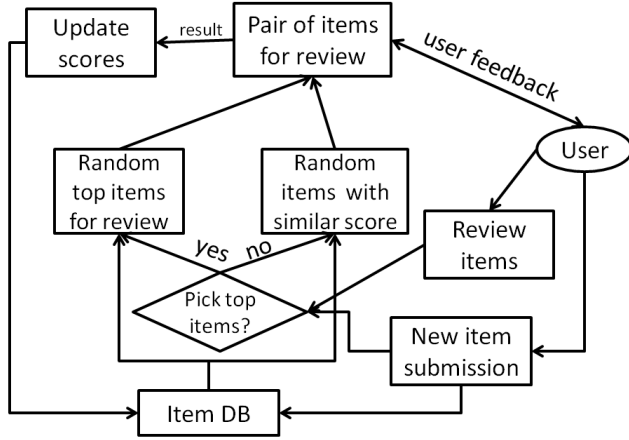


Figure 4: Architecture for Shoutvelocity’s Ranking System based on Pairwise Comparisons.

date process lowers the score of the losing item, and increases the score of the winning item based on Section 4.2.

5.3.2 Theoretical Guarantee

We are unable to analyze the above tournament algorithms theoretically. But we prove that a simplified version of the algorithm is able to estimate rank r within error ϵr with bounded number of comparisons per item for the specific distributions we study. The formal proof the following theorem appears in Section 7.

THEOREM 5.3. *With pairwise comparisons, if $f = \text{erf}(\alpha x)$ and g is the normal distribution there is an algorithm to estimate the rank within multiplicative error $1 \pm \epsilon$ with bounded feedback bandwidth per item.* \square

The algorithm that achieves the result above compares items with other items with successively increasing scores till it consistently loses: Consider items with geometrically decreasing percentile ranks $1/2, \rho/2, \rho^2/2, \dots, 1/n$. We compare a particular item to these items successively till we find two consecutive ranks between which the item lies. The number of comparisons with each of the above items is large enough to conclude whether we should proceed to the next item with high confidence. We clearly approximate the rank within a factor of $1 + \epsilon$ if $\rho = \frac{1}{(1+\epsilon)}$.

6. SHOUTVELOCITY SYSTEM

6.1 System Architecture

We elaborate on the comparison-based reviewing module adopted by our shoutvelocity system. Figure 4 shows the various components of our system, essentially obtained by expanding the reviewing module from Figure 1. In our system, when a user submits an item, he gets shown a pair of items for review. Therefore, for every item he submits, we get feedback on two items. Obviously, even without submitting an item, the user is given the option of reviewing any number of pairs of items.

In general, if we wish to obtain better rank estimates for the top items, items with higher score estimates should be



Figure 5: Screenshot of top shouts from shoutvelocity.com.

given more number of reviews. One method is to pick items randomly biased by the current score or rank estimates of the items. From a practical standpoint, another factor that needs to be taken into account is the age of the item. Clearly newer items should be preferred over older items. A simple way to achieve this is to time discount the score estimates.

6.2 Screenshots and Statistics

The shoutvelocity website was launched in June 2008. In the interest of confined testing of various UI and ranking approaches, we only made a limited release of the site. While the website remains open to public, we explicitly invited only a small number of people, and we’ve attracted 196 users since our release. Our website has seen around 1245 distinct shouts posted, and together these shouts have received 4853 reviews in all.

We give screenshots to illustrate two of the core features of shoutvelocity. Figure 5 shows the shouts that were rated among the best by users. We give a score for each shout, as well as links to see the “shouter”, comments left by users, and history of other shouts against which it won/lost. On top of Figure 5 we see the various ranking options supported by shoutvelocity, which include ranking purely based on score, filtered by recency, or weighted based on score decay with age.

Figure 6 shows an example of a review screen, where two shouts are shown. The user is asked to pick which shout she prefers, and may additionally leave comments/emoticons on either shout.

We reiterate that the above figures only show a small sampling of the various features supported by shoutvelocity. Since the screenshots, and even this paper covers only the core technical issues with shoutvelocity, we encourage the readers to visit <http://shoutvelocity.com> to enjoy the capabilities of our complete system.

7. PROOFS

Proof of Theorem 5.1: In order to ensure that every item is rated till it loses, we show that the average number of ratings μ_f required tends to infinity as $n \rightarrow \infty$, for $\alpha > 1$.

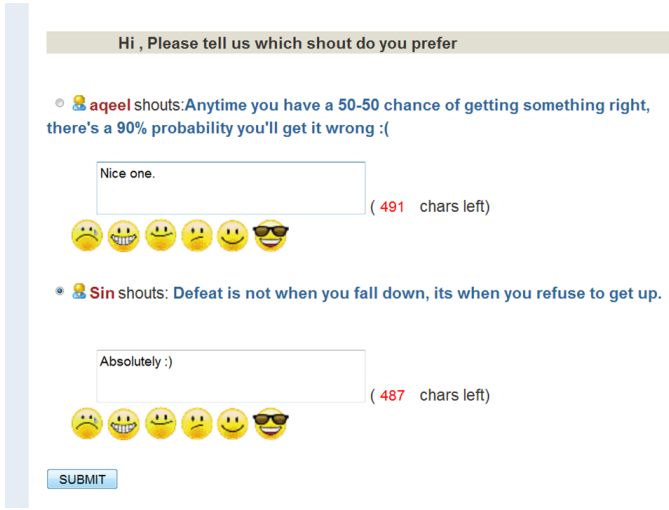


Figure 6: Screenshot of review screen from shoutvelocity.com.

If a mechanism does not rate an item till it loses at least once, then this item cannot be distinguished from better items. Therefore, it is impossible to estimate the ranks r within $r(1 \pm \epsilon)$.

Let $\bar{g}_c(x)$ denote $1 - g_c(x)$, and let $\bar{f}(x) = 1 - f(x)$. For normally distributed qualities $\bar{g}_c(x) = \text{erfc}(x)$. Let us estimate the score of the item with rank r ; at that point the cdf must have value $1 - \frac{r}{n}$. So the quality v_r is concentrated around $\bar{g}_c^{-1}(r/n)$. For large n we may simply write $v_r = \bar{g}_c^{-1}(r/n)$. Let $s = r/n$; then we can think of $v_s = \bar{g}_c^{-1}(s)$ as a function of s .

Let $x = v_s$. Number of thumb ratings required for an item with score x till it loses at least once is $\frac{1}{\bar{f}(x)}$, as in each rating it loses with probability $\bar{f}(x)$. Now $\bar{g}_c(x) = s$. For large x , $\bar{g}_c(x) = \text{erfc}(x)$ can be approximated as $g'(x)/x$. Now

$$\frac{1}{\bar{f}(x)} \approx \frac{x}{\bar{f}'(x)} = \frac{x}{(xs)^{\alpha^2}} \geq \frac{1}{x^{\alpha^2}} \frac{1}{(s)^{\alpha^2}} \geq \frac{1}{(\log n)^{\alpha^2/2}} \frac{1}{(s)^{\alpha^2}}$$

So total number of thumbs required is at least:

$$\sum_{r \in 1..n} \frac{1}{(\log n)^{\alpha^2/2}} \frac{1}{(r/n)^{\alpha^2}}.$$

So the average number of thumbs per item is:

$$\begin{aligned} & \frac{1}{(\log n)^{\alpha^2/2}} \sum_{r \in 1..n} \frac{1}{r^{\alpha^2}} \frac{1}{n} \\ & \approx \frac{1}{(\log n)^{\alpha^2/2}} \int_{s=\frac{1}{n}}^1 \frac{1}{s^{\alpha^2}} ds \\ & = \frac{1}{(\log n)^{\alpha^2/2}} O(n^{\alpha^2-1}) \end{aligned}$$

For $\alpha > 1$, clearly this quantity diverges to ∞ . More generally we get

$$\mu_f = \frac{1}{n} \sum_{r \in 1..n} 1/\bar{f}(\bar{g}_c^{-1}(r/n)) \approx \int_0^1 \frac{1}{\bar{f}(\bar{g}_c^{-1}(s))} ds$$

$$= \int_0^1 \frac{1}{1 - f(g_c^{-1}(1-s))} ds = \int_0^1 \frac{1}{1 - f(g_c^{-1}(s))} ds$$

□

Proof of Theorem 5.2: Let $p = f(x)$ denote the probability of winning in one comparison; the variance of the indicator variable for one comparison is $p(1-p)$.

By central limit theorem (or by similar concentration inequalities such as Hoeffding bounds), for a large number of comparisons k , the fraction of wins is expected to be around p with variance $\frac{p(1-p)}{k}$. The probability that the estimate for p has additive error more than ϵ is $e^{-\Theta(\frac{k\epsilon}{p(1-p)})}$. For confidence δ this gives:

$$k = \Theta\left(\frac{f(x)(1-f(x))}{\epsilon^2}\right) \log(1/\delta)$$

However, we need to estimate x within error $x \pm \epsilon x$, which means we need to estimate $p = f(x)$ within error $|f(x \pm \epsilon x) - f(x)| \approx |\epsilon f'(x)|$. Replacing ϵ by $\epsilon f'(x)$ gives us the desired expression: $\frac{f(x)(1-f(x))}{\epsilon^2 f'(x)^2} \log(1/\delta)$.

Observe that for $f(x)$ equal to the erf or the logistic function the quantity $\frac{f(x)(1-f(x))}{f'(x)^2}$ is decreasing in x . This means that the number of comparisons required is fewer when compared to an item with similar quality. □

Proof of Theorem 5.3: We show that successively comparing the given item with geometrically decreasing percentile ranks $1/2, \rho/2, \rho^2/2, \dots, 1/n$ achieves the result. We compare the item to these items successively till we find two consecutive ranks between which the item lies. Choosing $\rho = 1/(1+\epsilon)$ ensures that the algorithm approximates the rank within a factor of $1+\epsilon$. Next we bound the total number of comparisons required.

An item with percentile rank $s = \frac{r}{n}$ gets compared with items with percentile ranks $s(1+\epsilon), s(1+\epsilon)^2, s(1+\epsilon)^3, \dots, 1/2$. Thus it gets compared with $\log(1/s)/\log(1+\epsilon)$ items.

To get a good estimate the number of comparisons required is minimized when it is compared to an item with closest quality which is $s(1+\epsilon)$.

To estimate the rank within error $r\epsilon$, we need to estimate its quality v_r within error $\Delta = |v_{r \pm \epsilon r} - v_r| = |v_{s \pm \epsilon s} - v_s| \approx v'(s)\epsilon s$. Let $x = v_s$. Then $v'(s) = (\bar{g}_c^{-1})'(s) = 1/\bar{g}_c'(\bar{g}_c^{-1}(s)) = 1/\bar{g}_c'(x)$. Thus error in quality estimate is $\Delta = \epsilon s/g'(x)$. Since Δ is also the difference in the qualities of the two items, number of comparisons required is $\frac{f(\Delta)(1-f(\Delta))}{(\epsilon \Delta f'(\Delta))^2}$.

Now $\Delta = \epsilon s/\bar{g}_c'(x) = \epsilon s/(x\bar{g}_c(x)) = \epsilon/x = \Theta(\epsilon/\sqrt{\log(1/s)})$. Since Δ is small $f(\Delta)$, $1-f(\Delta)$ and $f'(\Delta)$ are constant, so number of comparisons is $O(1/\Delta^2) = O(\log(1/s))$. This is multiplied by at most $\log(1/s)/\log(1+\epsilon)$. Now total number of comparisons is at most $O(\sum_{x:1..n} \log^2(n/x)/\log(1+\epsilon)) = O(n/\log(1+\epsilon))$.

Thus, the average number of comparisons required is $O(1/\log(1+\epsilon))$ which is a constant. □

8. EXPERIMENTAL RESULTS

8.1 Simulations over Synthetic Data

We perform experiments to compare the ranking schemes of the thumbs algorithm and the comparison based shoutvelocity algorithm. Experiments are done by considering n players (set to 1000) and choosing their true scores based on $g(x)$ being the normal distribution with variance 1. We

perform m pairwise-comparisons or $2m$ thumb evaluations, where m is varied from 1 to kn .

Thumb-based Approach: In each iteration, in case of the thumbs algorithm, we pick a random item biased towards those with higher score estimates. Specifically, we rank all items with score estimates, and we pick the r th item with probability proportional to $\frac{1}{r^\gamma}$. In each thumb review, the item with score x wins with probability $f(x) = \text{erf}(\sqrt{2}\alpha x)$.

Comparison-based Approach: In each iteration, we first pick one item, based on rank estimates above: The r th item is picked with probability proportional to $\frac{1}{r^\gamma}$. Suppose the r th item is picked, the second item is the $r + 1$ th ranked one. Let the scores of these items be x and y respectively. In each comparison, the item with score x wins with probability $f(x - y)$.

The algorithm for updating scores also keeps track of the number of times every item has been selected for comparison. This is a discounting factor to capture the intuition that after a lot of comparisons, the scores start to *stabilize*, and so one comparison can alter the score only marginally. Suppose the items i_1 and i_2 have been evaluated k_1 and k_2 times respectively. The discounting is done based on $c_i = \frac{2}{(1+k_i)^{0.5}}$ where $i = 1, 2$. Further, let their current score estimates be s_1 and s_2 . If i_1 gets voted in the comparison, then s_1 is incremented by $c_1 * (1 - \frac{1}{1+e^{s_2-s_1}})$ and s_2 is decremented by $c_2 * \frac{1}{1+e^{s_1-s_2}}$. Similarly, the update is done in case i_2 wins.

Evaluation Metric: We compute the MRR (mean reciprocal rank) for the item with highest score in the rank produced by the two approaches above. The MRR is the mean of the quantity $\frac{1}{R}$, where R is the rank of the actual top item in the ranking produced by the approach above. Clearly, MRR is at most 1 and an MRR of 1 means the top item was always correctly identified as the best item. Our experiments are performed over several runs by varying α , γ , and m .

Comparison based ranking algorithm consistently outperforms the thumbs based algorithm for $\alpha = 1.0, 1.5$. Further, the rate at which the comparison based algorithms improve as the number of evaluations is increased is also higher. A similar behavior is observed with increasing number of players; the comparison based algorithm seems more resistant to a large number of players even when the evaluation is done on just the top few in the produced rankings. See Figures 7, 8, and 9 for the results for $\alpha = 1.5$, $\alpha = 1.0$, and $\alpha = 0.5$ respectively.

8.2 Shoutvelocity system

Our system has the history of all 4853 pairwise comparisons performed by users, over a set of 1245 items. We study the convergence of MRR for as the number of reviews increases. Since we do not have the “true” scores of each item, we take many random orderings of the collection of 4853 reviews, replay in that order using the ELO rating system. The score for each item, averaged over these random orderings, is taken to be the true score of the item.

As in the simulations above, we measure how MRR increases with the number of reviews. Figure 10 shows how MRR increases with the number of comparisons.

Next we provide latency of reviewing from the shoutvelocity system. Figure 11 plots for each latency x , what fraction

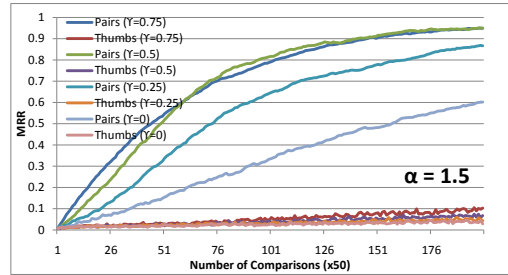


Figure 7: MRRs for thumb-based and comparison-based ranking for $\alpha = 1.5$. At $m = 1000$, for the best γ , MRR is 0.244 for comparison-based approach and 0.019 for thumbs.

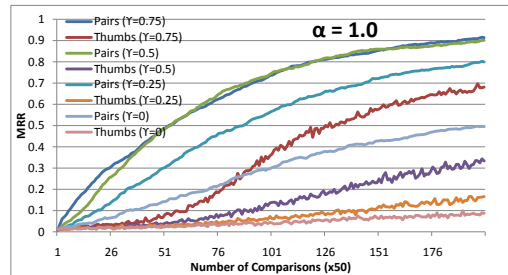


Figure 8: MRRs for thumb-based and comparison-based ranking for $\alpha = 1.0$. At $m = 1000$, for the best γ , MRR is 0.149 for comparison-based approach and 0.0186 for thumbs.

of items waited for x reviews to get their first feedback. We can see that all items got their first feedback in 20 reviews, and most had to wait for less than 5 reviews.

Finally, in Figure 12 we show the cumulative distribution of scores for shouts from the shoutvelocity system. (In our system, for displaying scores, we’ve added a score of 5 to every shout, so that they are nonnegative.) The scores roughly display a normal distribution, confirming Elo’s assumption of shouts’ performance being based on a normal distribution.

9. RELATED WORK

Ranking performs two central roles in today’s social networks - a) surface the most relevant result to the user query, and b) align the incentives with the goal of seeking relevant user generated content. There has been considerable amount of work in analyzing simple voting schemes like thumbs-up/down to comparison based voting schemes [13]. In the former, the user gets to see one result and gets to vote up or down as a choice suggesting her preference while in the latter, the user compares more than one result (usually two) and ranks them according to her preferences. The Bradley-Terry-Luce (BTL) model (Bradley & Terry [10] and Luce [15]) is often applied to pairwise comparison data to

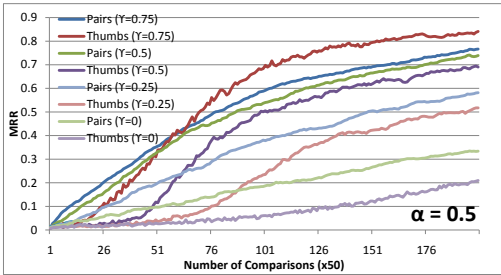


Figure 9: MRRs for thumb-based and comparison-based ranking for $\alpha = 0.5$. At $m = 1000$, for the best γ , MRR is 0.17 for comparison-based approach and 0.06 for thumbs.

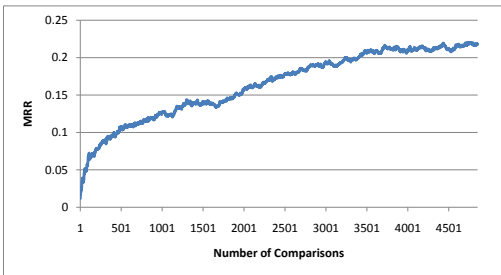


Figure 10: Convergence of MRR on real data from shoutvelocity.

scale preferences. For an excellent exposition on pairwise comparisons, the reader is referred to [11]. Thomas Saaty motivates the need for pairwise comparison in order to rank entities that have no tangible properties with known scales of measurement [16].

Hacker and Von Ahn [13] propose a 2-person online game, where users express preferences on photos, and these preferences are used for ranking. They compare the performance of their scoring function with other well-known functions such as ELO [12] and TrueSkill [14]. The Elo rating system [12] is used to compute the relative skill of players in two-player games such as chess. Unlike [13], our focus is to design a comparison-based mechanism taking into account feedback bandwidth consideration, where each person submitting an item reviews only one more pair of items. We borrow the score update function from the Elo system.

Ajtai et al [9] study the problems of selection and ranking with imprecise comparisons. Again, the idea behind their work is similar in spirit to what we suggest in this work: for every user there exists a value $\delta > 0$ which differentiates between a *just noticeable difference* and otherwise, i.e., if the values of the two elements being compared is less than δ , the result of the comparison could go either way. When the value of the difference is more than δ , then the comparison is correct.

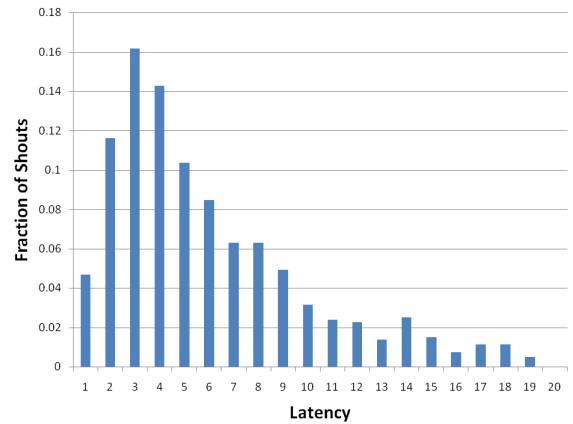


Figure 11: Latency of shouts plotted against fraction of shouts.

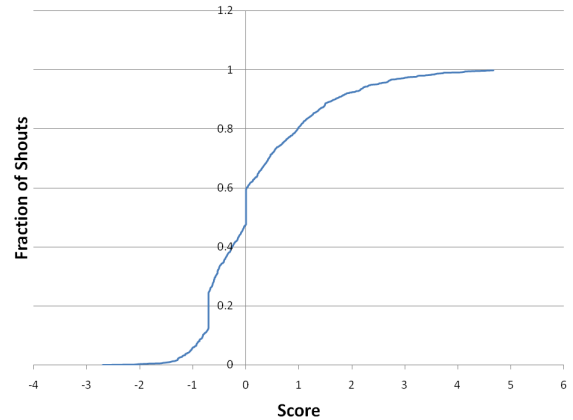


Figure 12: Cumulative distribution of scores from shoutvelocity.

10. CONCLUSIONS

This paper addressed the problem of designing ranking mechanisms for forums. Broadly, we studied independent thumb-based and comparison-based reviewing of items in the forum. We theoretically showed the benefits of comparison based ranking mechanisms based on desirable properties including accuracy and rapid convergence with minimal user feedback. We presented shoutvelocity, an online forum that fully implements the comparison-based ranking mechanism from this paper. We experimented with synthetically generated data as well as real data from shoutvelocity, and showed that shoutvelocity’s comparison-based ranking significantly outperforms thumb-based ranking on the desired properties.

Star-ratings are a clear generalization of thumb-based ratings, and our theoretical analysis can be easily extended to star-ratings: Intuitively, a rating of 3 stars out of 5 is roughly like getting 3 thumbs-ups and 2 thumbs-downs. Analogously, pairwise comparisons can be generalized to n-way comparisons, but the theory doesn’t directly carry over. Although the practicality of n-way comparisons is questionable as they may impose a more significant reviewing burden on users, their theoretical study would be interesting.

11. REFERENCES

- [1] Digg. <http://digg.com>.
- [2] Facebook. <http://www.facebook.com>.

- [3] IMDb: The Internet Movie Database. <http://www.imdb.com>.
- [4] Netflix. <http://www.netflix.com>.
- [5] Shoutvelocity. <http://shoutvelocity.com>.
- [6] Sidereel. <http://www.sidereel.com>.
- [7] TV Links. <http://tolinks.cc>.
- [8] twitter. <http://twitter.com>.
- [9] Miklós Ajtai, Vitaly Feldman, Avinatan Hassidim, and Jelani Nelson. Sorting and selection with imprecise comparisons. In *Proc. of ICALP*, 2009.
- [10] R. Bradley and M. Terry. The rank analysis of incomplete block designs: I. the method of paired comparisons. In *Biometrics*, 1952.
- [11] H. . David. *The Method of Paired Comparisons*. New York: Oxford University Press, 1988.
- [12] Arpad Elo. *The Rating of Chessplayers, Past and Present*. Arco Publications, 1978.
- [13] Severin Hacker and Luis von Ahn. Matchin: eliciting user preferences with an online game. In *Proc. of CHI*, 2009.
- [14] R. Herbrich, T. Minka, and T. Graepel. Trueskilltm: A bayesian skill rating system.
- [15] R. D. Luce. *Individual Choice Behavior: A Theoretical Analysis*. New York: Wiley, 1959.
- [16] T. L. Saaty. Rank from comparisons and from ratings in the analytic hierarchy/network processes. *European Journal of Operational Research*, 168(2), 2006.