Models of Web Page Reputation in Social Search

Kevin McNally, Michael P. O'Mahony and Barry Smyth
CLARITY Centre for Web Sensor Technologies,
School of Computer Science and Informatics,
University College Dublin
Email: {firstname.lastname}@ucd.ie

***Abstract*—To date web search has been a solitary experience for the end-user, despite the fact that recent studies highlight the potential for collaboration that is inherent in many search tasks and scenarios. As a result, researchers have begun to explore the potential for a more collaborative approach to web search, one in which the search actions of other users can influence the results returned. In this context, the expertise of other users plays an important role when it comes to ensuring the quality of recommendations that arise from their actions. The *reputation* of these users is important in collaborative and social search tasks, much as *relevance* is vital in conventional web search. In this paper we examine this concept of reputation in collaborative and social search contexts. We describe a number of different reputation models and evaluate them in the context of a particular social search service. Our results highlight the potential for reputation to improve the quality of recommendations that arise from the activities of other searchers.**

I. Introduction

There is little doubt the impact that search engines have had on the internet and on the lives of internet users. Most of us routinely turn to our favourite search engine when we want to locate information online. The success of web search has come about as a result of some very significant innovation that dates back to the early days of web search in the late 1990s. At the time, the first generation of web search engines (e.g. Altavista, Excite, Lycos etc.) relied on an information retrieval type approach to search, selecting and ranking results based largely on how well they matched the terms in the current search query. It quickly became apparent, however, that such an approach would not scale in the world of the web. While the presence of query terms in a result page might signal potential relevance, this type of matching did not provide a strong enough ranking signal and consequently search engine result lists performed poorly in terms of their overall precision. The significant breakthrough that led to modern web search engines came about through the work of Brin and Page [1], and Kleinberg [2], highlighting the importance of link connectivity when it came to understanding the importance of web pages. In the end, ranking metrics based on this type of connectivity data came to provide a key signal for all of today's mainstream search engines.

Today's search engines still rely on link connectivity, term overlaps, and other relevance signals. In the meantime however, the so-called *social web* has provided users with new types of information discovery tools. Many users frequently find that the content they are interested in is shared via their social graph; for example, page sharing on Twitter and Facebook is now commonplace and many users rely on their social feeds as a source of daily content. Consequently, there is now considerable interest in the concept of *social search*. If our social graphs are a valuable source of content then why not harness our social networks to improve mainstream search results? As a result, we have seen major search engines like Bing and Google introduce Twitter and Facebook feeds into the result pages and both search engines now provide users with an opportunity to effectively vote on pages (through Facebook "likes" and Google +1) so that this social signal can be used during result ranking. In this paper we focus on the HeyStaks system [3]. HeyStaks has been developed to add a layer of social search onto mainstream search engines, using recommendation techniques to automatically suggest results to users based on pages that members of their social graphs have found to be interesting for similar queries in the past. HeyStaks adds collaboration to conventional web search and allows us to benefit from the past searches of people we trust on topics that matter to us. Previous work has focused on the HeyStaks recommendation engine and on a novel model of *user reputation* based on instances of collaboration between users [4].

In this paper we describe in detail how user reputation can be used during the result recommendation process. While earlier work evaluates different types of user reputation models, in this work we describe how reputation from multiple users can be combined and aggregated to model *page reputation*, which can provide evidence for result quality and relevance at recommendation time. Specifically, we describe and evaluate 5 different reputation aggregation techniques and evaluate their impact on the quality of recommendations based on the results of a live-user trial. The remainder of this paper is structured as follows. In the next section we outline related work in the area of web search and recommender systems. In sections III and IV we provide a brief overview of the HeyStaks system and its user reputation model, which will serve as the basis for this work. In section V we describe a number of different page reputation models and show how they can be used to influence recommendations at search time. Finally, before concluding, in section VI we describe the results of a comparative study of these strategies, based on live-user data.

II. Related Work

Recently, research efforts have been focused on identifying new signals that can be harnessed to assist users to locate

relevant search results and product and service recommendations on the web. Here we briefly review some related work in this regard, examining some of the relevance models that drive modern web search engines and recommender systems. Further we describe recent work in the area of reputation systems and how reputation can be used to further enhance the quality of search results and recommendations that are delivered to end users.

One of the key differentiators between web search engines and traditional information retrieval approaches is that the underlying link structure of the web can be leveraged by search engines as a additional source of relevance and ranking information to complement query term matching techniques. Two of the best known examples of web search algorithms that utilise link structure are Google's PageRank algorithm [1] and the HITS algorithm developed by Kleinberg [2]. The PageRank algorithm models pages on the web as vertices in a directed graph with the hyperlinks between pages representing the edge set. The relative importance of a page is modeled by the number of inlinks to the page, which can be seen as a kind of recommendation from the wider community. PageRank is a recursive algorithm, where the ranks of pages are a function of the ranks of those pages that link to them, with pages that are linked to by many other important pages receiving higher ranks themselves. The HITS algorithm also utilizes link structure to rank web pages. In contrast to PageRank, HITS computes an *authority* and a *hub* score for each page, which measure the value of a page's content and the value of its links to other pages, respectively. There is no doubting the value and success of mining the link structure of the web as a page ranking signal; a recent comScore press release indicates that Google Sites achieved 65.4% market share in the U.S. explicit core search market[1].

In addition to conventional query-based search services, recommender systems have also been widely employed on the web to help people discover information, products and services that are relevant to their personal needs. In the literature, two main approaches to recommender systems have been described, known as *content-based* [5] and *collaborative-filtering* based [6] approaches. In content-based systems, items are recommended to users that are similar to items that have been liked in the past. Comparisons between items are calculated over the features that are associated with each item (for example, movies can be described by meta-data such as genre, director, cast etc.). Content-based approaches have been used in a variety of recommendation applications, including TV, e-commerce and travel recommenders [7], [8]. On the other hand, collaborative recommenders help users to make choices based on the preferences of other users in a system. The basic heuristic employed is that users who agreed or disagreed on items in the past are likely to agree or disagree on future items. To make recommendations, collaborative filtering algorithms typically find the most like-minded users in a system based on the similarity of their preference data, and weight and combine the preferences of those users. A key advantage of collaborative recommenders is their ability to capture relationships between users based on item preference information alone, and without the need for any rich meta-data about the item being recommended. In addition, collaborative and content-based techniques can be combined to form hybrid recommenders as described in [9].

Recently there has been considerable interest in *reputation systems* as an additional signal to enhance the quality and robustness of recommendations made to users. The work of O'Donovan and Smyth [10] addresses reputation in the context of collaborative recommender systems. In this case, a standard collaborative filtering algorithm is modified to add a user-user trust score to compliment the normal profile or item-based similarity score, so that recommendation partners are chosen from those users that are not only similar to the target user, but who have also had a positive recommendation history with that user. It is posited that reputation can be estimated by measuring the accuracy of a profile at making predictions over time. Using this metric average prediction error is improved by 22%. Similar to O'Donovan and Smyth, Massa and Avesani [11] propose a reputation algorithm called *MoleTrust* that can be used to augment an existing collaborative filtering system. The mechanism calculates a "trust metric" similar to item-based similarity, which propagates across a network of content producers. This algorithm can be tuned to propagate over a specific depth across a social graph, meaning reputable users only have influence over a set of users of a known size. The authors find that *MoleTrust* can improve the accuracy of predictions made by a recommender system, even in cases where users have provided few item ratings.

In this paper we show how reputation can be leveraged to influence the ranking of result page recommendations made by the HeyStaks social search service. In particular, we propose a number of models to estimate page reputation based on the reputation of those HeyStaks users who have previously interacted with the page. As with the related work discussed above, our findings indicate that the signal afforded by our reputation model leads to enhanced recommendation quality when combined with the HeyStaks page relevance model as described in the next section.

III. A Review of HeyStaks

HeyStaks is an approach to collaborative web search that is designed to work with mainstream search engines such as Google, Bing, and Yahoo; so users search as normal, on their favourite search engines, but benefit from search recommendations from people they trust. The HeyStaks system has been described previously in [3], where the focus was on a description of its recommendation technique. The aim of this paper is to investigate the role of a novel reputation model during recommendation, whereby the search reputation of users is allowed to influence recommendation directly. We will return to the issue of reputation in following sections, but

[1]http://www.comscore.com/Press_Events/Press_Releases/2011/5/comScore_Releases_April_2011_U.S._Search_Engine_Rankings. Accessed June 19th, 2011

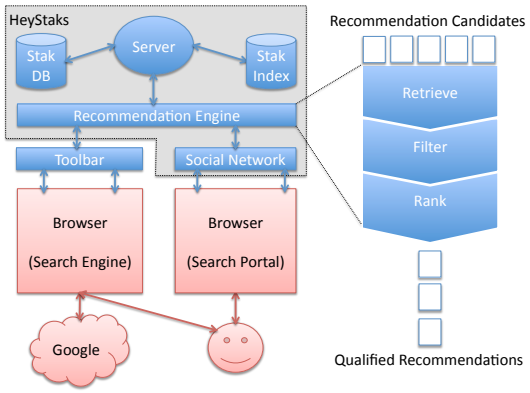


Fig. 1. The HeyStaks system architecture and outline recommendation model.

first we present a brief review of HeyStaks in order to provide sufficient technical context for the remainder of this paper.

A. System Architecture

Figure 1 presents the HeyStaks architecture. There are two key components: a client-side *browser toolbar/plugin* and a back-end *server*. The toolbar serves a dual purpose: It provides users with direct access to the HeyStaks functionality, allowing them to create and share staks, tag or vote for pages etc. It also provides for the type of deep integration with mainstream search engines that HeyStaks requires. For example, the toolbar captures the routine search activities of the user (query submissions and result click-thrus) and it also makes it possible for HeyStaks to augment the mainstream search engine interface so that, for example, HeyStaks' recommendations can be integrated directly into a search engine's results page. The toolbar also manages the communication with the back-end HeyStaks server. Search activities (queries, click-thrus, tags, votes, shares etc.) are used by the server to update the HeyStaks stak indexes. And these stak indexes provide the primary source of recommendations so that when a user submits a query to a mainstream search engine, in a given stak context, this query is fed to the HeyStaks server in order to generate a set of recommendations based on the target stak and, possibly, other staks that the user has joined.

B. The Recommendation Engine

Each stak in HeyStaks captures the search activities of its stak members. The basic unit of stak information is a result (URL) and each stak (S) is associated with a set of results, $S = \{r_1, ..., r_k\}$. Each result is also anonymously associated with a number of implicit and explicit interest indicators, based on the type of actions that users can perform on these pages, which include:

- *Selections (or Click-thrus)* – a user selects a search result (whether *organic* or *recommended*);
- *Voting* – a user positively votes on a given search result or the current web page;
- *Sharing* – a user chooses to share a specific search result or web page with another user (via email or by posting to their Facebook Wall etc.);
- *Tagging/Commenting* – the user chooses to tag and/or comment on a particular result or web page.

Each of these actions can be associated with a degree of confidence that the user finds the page to be relevant for a given query. Each result page r_i^S from stak S, is associated with these indicators of relevance, including the total number of times a result has been selected (Sl), the query terms $(q_1, ..., q_n)$ that led to its selection, the terms contained in the snippet of the selected result $(s_1, ..., s_k)$, the number of times a result has been tagged (Tg), the terms used to tag it $(t_1, ..., t_m)$, the votes it has received (v^+, v^-), and the number of people it has been shared with (Sh) as indicated by Equation 1.

$$r_i^S = \{q_1...q_n, s_1...s_k, t_1...t_m, v^+, v^-, Sl, Tg, Sh\} \; . \quad (1)$$

Importantly, this means each result page is associated with a set of *term data* (query terms and/or tag terms) and a set of *usage data* (the selection, tag, share, and voting count). The term data is represented as a Lucene (http://lucene.apache.org) index, with each result indexed under its associated query and tag terms, and this provides the basis for retrieving and ranking *recommendation candidates*. The usage data provides an additional source of evidence that can be used to filter results and to generate a final set of recommendations.

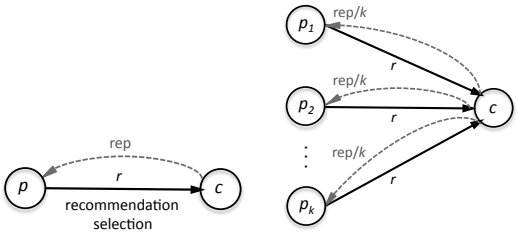
At search time, the searcher's query q_T and current stak S_T are used to generate a list of recommendations to be returned to the searcher. For the purpose of this paper we will discuss recommendation generation from the current stak S_T only, although in practice recommendations may also come from other staks that the user has joined or created.

There are two key steps when it comes to generating recommendations. First, a set of *recommendation candidates* are retrieved from S_T by querying the relevant Lucene index using the target query q_T. This effectively produces a list of recommendations based on the overlap between the query terms and the terms used to index each recommendation (query, snippet, and tag terms). Next, these recommendations are then filtered and ranked. Results that do not exceed certain activity thresholds are eliminated as candidates; such as, for example, results with only a single selection or results with more negative votes than positive votes (see [3]). Each remaining recommendation candidate r is then ranked according to a weighted sum of its relevance (*rel*) and reputation (*rep*) scores at time t as per Equation 2; where w is used to adjust the relative influence of relevance and reputation.

$$score(r, q_T, t) = w \times rep(r, t) + (1 - w) \times rel(q_T, r) \; . \quad (2)$$

The relevance of a result r with respect to a query q_T is computed based on Lucene's standard *TF-IDF* metric as per Equation 3; *TF-IDF* is a well-known information retrieval term-weighting function [12] that gives high weights to terms that are popular for a result r but rare across other stak results, thereby serving to prioritise results that match distinguishing index terms.

$$rel(q_T, r) = \sum_{\tau \in q_T} tf(\tau \in r) \times idf(\tau \in r)^2 \; . \quad (3)$$



(a) (b)

Fig. 2. Collaboration and reputation: (a) the consumer c selects result r, which has been recommended based on the producer p's previous activity, so that c confers some unit of reputation (*rep*) on p. (b) The consumer c selects a result r that has been produced by several producers, $p_1, ..., p_k$; reputation is shared amongst these producers with each user receiving an equal share of *rep/k* units of reputation.

IV. Reputation as Collaboration

The above relevance model pays no attention to the *source* of the recommendation; i.e. the users who originally contributed the page to a stak or whose subsequent activities resulted in the page being recommended. For this reason recent research has looked at the possibility of adding a reputation component to recommendation. Recommendation candidates can be scored by a combination of relevance and reputation according to Equation 2, so that pages that have been contributed by many reputable users are considered more eligible for recommendation than those that have been contributed by fewer, less reputable users.

The key idea in the user reputation model for HeyStaks is that reputation can be calculated by mining the implicit collaborations that occur between users as a result of their searches. If HeyStaks recommends a result to a searcher, and the searcher chooses to act on this result (i.e. select, tag, vote or share), then we can view this as a single instance of search collaboration. The current searcher who chooses to act on the recommendation is known as the *consumer* and, in the simplest case, the original searcher, whose earlier action on this result caused it to be added to the search stak, and ultimately recommended, is known as the *producer*. In other words, the producer created search knowledge that was deemed to be relevant enough to be recommended and useful enough for the consumer to act upon it. The basic idea behind the user reputation model is that this act of implicit collaboration between producer and consumer confers some *unit of reputation* on the producer (Figure 2(a)), and to calculate the overall reputation of a user (producer) we need to aggregate these units of reputation across past collaborations.

The reputation model calculates the reputation of a producer as a weighted sum of the collaboration events in which they have participated. The simplest case is captured by Figure 2(a) where a single producer participates in a collaboration event with a given consumer, benefitting from a single unit of reputation as a result. More generally, at the time when the consumer acts (selects, tags, votes etc.) on the recommended result, there may have been a number of past producers who each contributed part of the search knowledge that caused this result to be recommended. A specific producer may have been the first to select the result in a given stak, but subsequent users may have selected it for different queries, or they may have voted on it or tagged it or shared it with others independently of its other producers. Thus we need to be able to share reputation across these different producers; see Figure 2(b).

More formally, let us consider the selection of a result r by a user c, the consumer, at time t. The producers responsible for the recommendation of this result are given by $producers(r, t)$ as per Equation 4 such that each p_i denotes a specific user u_i in a specific stak S_j.

$$producers(r, t) = \{p_1, ..., p_k\} \ . \tag{4}$$

Then, for each producer of r, p_i, we update its reputation as in Equation 5. Reputation is shared equally among its k contributing producers.

$$rep(p_i, t) = rep(p_i, t-1) + 1/k \ . \tag{5}$$

In this way reputation is based on the accumulation of collaboration instances and essentially the reputation of a user is a weighted sum of the number of collaborations they have contributed to by way of their past searches. Importantly this is just one specific way to model reputation according to this producer-consumer model; there are many other ways to count and accumulate reputation and in [13] we describe and evaluate of a variety of different approaches.

In this paper, however, we assume the above user reputation model and focus instead on how it can be incorporated into the HeyStaks recommendation model. To do this we need a way to translate *user* reputation scores into corresponding *page* reputation scores which we consider in the following section.

V. Page Reputation Models

In this section we describe a number of approaches to model page reputation. In each case, the goal is to calculate the reputation score of a result page r at time t based on the reputation scores of the page's producers at that point in time; see Equation 6.

$$rep(r, t) = f\big(rep(p_1, t), ..., rep(p_k, t)\big) \ . \tag{6}$$

For the purpose of illustration we will calculate each reputation score based on a hypothetical recommendation scenario for a page r which is associated with a set of 10 producers with the following reputation scores at time t: {0.003, 0.014, 0.023, 0.052, 0.089, 0.097, 0.154, 0.297, 0.348, 0.581}. This includes a cross section of producers including some with low reputation scores and some with high scores. Note that in practice producer reputation scores are first normalised by the maximum producer reputation in the corresponding stak to ensure a score between 0 and 1.

A. Median Reputation

Perhaps the simplest way to translate user reputation into page reputation is to calculate the average reputation of the page's producers. We propose to do this by finding the median reputation of the producers as follows:

$$rep(r, t) = median\big(rep(p_1, t), ..., rep(p_k, t)\big) \ . \tag{7}$$

The advantage of this approach over a simple mean reputation is that the median statistic tends to better represent the central tendency of the set of user reputations. In the case of our hypothetical recommendation scenario the reputation of the page r is 0.093 according to this median model.

B. Max Reputation

Another simple way of scoring a page based on the reputation of its producers is to take the maximum reputation value from that set. Formally, Max Reputation is calculated thus:

$$rep(r,t) = max\big(rep(p_1,t),...,rep(p_k,t)\big)\ . \tag{8}$$

Scoring pages in this way is advantageous as the reputation of a page will not be harmed if, for example, many new, not yet reputable users have selected the page. In our recommendation scenario, the reputation of page r is 0.581 by this approach.

C. Harmonic Mean Reputation

Harmonic Mean is an average measure that tends towards the lower bound of a set of numbers, and thus is more conservative than arithmetic mean or median. It is calculated by finding the reciprocal of the arithmetic mean of the reciprocals. Formally, the harmonic mean of a set of user reputation scores is calculated as:

$$rep(r,t) = \frac{k}{\sum_{i=1}^{k} \frac{1}{rep(p_i,t)}}\ . \tag{9}$$

In this case, the reputation of page r is 0.020. Harmonic mean may be a good indicator of the utility of a page in the sense that a page is only as reputable as its least reputable producer. However, rather than simply using the minimum producer reputation score available, harmonic mean permits the full range of producer reputation scores to influence the overall page reputation.

D. Root Mean Square Reputation

In order to get a sense of the magnitude of a set of user reputation scores, it is useful to calculate the root-mean-square (RMS) of those values. RMS is defined as the square root of the mean of the squares.

$$rep(r,t) = \sqrt{\frac{\sum_{i=1}^{k} rep(p_i,t)^2}{k}}\ . \tag{10}$$

As RMS is a measure of the magnitude of a set of numbers, this value tends towards the upper bound of a set of user reputation scores. As such, RMS can be considered as a less conservative method of calculating page reputation compared to the previous harmonic mean approach. As per our worked example, the reputation of page r using RMS is 0.243.

E. Hooper's Reputation

In order to reinforce a page's reputation according to the number of producers, keeping in mind their score, a different approach is required. A simple technique is George Hooper's *Rule for Concurrent Testimony*, originally proposed as a technique to calculate the credibility of human testimony [14]. This is applicable in our case in the sense that users who have produced a result in HeyStaks are endorsing it, in the same way that a group of witnesses might attest to the same report. Hooper gives to the report a credibility of $1-(1-c)^k$, assuming k reporters, each with a credibility of c (where $0 \leq c \leq 1$). For HeyStaks, the quality of a page can be determined by performing the same calculation across the reputation scores if its producers.

$$rep(r,t) = 1-(1-c)^k\ . \tag{11}$$

As per [14], we can calculate the reputation of result r as $0.003 + (1-0.003)0.014 + (1-0.003)(1-0.014)0.023$ ... and so on. The reputation of this particular result r is 0.865.

In the following section we evaluate these five models by examining their influence on recommendations made by HeyStaks, according to Equation 2. Of course, the success of each model is determined by the extent to which they improve the effectiveness of the HeyStaks recommendation engine, which we analyse using a set of queries submitted to HeyStaks during the course of a live-user trial.

VI. Evaluation

In previous work [3] we have demonstrated how the standard relevance-based recommendations generated by HeyStaks can be more relevant than the top ranking results delivered by Google. In this work we wish to compare this relevance-based recommendation technique to an extended version of HeyStaks that also includes page reputation.

The purpose of this paper has been to propose a number of alternatives to calculating the reputation of content based on that of its producers who are helping other users (consumers) to search within the HeyStaks social search service. The hypothesis is that by allowing reputation, as well as relevance, to influence the ranking of result recommendations, we can improve the overall quality of search results. In this section we evaluate our page reputation models using data generated during a recent closed, live-user trial of HeyStaks, designed to evaluate the utility of the HeyStaks brand of collaborative search in fact-finding information discovery tasks.

A. Dataset and Methodology

Our live-user trial involved 64 first-year undergraduate university students with varying degrees of search expertise. Users were asked to participate in a general knowledge quiz, during a supervised laboratory session, answering as many questions as they could from a set of 20 questions in the space of 1 hour. Each student received the same set of questions which were randomly presented to avoid any ordering bias. The questions were selected for their obscurity and difficulty; see [15] for a list of questions used in the trial. Each user was allocated a desktop computer with the Firefox web browser and the HeyStaks toolbar pre-installed; they were permitted to use Google, enhanced by HeyStaks functionality, as an aid in the quiz. The 64 students were randomly divided into search groups. Each group was associated with a newly created search stak, which would act as a repository for the groups' search knowledge. We created 6 *solitary* staks, each containing

just a single user, and 4 *shared* staks containing 5, 9, 19, and 25 users. The solitary staks served as a benchmark to evaluate the search effectiveness of individual users on a non-collaborative search setting, whereas the different sizes of shared staks provided an opportunity to examine the effectiveness of collaborative search across a range of different group sizes. All activity on both Google search results and HeyStaks recommendations was logged, as well as all queries submitted during the experiment. Although trial users were made fully aware of how to use HeyStaks and its benefits, they were not explicitly encouraged to use the system during the trial, nor were they encouraged to perform one activity over another (sharing over tagging, for example).

During the 60 minute trial, 3,124 queries and 1,998 result activities (selections, tagging, voting, popouts) were logged, and 724 unique results were selected. During the course of the trial, result selections — the typical form of search activity — dominated over HeyStaks-specific activities such as tagging and voting. On average, across all staks, result selections accounted for just over 81% of all activities, with tagging accounting for just under 12% and voting for 6%.

In recent work we described the performance results of this trial showing how larger groups tended to benefit from the increased collaboration effects of HeyStaks [15]. For example, members of shared staks answered significantly more questions correctly, and with fewer queries, than the members of solitary staks who did not benefit from collaboration. In this paper we are interested in exploring how we can utilise reputation to measure the quality of recommendations. No reputation model was used during the live-user trial and so recommendations were ranked based on relevance only. However the data produced makes it possible for us to effectively *replay* the user trial so that we can construct our user reputation model and test each page reputation model by using each to re-rank HeyStaks recommendations. In order to ensure accuracy, we calculate the reputation of each recommendation made during the trial according to the reputation of its producers at the time the recommendation was made. We can retrospectively test the quality of re-ranked results versus the original ranking against a ground-truth relevance; since as part of the post-trial analysis, each selected result was manually classified as *relevant* (the result contained the answer to a question), *partially relevant* (the result referred to an answer, but not explicity), or *not-relevant* (the result did not contain an explicit or implicit reference to an answer) by experts.

B. User Reputation

To get a sense of how users were scored by our model, we first examine the type of user reputation values that are generated from the trial data. It should be noted that each trial user was a member of only one stak, and thus could only receive (as a consumer) recommendations resulting from the activities of fellow stak members, and likewise could only gain reputation (as a producer) from the consumer activity of other members of the stak. Figure 3 shows box-plots displaying user reputation scores for each of the 4 shared staks at the end of the trial. We see there is a clear difference in the median reputation score for members of the 5 person stak when compared to members of the larger staks. Despite the most reputable user in the trial hailing from the large 19-person stak, it is the 9-person stak which has the highest median reputation score. The interquartile range of reputation scores in this stak was relatively small, indicating that these users worked closely in collaboration with each other during the trial. It was these users who collectively outperformed most other users in the quiz, scoring higher marks and, on the whole, achieving a better correct answer per query rate (see [15] for more details). The box-plots show that there is a wide variation in reputation scores: Some users, particularly in the stak with 5 and 25 members, achieved an almost negligible amount of reputation. On the other hand, others received a score in excess of 20, the most reputable user scoring 37. These users were the primary drivers of search collaboration during the quiz.

C. Evaluating Page Reputation Models

The true test of the reputation models in this work is the extent to which they improve in the quality of results recommended by HeyStaks. We have described how HeyStaks combines term-based relevance and user reputation to generate its recommendation rankings; see Equation 2. For the purpose of this evaluation we regenerate each of the recommendation lists produced during the trial using each of the page reputation models, based on the user reputation scores calculated at the appropriate point in time. Since we have ground-truth relevance information for all of the recommendations (relative to the quiz questions), we can then determine the quality of the resulting recommendations. Specifically, we focus on the top recommended result and note whether it is relevant (that is, contains the answer to the question) or not relevant (does not contain the answer to the question). For each page reputation model we compute an overall relevance rate, as the ratio of the percentage of recommendation sessions where the top result was deemed to be relevant, to the percentage of those where the top result was not-relevant. Moreover, we can compare this to the relevance rate of the recommendations made by the standard HeyStaks ranking (i.e. when $w = 0$ in Equation 2) during the trial to compute an overall *relevance benefit*;

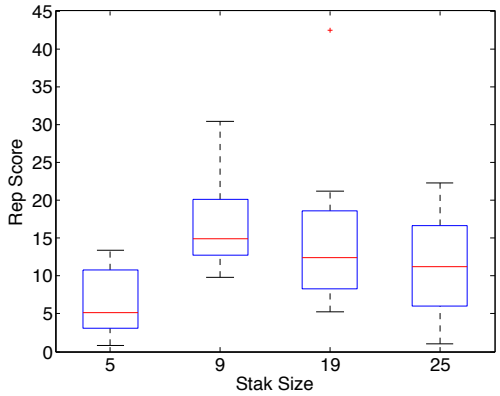


Fig. 3. User reputation score per user, per stak.

such that a relevance benefit of 40%, for a given reputation model, means that this model generated 40% more relevant recommendations than the standard HeyStaks ranking scheme.

Figure 4 presents a graph of relevance benefit versus the weighting (w) used in Equation 2 to adjust the influence of term-based relevance versus page reputation during recommendation. Remember that as a recommendation is made, its reputation score is calculated based on the reputation of its producers at the time the recommendation is made. A similar trend can be seen in the figure for each technique in that relevance benefit is observed to increase initially with w before decreasing as w approaches 1. Peak performance occurs at different weights for each technique. For example, Harmonic Mean peaks at $w = 0.2$ before dipping below 0% relevance benefit for all remaining weights. Hooper, the best performing technique by some distance, peaks twice at $w = 0.4$ and $w = 0.8$, each time achieving around 55% relevance benefit. Four of the five techniques result in considerable improvement over the standard HeyStaks relevance-based model, each achieving at least 30% relevance benefit at their respective optimal weighting. Harmonic Mean was the worst performer, only managing a maximum of 3.4% relevance benefit, and for most weights it delivered a negative relevance benefit, meaning the standard HeyStaks recommendation engine delivered proportionately more relevant results across the trial. This is most likely due to the fact that harmonic mean tends towards the lower bound of a set of producer reputation scores: if, for example, a page that has been produced by many users with varying reputation, the harmonic mean of those scores will tend towards the lower bound of the set. Conceptually, we may not wish to punish pages whose producer reputation scores have high variance, particularly where some producers have high reputation scores. Hooper achieves the best relevance benefit of all, and at 55% for $w = 0.4$, represents a realistic option for integration into a live HeyStaks reputation engine. This may be the most suitable option as the score it produces for a page is a consensus based on the reputation of its producers. The technique promotes the idea that a page will have a high score by way of reinforcement from its producers, assuming they are reputable.

Figure 5 shows the median relevance benefit across weights for each page reputation model. A Kruskal-Wallis test indicates that there are statistically significant differences between the performance of the reputation models at the .01 level. Examining the pairwise differences between models, Tukey's Range test indicates that there are significant differences between Hooper's technique and two other techniques at the .05 level, Harmonic Mean and Root Mean Square. These findings further highlight the strong performance achieved by the Hooper page reputation model.

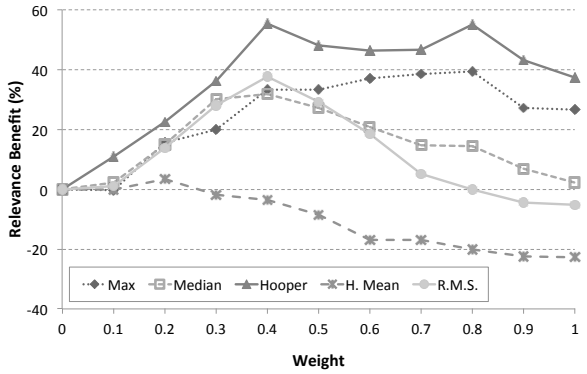


Fig. 4. Relevance benefit for page reputation models vs. weight.

D. Limitations

In the above we have compared a number of page reputation models to map user reputation scores onto the content users produce, based on real-user search data. One limitation of this approach is that although the evaluation uses trial-user data, the final recommendations are not themselves evaluated using the trial users. Instead we *replay* users' searches to generate reputation-enhanced recommendations. The main reason for this is the difficulty in securing sufficiently many users for a trial of this nature, which combines a number of reputation models and therefore a number of experimental conditions. That being said, our evaluation methodology is sound since we evaluate the final recommendations with respect to their ground-truth relevance. We have an objective measure of page relevance based on the Q&A nature of the trial and we use this to evaluate the genuine relevance of the final recommendations. The fact that our reputation models deliver relevance benefits above and beyond the standard HeyStaks recommendation algorithm is a clear indication that reputation provides a valuable ranking signal. Of course this evaluation cannot tell whether users would actually select these reputation-ranked recommendations, although there is no reason to believe that they would treat these recommendations differently from the default HeyStaks recommendations, which they are inclined to select. We view this as a matter for future work.

Another point worth noting is that the user trial is limited to a specific type of search task, in this case a Q&A search task. As such it would be unsafe to draw general conclusions in relation to other more open-ended search tasks. However, this type of focused search task is not uncommon among web searchers and as such we feel it represents an important and suitable use-case that is worthy of evaluation. Moreover, previous work [3] has examined the role of HeyStaks in more open-ended search tasks, where its default relevance-based recommendations were also found to be beneficial to end-

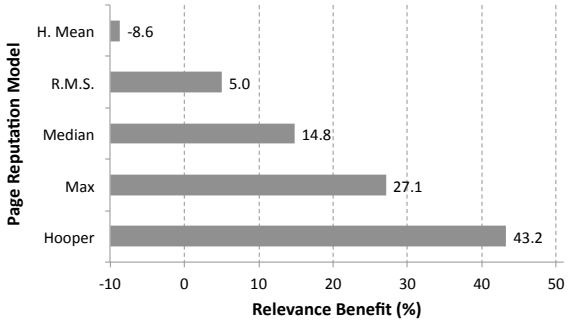


Fig. 5. Median relevance benefit across weights per page reputation model.

users. As part of our future work we are currently in the process of deploying and evaluating our reputation models across similar general-purpose search tasks.

VII. Conclusions

This paper is about social search and broadly speaking the research looks at how we can make web search more collaborative. In particular we describe the HeyStaks social search platform which brings collaborative web search to mainstream search engines such as Google, Bing, and Yahoo, via browser plugins. The specific contribution of this paper is the introduction of page reputation models that can be used to influence result recommendations made by HeyStaks at search time, recommendations that originate from the past searches of communities of collaborating searches. The HeyStaks user reputation model can be used to model the effectiveness of a user from a search standpoint. For example, users whose search results are frequently recommended to, and acted on by, other users are considered to be reputable searchers. The intuition behind this work is that these reputation scores can be viewed as a type of evidence in support of page recommendations so that recommendation candidates that come from many reputable users (producers) are considered to be more reliable that recommendations from less reputable users.

In this paper we have explored different ways to translate user reputation scores into an overall page reputation/relevance score. We have described the results of a comparative evaluation in the context of real-user data which highlights the ability of these techniques to improve overall recommendation quality, when combined with the relevance-based recommendation ranking metrics that are currently used by HeyStaks. For example, many of the page reputation models can improve recommendation relevance (compared to the standard HeyStaks benchmark) by over 30%. Moreover, we have found that one model, based on Hooper's rule for Concurrent Testimony [14] is capable of delivering relative improvements of up to 55%. We believe that this work lays the ground-work for future research in this area which will focus on scaling-up the role of reputation in HeyStaks and refining the combination of relevance and reputation during recommendation.

In this paper we have focused on the role of reputation during the recommendation process, in order to maximise the relevance of the community recommendations made by HeyStaks. But this is just one use of reputation in a system such as HeyStaks. For example, in many social systems there is the risk that malicious users will attempt to manipulate the outcome of social processes; see relevant work in recommender systems research [16]–[18]. In HeyStaks, for example, it is possible to malicious users to flood search staks with irrelevant or self-interested results, which could impact recommendation quality. By using reputation to mediate recommendation it will be possible to guard against this; these malicious users will have low reputation scores (assuming their contributions are rarely acted on by other users) and as such their contributions will be unlikely to appear in future recommendation sessions. Furthermore, reputation can be exposed to users of systems like HeyStaks as an important social signal. For example, although HeyStaks' recommendations are anonymous (so users do not know the source of result recommendations at search time) it may make sense to explain recommendations with reference to the reputation of producers in the future.

Acknowledgment

This work is supported by Science Foundation Ireland under grant 07/CE/I1147.

References

[1] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," in *WWW '98: Proceedings of the 7th International Conference on World Wide Web*. Brisbane, Australia: ACM, 1998, pp. 107–117.

[2] J. M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[3] B. Smyth, P. Briggs, M. Coyle, and M. P. O'Mahony, "Google Shared. A Case Study in Social Search," in *User Modeling, Adaptation and Personalization*. Springer-Verlag, June 2009.

[4] K. McNally, M. P. O'Mahony, B. Smyth, M. Coyle, and P. Briggs, "Towards a Reputation-based Model of Social Web Search," in *IUI '10: Proceedings of the 14th International Conference on Intelligent User Interfaces*, Hong Kong, China, 2010, pp. 179–188.

[5] M. Balabanović and Y. Shoham, "Fab: Content-based, collaborative recommendation," *Communinations of the ACM*, vol. 40, no. 3, pp. 66–72, 1997.

[6] U. Shardanand and P. Maes, "Social information filtering: Algorithms for automating "word of mouth"," in *CHI '95: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1995, pp. 210–217.

[7] B. Smyth and P. Cotter, "A personalised TV listings service for the digital TV age," *Knowledge-Based Systems*, vol. 13, no. 2–3, pp. 53–59, 2000.

[8] J. B. Schafer, J. A. Konstan, and J. Riedl, "E-commerce recommendation applications," *Data Mining and Knowledge Discovery*, vol. 5, no. 1-2, pp. 115–153, 2001.

[9] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[10] J. O'Donovan and B. Smyth, "Trust in Recommender Systems," in *IUI '05: Proceedings of the 10th International Conference on Intelligent User Interfaces*, 2005, pp. 167–174.

[11] P. Massa and P. Avesani, "Trust-aware Recommender Systems," in *RecSys '07: Proceedings of the 2007 ACM Conference on Recommender Systems*. ACM, 2007, pp. 17–24.

[12] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[13] K. McNally, M. P. O'Mahony, and B. Smyth, "Evaluating User Reputation in Collaborative Web Search," in *3rd Workshop on Recommender Systems and the Social Web, in association with The 5th ACM Conference on Recommender Systems (RecSys 2011)*, 2011.

[14] G. Shafer, "The Combination of Evidence," *International Journal of Intelligent Systems*, vol. 1, no. 3, pp. 155–179, 1986.

[15] K. McNally, M. P. O'Mahony, and B. Smyth, "Social and Collaborative Web Search: An Evaluation Study," in *IUI '11: Proceedings of the 16th International Conference on Intelligent User Interfaces*. Palo Alto, California USA: ACM, 2011, pp. 387–390.

[16] S. K. Lam and J. Riedl, "Shilling recommender systems for fun and profit," in *WWW '04: Proceedings of the 13th International World Wide Web Conference*. New York, NY, USA: ACM, May 17–20 2004, pp. 393–402.

[17] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, "Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness," *ACM Transactions on Internet Technology (TOIT)*, vol. 7, no. 4, pp. 1–40, 2007.

[18] M. P. O'Mahony, N. J. Hurley, and G. C. M. Silvestre, "Promoting recommendations: An attack on collaborative filtering," in *DEXA '02: Proceedings of the 13th International Conference on Database and Expert Systems Applications*. Aix-en-Provence, France: Springer, 2002, pp. 494–503.