An ant-colony based approach for real-time implicit collaborative information seeking

Alessio Malizia[a,*], Kai A. Olsen[b,c], Tommaso Turchi[a], Pierluigi Crescenzi[d]

[a]Human Centred Design Institute, Brunel University London, London, UK
[b]Faculty of Logistics, Molde University College, Molde, Norway
[c]Department of Informatics, University of Bergen, Bergen, Norway
[d]Department of Information Engineering, University of Florence, Florence, Italy

Abstract

We propose an approach based on Swarm Intelligence — more specifically on Ant Colony Optimization (ACO) — to improve search engines' performance and reduce information overload by exploiting collective users' behavior. We designed and developed three different algorithms that employ an ACO-inspired strategy to provide implicit collaborative-seeking features in real time to search engines. The three different algorithms — NaïveRank, RandomRank, and SessionRank — leverage on different principles of ACO in order to exploit users' interactions and provide them with more relevant results. We designed an evaluation experiment employing two widely used standard datasets of query-click logs issued to two major Web search engines. The results demonstrated how each algorithm is suitable to be employed in ranking results of different types of queries depending on users' intent.

Keywords: Ant Colony Optimization, Cooperative Systems, Evolutionary Computation, Information Filtering, Information Retrieval, Recommender Systems, World Wide Web

*Corresponding author

Email addresses: alessio.malizia@brunel.ac.uk (Alessio Malizia), kai.olsen@himolde.no (Kai A. Olsen), tommaso.turchi@brunel.ac.uk (Tommaso Turchi), pierluigi.crescenzi@unifi.it (Pierluigi Crescenzi)

1. Introduction

Traditionally text retrieval was based on keywords. However, not all documents had been adequately tagged, neither could the keywords describe all aspects of a document. With faster computers, it became possible to perform full-text searches. Then we got the problem of too many hits, i.e. the supplied keywords were found in too many documents. One tried to cope with this by determining relevance as the number of occurrences of each search term in the document, in relation to document size. The first search engines on the Web used this approach.

There were several disadvantages to this approach. Looking for information on a given car, using maker and model as keywords, the search engine did not direct you to any official site. Instead, one was overloaded with car for sale advertisements, as these had a good occurrence to size ratio of the keywords. It was also quite easy to fool the search engines, for example by adding long list of repeated keywords to a Web page, often using a small white font so that it did not clutter the page.

Google's PageRank algorithm saved the day. By letting relevance be determined by the number of links to a page, adding up the score if the links also came from pages that had many links to them, Google had captured a semantic understanding of the relevance concept. For example, many Web pages may say something about The White House, and there may be many white houses, but Google puts the official site on top, most probably the page that the user wants. And every time someone makes a link to this page, they increase its relevance.

The disadvantage of this approach is that it is static. Pages found important by the PageRank algorithm will probably get more important as they are found by the search engine. That is, important pages will get more important. New pages on similar topics will be hard to find, i.e. placed further down on the search engine result page, and will thus be considered less important. Over time, an algorithm used to determine relevance might be self-fulfilling.

Ideally, we would need a search algorithm that were more dynamic, but still gave a good idea of relevance. Our idea is to use data from the actual searches - what we call dynamic trail information. While PageRank uses static information as link structure, we want to collect data from the actual searches performed by other users. For example, you may be interested in renting a boat to go deep-sea fishing outside the Lofoten Islands in Northern

Norway. Your keywords may be rent, boat, fishing, Lofoten. The search engine will then return a standard list of relevant pages; however, in addition you will find a list that says: "other users found these pages". That is, the system have collected data on what other users with similar query terms did. They may have started with the same keywords as you, but may also tried other searches, ending up with a few interesting pages. That is, the effort that other users have put in finding relevant pages can be important information to you.

The data needed to offer an "other users found these pages" list can be collected quite easily, but one will need access on the server level, i.e. to collect data from many users. One could strengthen the trail if the user performed some action at the end. This could either be implicit, such as noting that the users stayed on the site for some time, typed in data, printed from the page, bought or booked, etc. Alternatively, it could be explicit, where the users use a "like" button to tell that the page is interesting, e.g. the Google+1 service[1].

Such an approach falls into the implicit collaborative information-seeking area in which developing new collaborative search interfaces is still needed, as recently suggested by Hearst [1].

According to Golovchinsky et al. [2], a collaborative information search system can be either implicit or explicit, meaning that users can explicitly collaborate on query formulations and review search results or can implicitly take advantage of other users' search intents. Normally, implicit collaboration systems provide a recommendation and filter the results already explored by previous users, making them available to others with similar information needs.

The majority of studies in the implicit area are based on collaborative querying techniques that upgrade information systems with data on past query preferences related to other users. As recently demonstrated [3], such studies primarily tested implicit collaborative information-seeking systems using simulated query formulation instead of employing user analysis involving human participants. In our research, we employed a classic approach by using two existing datasets to simulate queries to evaluate our system in a real setting.

Hence, we deal with the problem of improving search engines' perfor-

[1]`https://developers.google.com/+/web/+1button/`

mance by exploiting the actions performed by users. In fact, search engines are tools designed to help people solve their own informational needs and significant room exists for improvements. Queries submitted to search engines can be clustered into three main categories on the basis of users' aim [4]:

Informational queries are issued by users willing to acquire information that they assume is present on one or more Web pages;

Navigational queries are being used to get to a particular Web page belonging to an organization or an individual; and,

Transactional queries are issued to perform activities using the Web, such as booking a trip or downloading a file.

Ranking results produced through navigational queries can be effectively addressed using existing Link Analysis Ranking (LAR) algorithms, such as PageRank, Hyperlink-Induced Topic Search (HITS), or Stochastic Approach for Link-Structure Analysis (SALSA): a higher number of hyperlinks pointing toward one particular page results in a higher page relevance (in other words, algorithms assume that this page is the one that the user was looking for when she issued the query). Ranking results of informational and transactional queries is another matter: given the high frequency of Web pages' updates and the ever-increasing need to obtain answers in real time, the World Wide Web hyperlinks' configuration is no longer the only effective relevance measure that users assign to Web pages. Thus, devising new relevance indicators — to be placed alongside the existing ones (in other words, those based on Link Analysis Ranking) — with the goal of further improving the ranking by considering other relevance measures valued by users is necessary [5].

In this paper, we propose to employ the concepts of Swarm Intelligence (SI) in relation to the Ant Colony Optimization (ACO) meta-heuristic to improve search engine performance and to reduce the information overload[2] by exploiting collective users' behavior in their usage of search engines.

[2]The inability to make a decision because of the huge quantity of information obtained by the users.

2. Related Work

2.1. Search Engines

Different studies pointed out users' low degree of satisfaction with search engines. Fox et al. [6] devised a machine learning approach that employs users' actions (for example the time spent on a page, scrolling usage, and page visits) and concluded that users consider 28% of search sessions unsatisfactory and 30% only partially satisfactory. Xu and Mease [7] measured the average duration of a search session and found that users typically quit a session — even without having satisfied their informational need — after three minutes.

The main purpose of search engines is to satisfy users' informational needs, thus they are being used as a starting point of users' Web browsing [8, 4, 9]; nevertheless, the search experience is far from perfect. In fact, a substantial number of searches end up unsatisfied. Many researchers attempted to improve search engines results' relevance by exploiting query-click logs (in other words, logs of all the interactions users carry out with the search engine), usually in the form of click-through data. Joachims [10] was the first to exploit these logs as implicit relevance judgments about search engine results and trained a meta-search engine to outperform many other famous ones. After the work of Joachims, using query-click logs to improve search engine performance became a widely popular technique [11].

2.2. Information Foraging on the Web

Many theories attempted to explain users' behavior when searching for information in complex systems (for example, the Web). For the scope of this paper, we refer to two approaches related to the proposed algorithms: the ScentTrails system [12], which continuously allows users to supply keywords and enriches hyperlinks to provide a path that achieves the goal described by them, and the method by Wu and Aberer [13], which operates within a single Website to enrich the information provided by hyperlinks with a technique inspired by ant-foraging behavior (in other words, heavily clicked links are recommended in favor of less visited links).

2.3. Learning to Rank

Learning to Rank aims at automatically learning the right ranking function from a training set, typically a click-through dataset. Joachims [10] outlined three major key points: (1) explicit feedback provided by users cannot be taken for granted and, as a matter of fact, is unnecessary because the information given by the query-click logs are enough; (2) in fact, they can be used as a measure of relevance (with a relative scale); and (3) a machine learning method — Joachims used a Support Vector Machine (SVM) — can be used to obtain a new ranking function that improves search engine performance.

In addition to SVMs, other machine learning algorithms may be used, such as RankBoost [14], RankNet [15], QBRank [16], GBRank [17], AdaRank [18], and MCRank [19].

However, Learning to Rank approaches exhibit two drawbacks: (1) the training phase is computationally expensive and faster methods are being sought [20]; (2) because most of the outlined machine learning methods are offline, the system must be trained again each time new data become available, which occurs quite frequently because we are dealing with click-through data; thus, online methods are also being investigated [21, 22, 23].

In conclusion, it is important to point out that Learning to Rank techniques are not the only ones employed in training ranking functions: other studies described alternative soft computing methods, such as genetic programming [24] and Swarm Intelligence (SI) [25].

3. Ant Colony Ranking

In the introduction, we described how information overload is a major problem affecting Internet users and outlined some useful approaches to address this issue: software agents, implicit feedback, collaborative filtering, and assisted browsing/searching [26].

Given the current wide usage of Web search engines, we focused on all the unsatisfactory searches with the goal of addressing the information overload problem. Some techniques aiming at improving their performance have been summarized (for example, Learning to Rank) to highlight a few key concepts: (1) the relationship between users seeking information and the optimal foraging theory; (2) the need for a search engine to adapt itself to users' behavior; and (3) the need to perform such adaptation in real time.

Almost none of the aforementioned approaches take into account all three of these aspects, as stated by Wu and Aberer [13] and Olston and Chi [12]. Beyond a doubt, a Swarm-based approach can take into account all three key factors and is, nonetheless, a much more elegant and simple method than all of the other "ad-hoc" alternatives.

For these reasons, in the next section we introduce a model able to describe ranking algorithms inspired by Swarm Intelligence (SI) that can improve the performance of a search engine by adapting themselves to users' behavior.

4. A Model for Ant Colony Ranking Algorithms

Each day ants leave the colony in search of food and building materials; they will exploit the surroundings in all directions in a somewhat random fashion. If an ant finds anything of interest, it will return to the colony depositing pheromone, a chemical substance that the other ants are able to detect. Thus they create trails to signal the path between the colony and the food. The quantity of pheromone deposited, which may depend on the quantity and quality of the food, will guide other ants to the food source. That is, the other ants in the colony may now use the pheromone as a trail marker to reach the food. This marker evaporates over time, so that uninteresting trails disappear. Shorter trails will get a higher level of pheromone, thus shorter trails will endure longer, providing a notion of optimization.

Normally, ants from different colonies exhibit aggression toward each other. However, some ants exhibit the phenomenon called unicoloniality. Here worker ants freely mix between different colonies. These species of ants live in populations known as supercolonies that may be used to characterize social behavior on the Web.

Let us assume that a set of users all start with the same query, for example "compact camera GPS". That is, they are all interested in finding Web sites that can offer a good bargain for such a camera ("food"). Our group may start with a Google query, and click on links to explore the results. These click streams will define our "pheromone" or virtual trail. They may for example be implemented by adding score values to each link, or visualized by representing the links by large fonts, stronger color, etc.

However, on the Web we can optimize, leaving the trail metaphor and lead subsequent users directly to interesting pages. That is, we let our ants (users) explore the Web, but we let them deposit the pheromone on the most interesting pages. The rest of the colony (i.e. other users with similar interest) can then go directly to these sites.

Swarm Intelligence (SI) refers to the emergence of “intelligent” behavior from a group of simple and/or loosely organized agents. Ants are a typical example of SI and their use of stigmergic processes[3] inspired the famous family of Ant Colony Optimization (ACO) algorithms [27, 28] and many variants, including Max-Min Ant System (MMAS) [29], Continuous Orghogonal Ant Colony (COAC) [30], and Rank-based Ant System (ASrank) [31]. These classes of algorithms are bio-inspired (Ant Colony) probabilistic metaheuristics for solving computational problems related to searching for an optimal path in a graph; the probabilistic nature of such techniques — along with some basic rules driving agents towards appropriate solutions — allows for their convergence to an optimal solution, avoiding local optimums.

As we have previously stated, we will adapt the strategies employed in food searching by ant colonies in the building of ranking algorithms employing users' behavior; without a doubt, humans are more intelligent and organized than ants. However, some complex phenomena stems from Web surfing, since collective activities like Wikipedia, del.icio.us, or even the entire Web, are indeed stigmergic processes [32, 33].

Summarizing, users surfing the Web issue relevance judgments every time they submit a query and select a result among the ones provided by a search engine. Ultimately, a SI-based approach seems a valid idea to make such systems able to exploit users' seeking behavior.

It's pretty intuitive to find a parallelism between the way ants forage for food and the way users employ search engines to satisfy their informational needs; yet the latter, unlike ants, don't leave any trace at all, so they can't provide any clues to the next users with their same informational needs, and — since about 30–40% of queries issued to a search engine are already been submitted [34] — that's a pretty common scenario.

So, by using a virtual form of pheromone — controlled in the same way as the one used by ants — it's possible to define a ranking algorithm that ranks relevant results based on the pheromone left on each document: the more we'll find on a document the higher its ranking will be, since that document was considered relevant by a large amount of users.

[3] Pierre-Paul Grasse introduced the term in 1950s during his research on termites. It is defined as a method of communication based on individuals modifying their surrounding environment.

4.1. Formalization

Here we introduce a brief formalization of the model we just proposed. We will assume that interactions between users and the search engine are available in the form of query-sessions — briefly "sessions": by the definition of Wen and Zhang [35], a session is formed by the query a user submitted to the search engine — i.e. the text describing what he/she is looking for — together with the visited Web pages consequently to his/her request; an example of a query session can be found in table A.2 in the Appendix.

Borrowing the notation of [35], let $D(q)$ be the set of Web pages the search engine presents to the user as results for the query q, selected by filtering only the relevant ones for q through any available retrieval strategy [36]. The page set a user clicked on for a query q may be seen as

$$DC(q) = \{d_{q1}, d_{q2}, \ldots, d_{qi}\} \subseteq D(q),$$

where d_{qi} represents the i-th document the user clicked among the results of the query q (i.e. d_{q1} being the first selected result — if any — d_{q2} the second — if any — and so on); on the other end, we denote d_q^i the document currently ranked in position i among the results of the query q by the ranking algorithm in use.

The pheromone associated to a document d with respect to a query q is denoted by φ_{dq} and is updated every time a user selects the document among $D(q)$ or — carrying on the similarity with ACO — he/she covers the path $q \to d$; the amount of pheromone deposited each time depends on the specific user session $DC(q)$ and will be detailed in the next section.

Considering each document d and any known query q, pheromone evaporation follows an exponential decay based both on the current value φ_{dq} and the elapsed time since its last update — denoted with τ_{dq}. In mathematical terms, denoting the new pheromone value by φ'_{dq}, the evaporation rule can be expressed by the equation

$$\varphi'_{dq} = \varphi_{dq} e^{\lambda \tau_{dq}},$$

where λ is the exponential decay constant; this rule can be transformed in a much simpler version by defining a new constant

$$\delta = \frac{\ln(2)}{\lambda},$$

which represents the amount of time needed for the pheromone deposited on each document to half its value since its last update for any given query.

The evaporation rule becomes then

$$\varphi'_{dq} = \varphi_{dq} 2^{-\frac{\tau_{dq}}{\delta}}.$$

Pheromone evaporation is performed periodically and its frequency depends on how the relevance of documents changes over time: since evaporation is a mechanism that enables the system to forget registered behaviors, the more frequent it gets triggered the more newly registered behaviors will be considered important. To the best of our knowledge, the only similar approach is the one by Koychev and Schwab [37].

Finally, the set of documents $D(q)$ — i.e. the results for any query q — are ranked by exploiting the amount of pheronome φ_{dq}, for each $d \in D(q)$.

The Ant Colony Ranking strategy can be viewed as an interplay of the three procedures just described, as summarized by Algorithm 1 [38]: the ranking computed using the pheromone deposited over each document $d \in D(q)$ is prompted to the user issuing the query q (`ShowAntColonyRanking()`), user's clicks get processed and partake in the existing pheromone's configuration (`ManageUserActivity()`), and finally the pheronome evaporation is triggered (`EvaporatePheromone()`).

Algorithm 1 The Ant Colony Ranking strategy in pseudo-code.

```
procedure AntColonyRanking
    scheduledactivities
        ShowAntColonyRanking()
        ManageUserActivity()
        EvaporatePheromone()
    end scheduledactivities
end procedure
```

To summarize, we can now define different Ant Colony Ranking algorithms by specifying (1) how the amount of pheromone φ_{dq} is updated every time a user selects the document $d \in D(q)$ for any query q (i.e. `ManageUserActivity()`), (2) an evaporation strategy (i.e. `EvaporatePheromone()`), and (3) how it exploits the amount of pheromone φ_{dq} to retain the position of a document $d \in D(q)$ in the final ranking presented to the user (i.e. `ShowAntColonyRanking()`).

5. Three Ant Colony Ranking Algorithms

We focus on unsatisfactory search sessions (approximately 50% of all search sessions, according to [39]) and attempt to improve the entire search users' experience. To do that, we proposed a framework for the definition of ranking algorithms that exploit users' interactions with search engines on the basis of the Ant Colony Optimization (ACO) meta-heuristic by defining a pheromone's update rule, how it evaporates over time and the ranking strategy for the set of pages $D(q)$ presented as results for each query q.

In this section, we present three algorithms. The first algorithm is a simple application of the ACO principles to Web pages' ranking, the second algorithm attempts to reinstate the probabilistic nature typical of the ACO meta-heuristic, and the third algorithm is our attempt to leverage on the complete users' search sessions and not just on their single interactions with the search engine.

NaïveRank. The first algorithm is the simplest and most direct implementation of the principles described so far, and is inspired by the stochastic ranking algorithm by Gayo-Avello and Brenes [40]. We employ the simplest incrementing function, namely the successor; thus, given any user search session $DC(q) = \{d_{q1}, d_{q2}, \ldots, d_{qi}$, the pheromone deposited on each document $d_{qi} \in DC(q)$ will be updated with the rule

$$\varphi'_{d_{qi}} = \varphi_{d_{qi}} + 1,$$

where $\varphi'_{d_{qi}}$ indicated the new value after the update. $D(q)$ is ranked decrementally based on the amount of pheromone deposited on each document $d \in D(q)$, thus for any given query q and two documents $d_q^i, d_q^j \in D(q)$ with $i < j$ (recall that d_q^i stands for the document ranked in position i among the results of the query q) we have

$$\varphi_{d_q^i} \geqslant \varphi_{d_q^j}.$$

Despite the resemblance to the algorithm described in [41], NaïveRank runs in real time and, more importantly, naturally takes into account the shifts in users' interests by using the evaporation process.

RandomRank. The second algorithm uses an alternative approach taken from the basic principles of ACO [28], through which we considered search engines' users as agents of a Swarm Intelligence (SI) system. Given our previous algorithm, the probabilistic nature of the original model — which represents one of the ACO strengths — fades out: this phenomenon causes neglecting new paths' discovery once the system reaches convergence. Going back to Web searching, a gradual empowering of the most popular pages' pheromone occurs at the expense of the less popular ones. This effect, known as self-reinforcement, is typical in many techniques of Web ranking [42].

Therefore, we want to encourage the discovery of new pages — in other words, new paths to be explored — by reinstating the probabilistic nature of ACO algorithms into the ranking mechanism, and keeping the update rule the same employed by NaïveRank. Thus, we randomly rank each result in $D(q)$ for any query q with the probabilistic procedure described in Algorithm 2, using a probability distribution based on the quantity of pheromone of each one of them. This way, highly visited pages yield a higher ranking — through a higher probability of selecting one of them in one of the first positions — but less relevant documents still have the opportunity of becoming popular (thanks to the probabilistic nature of the algorithm).

Each cycle in the loop is responsible for the ranking of a document in the set of results: it builds the set $\bar{D}(q)$ of pages that still need to be ranked and randomly picks a document based on its pheromone configuration. This random selection is performed every time a user issues a query q, yielding in a renewed opportunity of discovering new and relevant documents in $D(q)$ and let them gain positions in the ranking.

SessionRank. The last algorithm employs yet another mechanism of the ACO approach. Indeed, hitherto the increment function has always used a fixed amount of pheromones regardless of the click's position among the user's search session. Within the ACO meta-heuristic, in order to achieve convergence quicker, the pheromone's quantity is set to decline on the basis of the quality of the solution found.

On the Web though, users cannot provide a solution to the ranking problem but can assist by providing their own view of relevance. In fact, the first document that a user selects among the results in a given search session is the one that, based on the available clues, is perceived as the most relevant to the user [44]. The most relevant document according to a user is the one that should be in a highly relevant position in the optimal solution to

Algorithm 2 Procedure ranking $D(q)$ used by RandomRank based on [43].

procedure ShowAntColonyRanking
  for $i \leftarrow 1, \#D(q)$ **do** ▷ Rank one result at a time
    $\bar{D}(q) \equiv \{d : d \in D(q) \wedge d \neq d_q^j, 1 \leqslant j < i\}$ ▷ Not yet ranked
    Select $d \in \bar{D}_q$ with probability $\frac{\varphi_{dq}}{\sum_{\bar{d} \in \bar{D}(q)} \varphi_{\bar{d}q}}$ and rank it in position i
  end for
end procedure

the ranking problem for that given query. The next document, selected in the same session, is considered less relevant since it was selected after the previous document.

The SessionRank algorithm employs the relative order of clicks performed by each user during a session and increments the pheromone's quantity accordingly. Therefore, choosing an exponential decay, the update rule becomes:

$$\varphi'_{dq} = \varphi_{dq} + 2^i,$$

where $d \equiv d_{qi}$ and $d_{qi} \in DC(q)$ for any user search session $DC(q)$ yielding $D(q)$.

To summarize, the model proposed in the previous section to define ranking algorithms on the basis of ACO employs pheromone traces on each document in relation to any query issued to the search engine; the pheromone increases each time a user selects a page among the results of a query and vaporizes over time taking into account users' gradual loss of interest. Therefore, once a user performs a query already performed by others, the search engine is able to present a new ranking based on the behavior shown by users with the same informational need, de-facto exploiting pheromone traces.

We described three different algorithms that, by exploiting the aforementioned model, use different ACO-inspired mechanisms to improve the proposed ranking. Thus, since establishing whether improving search engine performance is truly possible by employing this new approach is important, we devised an evaluation of the different algorithms using real query-click logs. In the following section, we present details on measures, setups, and results.

6. Evaluation

6.1. Search Engine Evaluation

In an ideal situation, an Information Retrieval (IR) system (for example, a search engine) should only provide relevant results for the issued queries. The tendency is to accept that such systems provide the widest set of relevant documents, along with some less relevant results. Normally, evaluating an IR system requires experimental sets containing queries, documents, and relevance judgments; however, building such collections requires a significant amount of work (in other words, data on queries and judgments). Thus, in many recent studies [45, 46, 40, 47, 5], click-through data were employed to evaluate search engines' performance. The concept is simple: employ clicks as relevance judgments, assuming that a user evaluates a result as relevant if it is chosen among the search results related to a query.

Consequently, in the following experiments, we employed query-click log datasets provided by two famous search engines — AOL [48] and Yahoo! — to carry out experiments on the proposed algorithms (further details on these datasets can be found in the Appendix); we have validated the new ranking produced by each algorithm using the very same datasets clustered by each user's search session, applying a simple temporal threshold (30 minutes, as suggested by several previous studies [6, 49]) to decide whether two actions performed by a single user belong to the same search session.

Hence, we considered two interactions as belonging to the same search session when they were both (1) issued by the same user, (2) contained the same query, and (3) performed within 30 minutes.

After selecting the two datasets, we needed a performance measure to evaluate all the different algorithms we propose.

Sakai [50] compared different performance measures that take into account the documents' positions and recommended the so-called Normalized Discounted Cumulative Gain (NDCG) for its simplicity and robustness [51].

The NDCG consists of a parameter and two functions: 1. $k \in \mathbb{N}_0$ is a cut-off parameter defining the number of elements in the results list to be considered; 2. the gain function measures the benefit earned by the user (if a document is only relevant or irrelevant, then the gain is binary), whereas

3. the discount function weighs the documents' relevance on the basis of the position in the results list.

Moreover, to obtain an absolute measure — in the real interval [0,1] — we normalized it with respect to the maximum obtainable gain.

More formally, let $\vec{y} \in \mathbb{R}^n$ be an array containing relevance values belonging to a sequence of n elements (for example, the results of a query) and let $\vec{\pi} \in \mathbb{R}^n$ be a permutation of the same sequence (for example, the ranking produced by an algorithm). Let $\vec{\pi}(q)$ be the index of the q-th element in $\vec{\pi}$ and let $\vec{y}_{\vec{\pi}(q)}$ be the value of its relevance. The Discounted Cumulative Gain (DCG) of the permutation $\vec{\pi}$ is defined as:

$$\mathrm{DCG@}k(\vec{y}, \vec{\pi}) = \sum_{q=1}^{k} \frac{2^{\vec{y}_{\vec{\pi}(q)}} - 1}{\log_2(2+q)}.$$

In this case, the gain function is a power of 2, whereas the discount function has logarithmic decay over the permutation length. Thus, the NDCG is defined as:

$$\mathrm{NDCG@}k(\vec{y}, \vec{\pi}) = \frac{\mathrm{DCG@}k(\vec{y}, \vec{\pi})}{\mathrm{DCG@}k(\vec{y}, \vec{\pi}^*_{\vec{y}})},$$

where $\vec{\pi}^*_{\vec{y}}$ is the permutation corresponding to a perfect ranking w.r.t. the relevance judgments in $\vec{y}$ or, in other words:

$$\vec{\pi}^*_{\vec{y}} = \operatorname*{argmax}_{\vec{\pi}} \mathrm{DCG@}k(\vec{y}, \vec{\pi}).$$

Few publicly available datasets already provide explicit relevance judgements for each document they refer: the Yahoo! dataset is one of them, but the AOL one does not; as we state before though, user clicks can be used as a relative measure of the perceived relevance by each user. In this case, we then build the array $\vec{y}$ by assigning 1 to each document selected by the user in the search session we are currently trying to compute the NDCG for, 0 for the ones that were not selected.

To summarise, computing NDCG scores for each search session contained in the datasets is possible using the available relevance judgments for Yahoo! and by assuming that the visited results are the relevant ones for AOL. Effectively, by doing so we are actually evaluating the current performance of the two search engines, which will be the baseline we are comparing our algorithms against.

6.2. Evaluation of Ant Colony Ranking Algorithms

We described hitherto how search engine performance are evaluated by employing the query-click log containing users' interactions. As we previously stated, the proposed algorithms require queries, clicks, and search sessions to adjust the pheromone deposited on each document in relation to any query submitted to the search engine. The chosen query-click logs contain all this information and, thus, can be employed to simulate our algorithms in a real world scenario.

The validation strategy is dictated by the constraints over our context: since we are using these datasets to simulate real users' interactions, we have to obey to time constraints; our test set will then always be consequent to the training set, since the system can only be trained using past interactions, forbidding any kind of cross-validation. According to the most common strategy, we chose then to split our data and use the first two thirds for training and the remaining for testing: moreover, using one third of the available interactions provides a significant amount of data to thoroughly test our algorithms.

Therefore, we prepared two partitions for each available query-click log: the AOL partition includes the set of the almost 20 million clicks performed from March to April 2006 and was used for training, whereas the remaining set of approximately 10 million clicks performed in May 2006 was used for the evaluation. The Yahoo! training set contained almost 40 million clicks performed during the first 20 days of July 2010 (excluded), whereas the evaluation set contained the remaining 26 million clicks issued until the end of the same month.

After the training phase, we compared the search sessions contained in the test sets with the ranking given by each algorithm and devised a sequence of potential clicks; this procedure is ruled by a simple and reasonable assumption [40]: during a search session, if a user chooses a result for a given query, we safely assume that the same user in the same search session would have chosen that same result even if it was found in a higher position in the results' list. Finally, we computed the mean of NDCG for each session.

Furthermore, because we sought to evaluate how the algorithms parameters affect performance, we tested three different values of δ and of the evaporation time (one hour, one day, and one week), and three session duration values for the SessionRank training (one, five, and 25 minutes, namely very brief, average — according to [7] — and long-duration search sessions); besides, since RandomRank is a probabilistic algorithm, each experiment were repeated 5 times using different seeds of the random number generator.. Thus, the evaluation involved 162 different experiments. We carried them out using a t2.small Amazon EC2 instance running Amazon Linux AMI and the DEX library to manipulate both datasets [52]; each run took about 3 hours to complete.

To summarize, our evaluation's aim was to measure and compare the proposed ranking algorithms' performance using data provided by two famous search engines. Because our methods are based on users' interactions to discover the most promising results, the datasets were partitioned into a training set and a test set. In the next sections, we provide the results obtained from using this evaluation method.

6.3. Results

As previously stated, we defined a framework for the evaluation of the three proposed algorithms performance using the interactions from the two different query-click logs. We described (1) the way we selected the training and test sets, (2) how we used the former to simulate real users' behavior, and (3) from the latter we yielded the potential clicks combining new rankings with the original click-through data.

In the following, we analyze the results of the performed experiments, reporting NDCG scores for three different cutoff values: 1, 3 and 10 results (NDCG@1, NDCG@3, NDCG@10). We recall that NDCG is a measure in the interval [0,1] where 1 is the ideal ranking. We chose to test our algorithms on three different cutoff values meaning respectively: the result ranked first with NDCG measure representing the ratio between our algorithm result and the first best ranked from the training set; the three highest ranked results and the ten highest results. We can imagine comparing the results within a search engine result page in which only first result is returned, or the first three or the first ten. We evaluated those results for all three proposed algorithms. Graphs showed in the following figures are organized as follows: on the x-axis the (δ) factor is shown representing the timeframe set for halving the pheromone (representing evaporation); the wider the timeframe the longer will take to half the pheromone and thus results appearing in the ranking will be more persistent: it, basically, represents the magnitude of the evaporation set to 0.5 (delta factor in section 4.1). On the y-axis on the left we show the NDCG values [0,1] related to the corresponding pheromone upgrading timeframe (upgrades are run hourly, daily and weekly) showed on the right (y-axis), while on the top (x-axis on top) the different NDCG cut-offs are shown. Finally, the black segments represent our benchmark, that is the NDCG values computed respectively on the Yahoo! and AOL datasets by maintaining the default search engine's ranking.

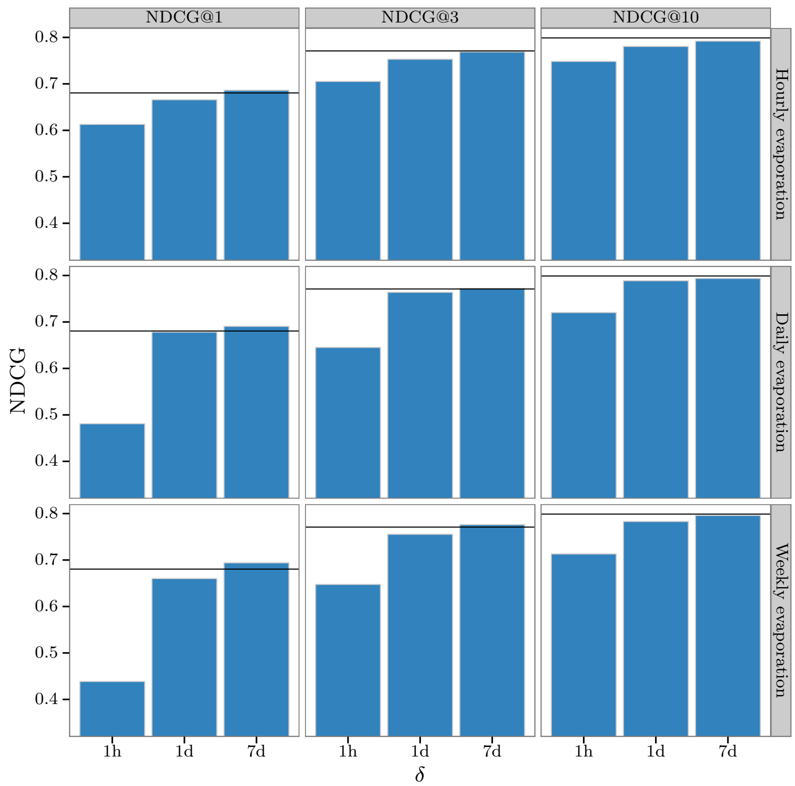

Figure 1: NaïveRank results related to the AOL Dataset. The x-axis represents the timeframe set for halving the pheromone, the left y-axis the NDCG values related to the corresponding pheromone upgrading timeframe on the right y-axis, while on the top x-axis the different NDCG cutoffs are shown. The black segment indicates the benchmark, namely the result given by the default AOL's ranking.

Figures 1 and 2 show respectively the performance given by the NaïveRank algorithm by using AOL and Yahoo! dataset. As expected, our algorithm performs better on the larger dataset (i.e. Yahoo! in figure 2); this seems reasonable, since it follows from the Ant Colony Optimization (ACO) approach: the more data is recorded about users' behavior the better the algorithm will perform. More surprising are the differences obtained using different

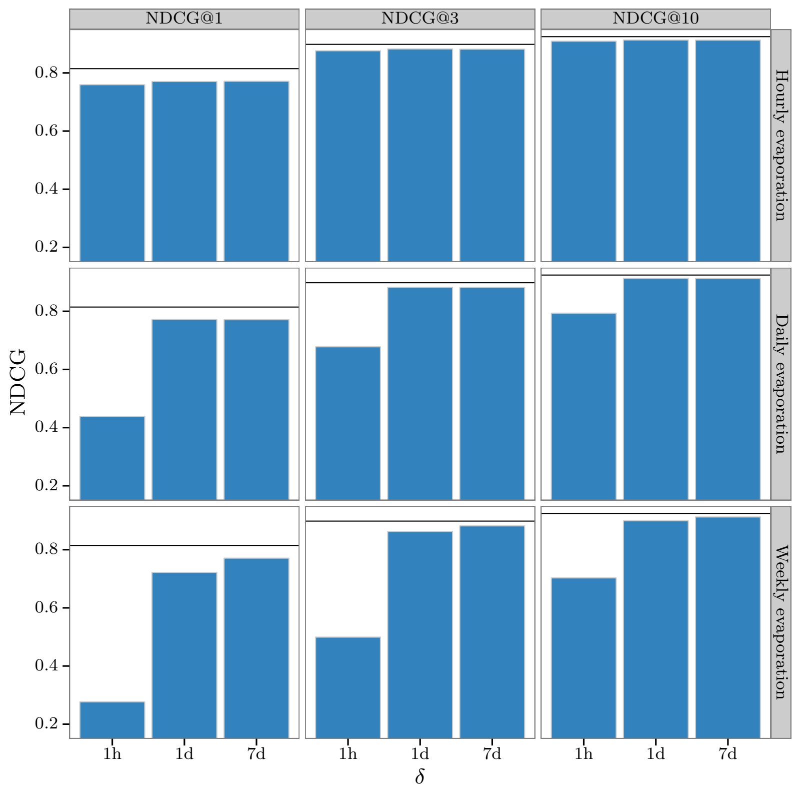

Figure 2: NaïveRank results related to the Yahoo! Dataset. The x-axis represents the timeframe set for halving the pheromone, the left y-axis the NDCG values related to the corresponding pheromone upgrading timeframe on the right y-axis, while on the top x-axis the different NDCG cutoffs are shown. The black segment indicates the benchmark, namely the result given by the default Yahoo!'s ranking.

pheromone evaporation intervals and by halving (δ) times. Normally, one would expect that depending on the halving delta factor timeframe the algorithm would perform better in adapting to users' behavioral changes: in fact, no matter which dataset we used, we obtained the best performance by setting $\delta = 7d$ (a week timeframe); this confirms what implied by Liu et al. [53] about the weekly cycle of the majority of queries which states that users will perceive search engine results as relevant and up to date generally only during one week after which they would expect the results to be updated.

Regarding the evaporation time, we noted an interesting effect: although using non-optimal δs (as stated above, we found out that seven days was the optimum) doesn't affect the performance, choosing an optimal δ makes slightly no difference at all.

Thus, when implementing the NaïveRank we can safely act on δ to reduce the evaporation frequency, in order to reduce the amount of computations needed. This implies that the algorithm will be more computationally efficient in real time. Effectively, increasing the evaporation frequency will affect the computational cost of the algorithm since the upgrade denoted by the evaporation rule in section 4 has to be computed less frequently.

Consequently, just by observing the first results we can argue that the data size required to get good performances from the algorithm is substantial. While the evaporation time is important to achieve good performance, using a δ set to the weekly cycle of queries allowed us to arbitrarily choose the evaporation time, significantly reducing the computational costs.

Considering figures 3 and 4 which report on the RandomRank performance, it is interesting to notice how it differentiates from NaïveRank: the variations take place mostly for NDCG@1 (i.e. the score related only to the first displayed result), while for NDCG@3 and NDCG@10 results are almost identical to NaïveRank. This is due to the probabilistic nature of the random ranking; indeed, probability has on average an higher effect when smaller set of documents are considered due to probability of selecting more than one element of the set for NDCG@3 and NDCG@10.

A slightly more interesting result comes from analyzing how such variations actually occur: when tested against the AOL dataset — the smaller one — RandomRank underperforms NaïveRank by about 10%. For Yahoo! Dataset, performances are almost the same for both the algorithms. The exception is the configuration; with $\delta = 1h$ (i.e. the halving factor) and a weekly evaporation cycle. Then RandomRank doubles the score obtained with the same configuration by NaïveRank. This could confirm once more what we stated previously when discussing NaïveRank's results about the weekly cycle of search queries, as introduced by Liu et al. [53]: the introduction of probability helps users discovering new interesting documents among results. In this particular case the pheromone updates on weekly basis according to users' perception of documents relevance.

Finally, figures 5 and 6 show the results related of the SessionRank algorithm; these graphs — apart from the evaporation frequency on the right y-axis, the halving factor in the x-axis and the segments representing the benchmarks as for the previous graphs — show the different timeframes used by the algorithm to identify a user's search session in its training phase. We

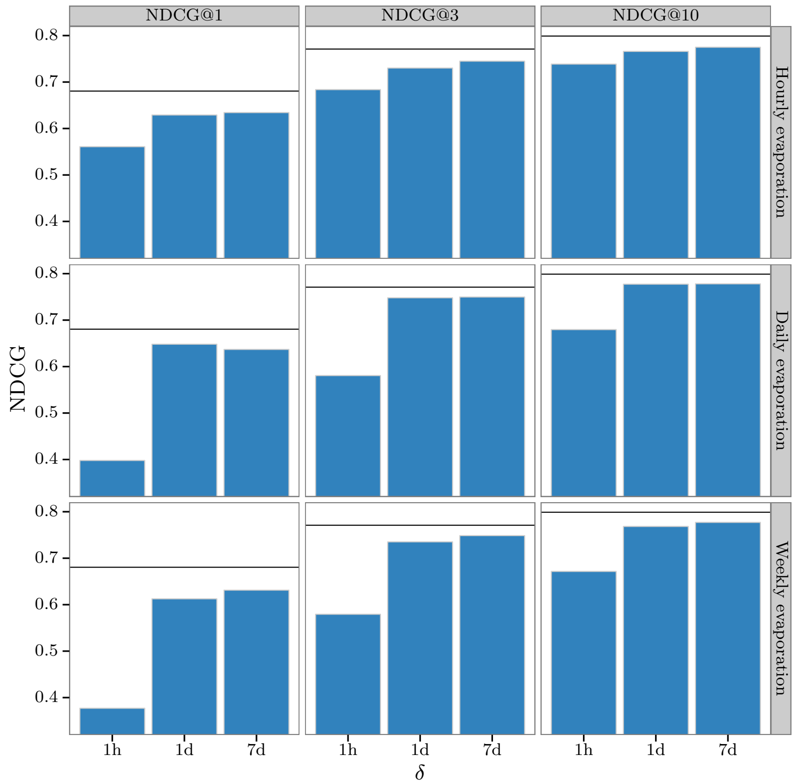

Figure 3: RandomRank results related to the AOL Dataset. The x-axis represents the timeframe set for halving the pheromone, the left y-axis the NDCG values related to the corresponding pheromone upgrading timeframe on the right y-axis, while on the top x-axis the different NDCG cutoffs are shown. The black segment indicates the benchmark, namely the result given by the default AOL's ranking.

recall that based on the session's duration, the algorithm distributes differently the quantity of pheromone to be deposited on a document once; by performing multiple experiments, each time with a different session's duration, the algorithm will consider a sequence of interactions performed by the same user as a single session if they were all done within the allowed timeframe, otherwise it will split them into multiple sessions. We chose to test three configurations for the sessions duration used by the algorithm (as reported in different colors in the figures) namely 1 minute, 5 minutes and 25 minutes. We selected these three configurations in order to evaluate the different performance of our algorithms with very brief search sessions, i.e.

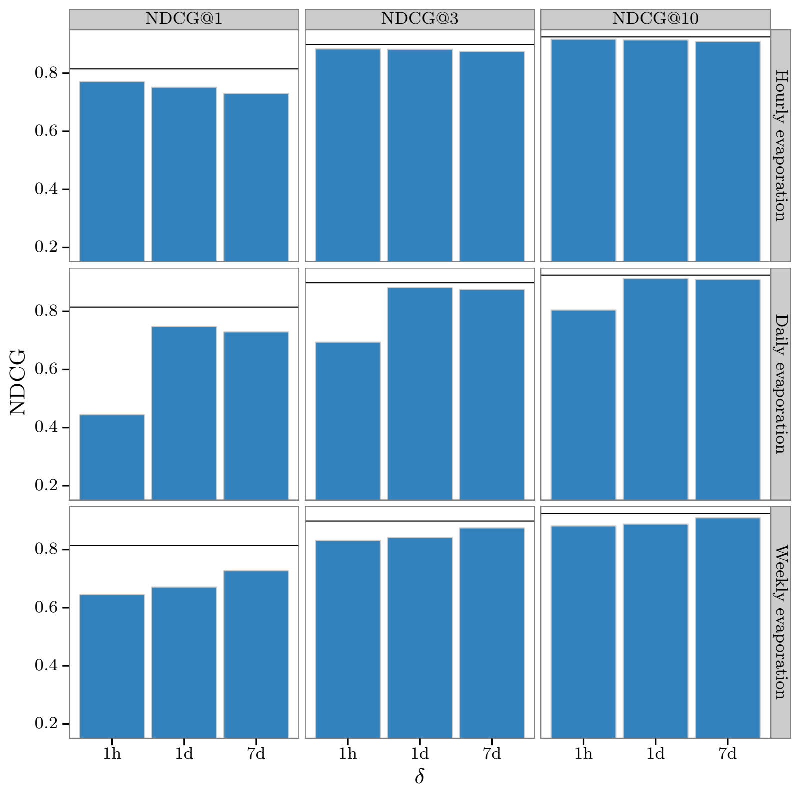

Figure 4: RandomRank results related to the Yahoo! Dataset. The x-axis represents the timeframe set for halving the pheromone, the left y-axis the NDCG values related to the corresponding pheromone upgrading timeframe on the right y-axis, while on the top x-axis the different NDCG cutoffs are shown. The black segment indicates the benchmark, namely the result given by the default Yahoo!'s ranking.

1 minute which is shorter than the average search session — 3 minutes according to [7]; we also tested our algorithms with 5 minutes sessions that can be considered as of average duration and finally we chose to also include 25 minutes sessions to sample longer durations.

One can notice straightaway that the training sessions duration is mostly irrelevant; this may be caused by the average short length of search sessions, as demonstrated in [54], since users tend to perform brief sessions, performing different queries; thus search sessions will have only few clicks performed in a short time span and the amount of pheromone to be deposited by the algorithm will be rather fixed, causing no effect.

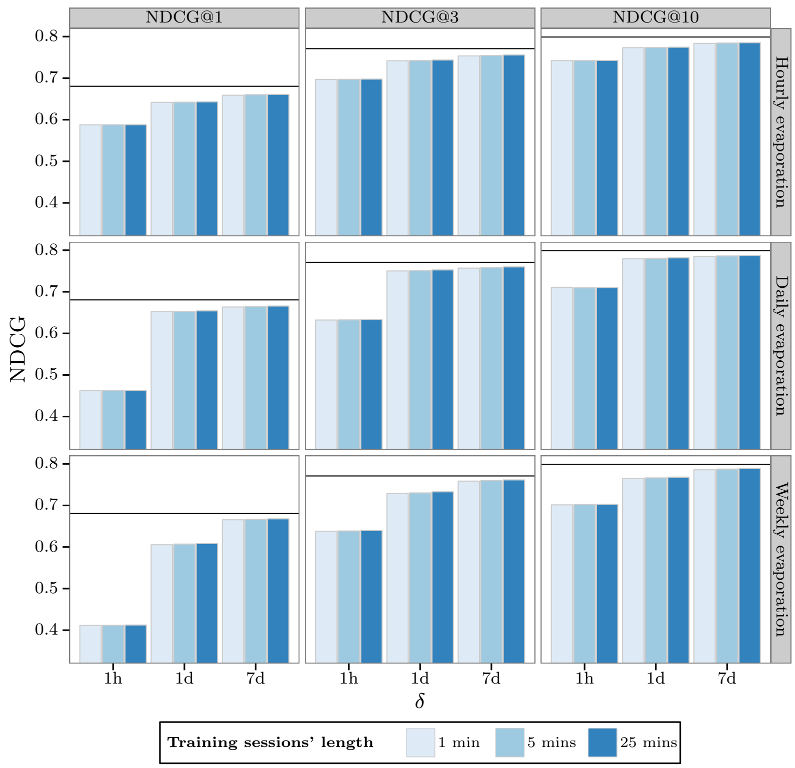

Figure 5: SessionRank results related to the AOL Dataset. The x-axis represents the timeframe set for halving the pheromone, the left y-axis the NDCG values related to the corresponding pheromone upgrading timeframe on the right y-axis, while on the top x-axis the different NDCG cutoffs are shown; the different colors represent the timeframes used to identify a user's search session in the training phase. The black segment indicates the benchmark, namely the result given by the default AOL's ranking.

Also, the algorithm performs worse than NaïveRank using the first dataset, while outperforming it with the second one; this is still due to the variation in training set size: a greater number of search sessions used in training causes an improvement due to the greater quantity of pheromone available to be deposited by each user; thus, the pheromone's modulation — inspired by the ACO metaheuristic — improves the algorithm performance proportionally to the number of interactions recorded during the training phase.

Figure 7 recaps our results: the two plots represent only the *best* performance obtained by the different configurations of our algorithms — showed on the x-axis — with the two datasets used; on the y-axis we show the dif-

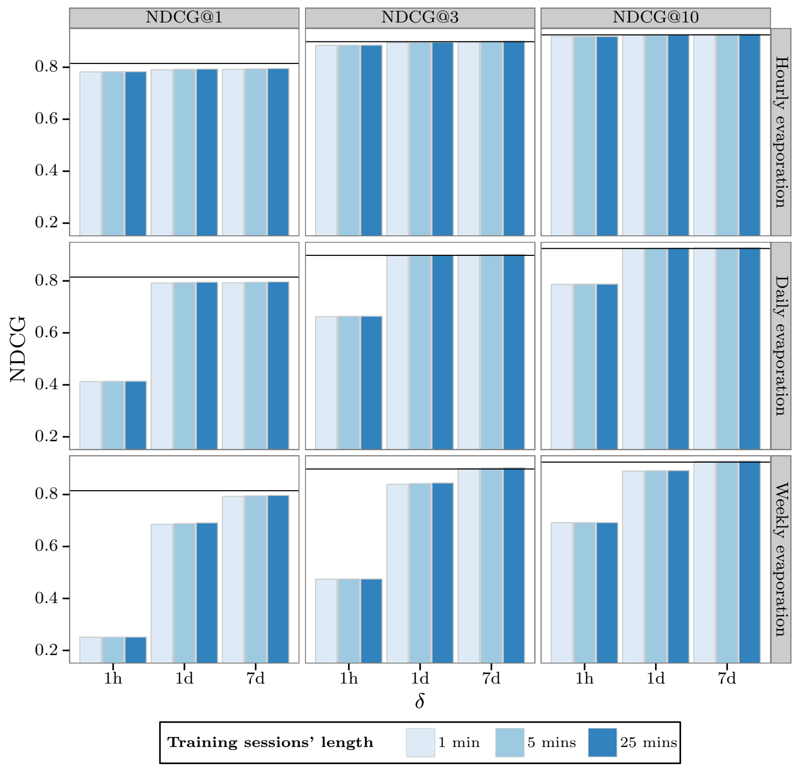

Figure 6: SessionRank results related to the Yahoo! Dataset. The x-axis represents the timeframe set for halving the pheromone, the left y-axis the NDCG values related to the corresponding pheromone upgrading timeframe on the right y-axis, while on the top x-axis the different NDCG cutoffs are shown; the different colors represent the timeframes used to identify a user's search session in the training phase. The black segment indicates the benchmark, namely the result given by the default Yahoo!'s ranking.

ferent cutoff points used to compute the related NDCG measure, and the size of the points show the actual NDCG value we obtained, bigger for large values. Finally, the green color is used to show a NDCG measure which improves over the baseline ranking algorithm applied by the search engine, red if otherwise.

Summarizing these findings, we argue that our first two proposed algorithms could be employed in improving results ranking produced by two particular sets of queries: being a simple application of the ACO technique to Web pages ranking, NaïveRank works well for embedding a plain concept of popularity into the ranking measure. Thus it could be very effective in

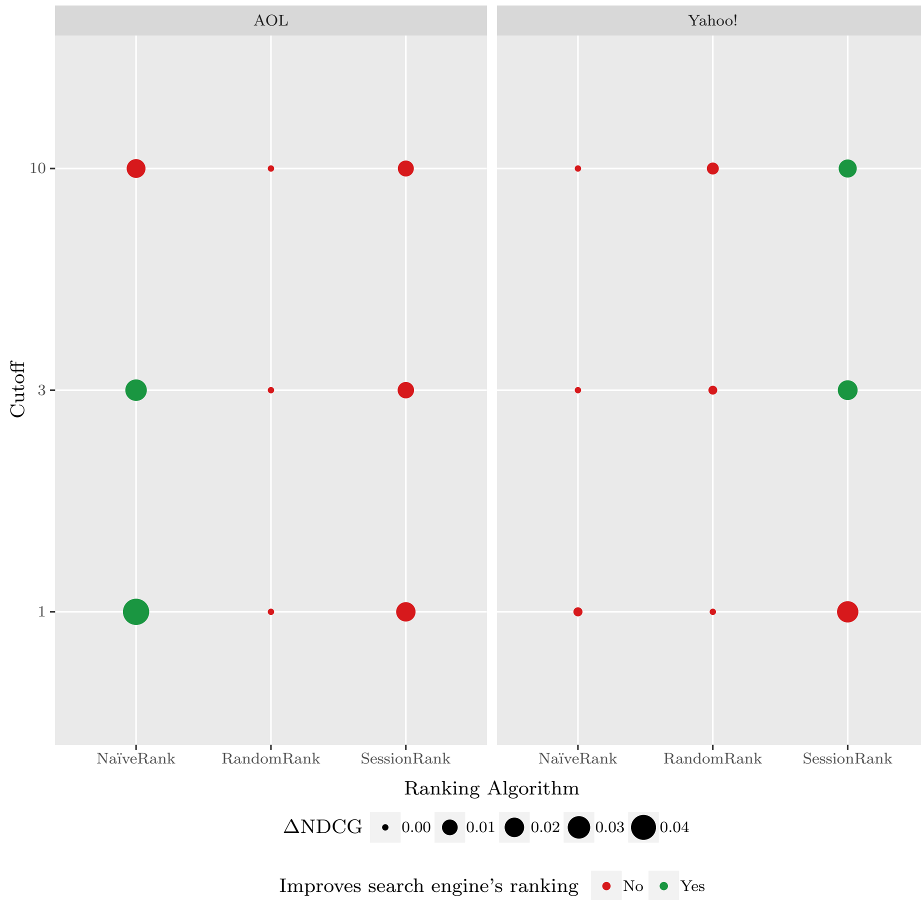

Figure 7: Summary of the three algorithms' best performance: the x-axis shows the algorithm performance related to each dataset we used (on the top x-axis), while the y-axis shows the cutoff point for the corresponding NDCG measure; the size of the points represents the related NDCG's goodness, while the colors indicates whether we achieved an improvements over the default ranking operated by the search engines.

ranking results related to transactional or informative queries whose results do not become obsolete frequently (i.e. it is rare to see a new document containing updated information suddenly appear, making the already popular ones out-of-date, e.g. encyclopedia definitions or catalog's products).

By reinstating the probabilistic nature typical of the ACO metaheuristic, RandomRank allows new and bleeding-edge documents to be discovered by users, thus it could be very effective in ranking results related to breaking news and current events.

Finally, SessionRank shifts on a whole new dimension in terms of the kind of information it exploits and — thus — the search settings that could benefit from its introduction in the ranking mechanism. Albeit the majority of search sessions are brief, composed by just one query and focused only on the first results page, there are some particular sessions that might be longer and could be very frustrating for users. We noticed three types of such problematic sessions in our datasets:

atypical Web search sessions [55] are being produced by users with atypical information needs, i.e. those outside their regular areas of expertise (often triggered by external events, such as pending medical treatments, financial deadlines or upcoming vacations);

exploring sessions [56] are those where users are engaged in an open-ended and multi-faceted information-seeking task to foster learning and discovery;

struggling sessions [56] are those where users are experiencing difficulty locating the required information.

Examples of both exploring and struggling sessions can be found in figure 8.

Given that SessionRank exploits not just single interactions with the search engine but whole search sessions, it seems reasonable to argue that the ranking of results related to the aforementioned search sessions could benefit from the introduction of our last ACO-inspired algorithm. Thus it could be useful to test our algorithm against a different dataset containing long-lasting search sessions of this type. Alternatively, one could use some query-similarity measure with the same datasets we employed, in order to cluster similar queries belonging to the same session.

Our findings are summarized in figure 9: it shows both dimensions that our algorithms operate on: search sessions' length on the x-axis and documents update frequency on the y-axis. The horizontal stripes represent the aforementioned examples of documents sets to be ranked, such as breaking news, encyclopedia definitions and catalog's products, while the vertical ones are the three kinds of search sessions outlined in the previous paragraph [55, 56]. The dashed blocks indicate which algorithm we think could be the most effective in ranking results generated for each case. For example — as

Query can you use h & r block software for more than one year
Query how do I file 2012 taxes on hr block
Click `http://www.hrblock.com`
Query can you only use h & r block one year
Click `http://www.www.consumeaffairs.com/finance/hr_block_free.html`
Click `http://financialsoft.about.com/od/taxcut/gr/HR-Block-At-Home-...`
Query do I have to buy new tax software every year
Click `http://financialsoft.about.com/od/simpletips/f/upgrade_yearly.htm...`
Click `http://askville.amazon.com/buy-version-Tax-Software-year/Answer...`
END OF SESSION

(a) A *struggling* session.

Query career development advice
Click `http://www.sooperarticles.com/business-articles/career-devel...`
Query employment issues articles
Click `http://jobseekeradvice.com/category/employment-issues/...`
Query professional career advice
Click `http://ezinearticles.com/?Career-Advice-and-Professional-Ment...`
Click `http://askville.amazon.com/buy-version-Tax-Software-year/Answer...`
Query what is a resume
Click `http://en.wikipedia.org/wiki/R%C3%A9sum%C3%A9...`
END OF SESSION

(b) An *exploring* session.

Figure 8: Examples of struggling and exploring sessions taken from [56].

we stated previously — RandomRank could be beneficial in ranking results among frequently updated documents and does not really take into account any information about the whole search sessions (focusing only on single interactions), thus it won't work for longer sessions, such as atypical, struggling or exploring ones for which SessionRank could be more suitable. Ranking more static collections of documents, such as products inside a catalog for example (e.g. Amazon or Google Shopping) or encyclopedia entries doesn't require a very refined collaborative filtering mechanism — due to the low frequency of updates — thus NaïveRank could fit well with these situations. Indeed, catalogs can be dynamic but vary less frequently than news.

7. Conclusion and Future Work

We presented an approach to developing real time implicit collaborative information-seeking algorithms. Providing implicit collaboration is becoming increasingly relevant in search engine research and application areas. Recently, Google introduced their Social Search service, declaring that, "with these changes, we want to help you find the most relevant information from

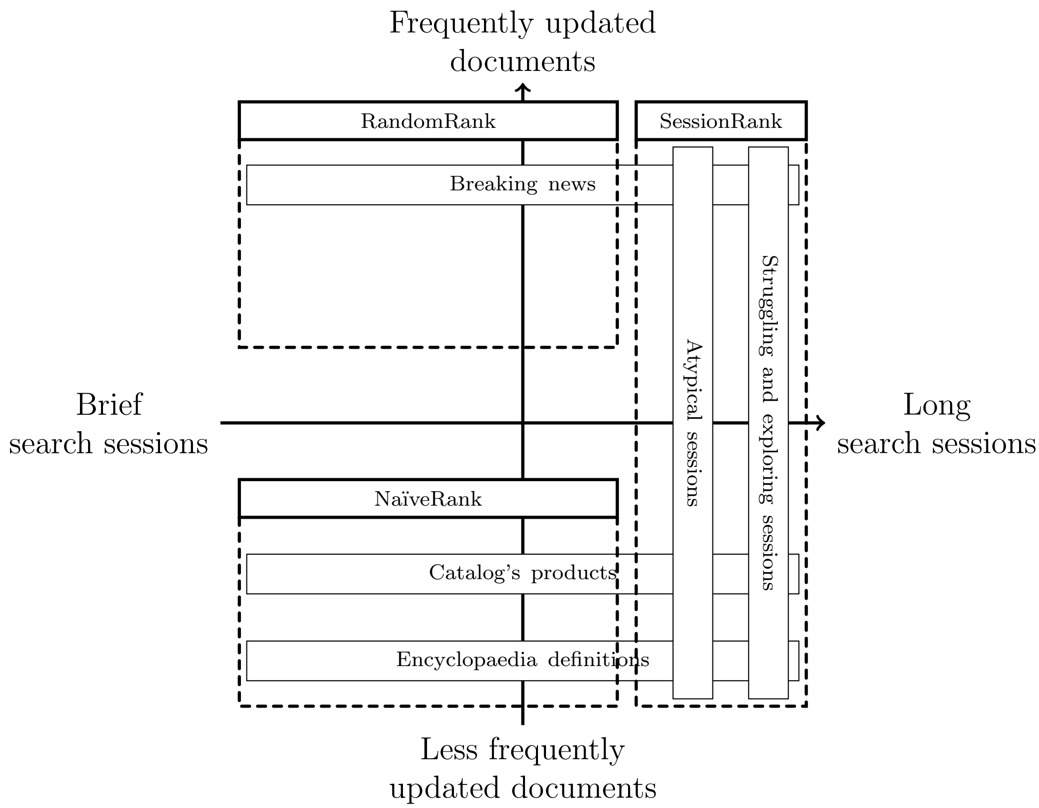

Figure 9: A plot of our findings: on the x-axis we have the search sessions' length, while on y-axis the documents update frequency; vertical and horizontal stripes represent examples of both those settings, while colored blocks indicate the most suitable algorithm to rank results generated by the revealed search setting.

the people who matter to you". In a way, that statement represents our definition of a colony. The mechanism is the Google+1 button, which allows users to share interesting pages with their contacts — a way to release pheromones. Bing, Microsoft's new search engine, employs Facebook's social graph for each user to rank search results and to present search history. That is, they define the colony as our own Facebook contacts. Again, we deposit pheromones through a click on the "like" button. This method is viewed as a way to implement pheromone evaporation. However, these stylish interactions can be modelled by ants. As ants, we leave "pheromones" to allow others to follow our trails. Additionally, as ants, we use this information to enhance searching.

We designed three different algorithms employing an Ant Colony Optimization (ACO) strategy to provide implicit collaborative-seeking features in real time to search engines. The three different algorithms — NaïveRank, RandomRank, and SessionRank — all proved to be effective in real time depending on the nature of the queries submitted by users. Real time performance is crucial for search engines, particularly when using ACO-inspired algorithms for which a large graph of queries and documents might be created.

The NaïveRank seems particularly interesting for informational queries that seek to retrieve results on relatively static information on the Web, such as looking for products in a catalog or encyclopedia entries. RandomRank proved effective for the inverse situation, such as breaking news or a sports event. The SessionRank algorithm was suited for struggling and explorative sessions (in other words, open-ended information-seeking tasks fostering users' learning) or atypical query sessions (generated by external events such as specific treatments, deadlines, or upcoming holidays).

We evaluated the three algorithms by designing an evaluation, where we compared the performance of the three proposed ranking algorithms with the data provided by two famous search engines: Yahoo! and AOL. Because our methods are based on users' interactions to discover the most promising results, the datasets were partitioned into a training set and a test set.

We plan to run an online experiment with a wide sample of participants and test the three algorithms in a real time scenario with users in the future. We hope to prove that in an online environment, real time relevant results can also be obtained by users employing an implicit collaborative approach for information seeking and by selecting the right algorithm depending on the types of queries. We also plan on further investigate how blend some concepts explored by existing ACO extensions in our model, such as limiting the amount of deposited pheromone to avoid deposited as in Max-Min Ant System (MMAS) [29].

Acknowledgment

This work has been partially supported by ANTASTIC — a bio-inspired approach to social search interactions. YGGDRASIL: the Research Council of Norway Grants for highly qualified, younger researchers (2013). Project n. 220050/F11. We would like to acknowledge the Research Visibility Award funded by Brunel University London that allowed the main author to establish a research network on collaborative information seeking with the co-authors of this work.

Appendix A. AOL query-click log

This archive, released in August 2006, contains more than 30 million clicks issued by 650000 users, recorded in a timespan going from March to May 2006; each record (table A.1, from left to right) is made up by the user ID, the query, the timestamp, the document's position among the results, and the URL. The position 0 illustrates that the user issued the query but didn't click on any result at all.

We didn't employ the whole log in our evaluation, instead we only considered the subset of queries issued on average once a day during the observed period (i.e. March to May 2006): this process allowed us to only employ "significant" interactions with the search engine, ignoring the ones issued less frequently without biasing our later evaluation. By doing so, we obtained about 5 million different clicks related to 22000 different queries.

285103	ants	2006-04-01 19:45:23	1	http://www.dna.affrc.go.jp
285103	ants	2006-04-01 19:45:23	3	http://www.uky.edu
285103	ants	2006-04-01 19:50:59	13	http://ohioline.osu.edu
285103	ants	2006-04-01 19:50:59	14	http://ohioline.osu.edu
285103	ants	2006-04-11 21:44:45	7	http://ohioline.osu.edu
889138	ants	2006-03-05 13:22:31	4	http://www.ants.com
889138	ants	2006-03-05 13:22:31	8	http://ohioline.osu.edu
889138	ants	2006-03-05 13:26:14	11	http://www.infowest.com
889138	ants	2006-03-05 13:26:14	19	http://www.greensmiths.com
3519280	ants	2006-03-30 17:14:14	0	
3519280	ants	2006-03-30 17:15:53	1	http://ant.edb.miyakyo-u.ac.jp
3519280	ants	2006-03-30 17:15:53	3	http://www.uky.edu
3519280	ants	2006-03-30 17:15:53	10	http://en.wikipedia.org
3519280	ants	2006-03-30 17:27:46	0	
3519280	ants	2006-04-01 13:55:03	2	http://www.lingolex.com
3519280	ants	2006-04-01 13:55:03	3	http://www.uky.edu
3519280	ants	2006-04-01 14:20:53	0	

Table A.1: AOL Query-click log fragment for the query 'ants'. Horizontal lines separate users by "user ID".

After selecting the database to be used in our experiments, we detected the actions' sequence (i.e. clicks) performed by each user during each search session. Therefore, we applied a simple temporal threshold: if two actions were performed within a 30 minutes' timespan then they would belong to the same session.

In tables A.1 and A.2 we can observe the sessions' detection process in the original query-click log.

After a controversial discussion about the users' privacy following the initial public release, AOL chose to remove the log from its servers and doesn't offer the download anymore, although it's still available to researchers.

285103	ants	2006-04-01 19:45:23	1	http://www.dna.affrc.go.jp
285103	ants	2006-04-01 19:45:23	3	http://www.uky.edu
285103	ants	2006-04-01 19:50:59	13	http://ohioline.osu.edu
285103	ants	2006-04-01 19:50:59	14	http://ohioline.osu.edu
285103	ants	2006-04-11 21:44:45	7	http://ohioline.osu.edu
889138	ants	2006-03-05 13:22:31	4	http://www.ants.com
889138	ants	2006-03-05 13:22:31	8	http://ohioline.osu.edu
889138	ants	2006-03-05 13:26:14	11	http://www.infowest.com
889138	ants	2006-03-05 13:26:14	19	http://www.greensmiths.com
3519280	ants	2006-03-30 17:14:14	0	
3519280	ants	2006-03-30 17:15:53	1	http://ant.edb.miyakyo-u.ac.jp
3519280	ants	2006-03-30 17:15:53	3	http://www.uky.edu
3519280	ants	2006-03-30 17:15:53	10	http://en.wikipedia.org
3519280	ants	2006-03-30 17:27:46	0	
3519280	ants	2006-04-01 13:55:03	2	http://www.lingolex.com
3519280	ants	2006-04-01 13:55:03	3	http://www.uky.edu
3519280	ants	2006-04-01 14:20:53	0	

Table A.2: The interactions depicted in table A.1 grouped in 30-minutes long sessions.

Appendix B. Yahoo! query-click log

Yahoo!'s dataset contains only anonymous information due to the same privacy issues experienced by AOL. It includes 66 million clicks recorded in July 2010 and relevance judgments of 650 thousand Web pages issued by experts between 2009 and 2010 related to some of the logged queries are also available. Each record, contains the interactions related to a single page results of each user, and is made up by `query cookie timestamp url_1 ...url_10 nc et_1 pos_1 ...et_nc pos_nc`, where

`query` is the anonymized version of the query, `cookie` is the anonymized version of the user's cookie,

`timestamp` is Unix time (the amount of seconds passed since 1 January 1970) of the issued query,

`url` is the anonymized version of the URL,

`nc` is the number of clicks performed during the entire session,

`et` is the time passed between each click and the beginning of the session,

`pos` is the position of each click, which could be:

- `1 ...10` one of the 10 results,
- `0` above the first result (spelling corrections, header advert, etc.),
- `11` below the last result (next page, footer advert, etc.),
- `s` new query,
- `o` other clicks.

As for the previous dataset, we employed in our experiments the subset of records related to the queries performed on average once a day during the reference timespan and for which we have the relevance judgments. This way, we obtained about 65 million different clicks related to almost 44500 queries, then grouped in 30 minutes long sessions, as the AOL query log; in table B.3 there is a fragment of the log.

`00002efd 1deac14e 1279486689 2722a07f 24f6d649 1b2b5a1c 9ca4edf1 23045132 84c0d8b5 de33d1de 9f5855b2 477aabf6 e1468bbf 3 10 1 175 o 215 0`
`00002efd 3fef0ac3 12799559361 2722a07f 8f59fce1 de33d1de a2c8d464 57a7dd83 a11dbd14 08b5c87e 44a77e61 c21b6dbe 6b0a7915 1 2 0`

Table B.3: Yahoo! query-click log.

References

[1] M. A. Hearst, What's Missing from Collaborative Search?, Computer 47 (2014) 58–61.

[2] G. Golovchinsky, J. Pickens, M. Back, A Taxonomy of Collaboration in Online Information Seeking, CoRR abs/0908.0704 cs.IR (2009).

[3] Z. Yue, S. Han, D. He, J. Jiang, Influences on Query Reformulation in Collaborative Web Search., Computer 47 (2014) 46–53.

[4] A. Broder, A taxonomy of web search, ACM SIGIR Forum 36 (2002) 3.

[5] A. M. Zareh Bidoki, P. Ghodsnia, N. Yazdani, F. Oroumchian, A3CRank: An adaptive ranking method based on connectivity, content and click-through data, Information Processing & Management 46 (2010) 159–169.

[6] S. Fox, K. Karnawat, M. Mydland, S. Dumais, T. White, Evaluating implicit measures to improve Web search, ACM Transactions on Information Systems 23 (2005) 147–168.

[7] Y. Xu, D. Mease, Evaluating web search using task completion time, in: the 32nd international ACM SIGIR conference, ACM Press, New York, New York, USA, 2009, p. 676.

[8] S. Lawrence, C. L. Giles, Accessibility of information on the Web, intelligence 11 (2000) 32–39.

[9] D. E. Rose, D. Levinson, Understanding user goals in web search, in: Proceedings of the 13th International Conference on World Wide Web, WWW '04, ACM, New York, NY, USA, 2004, pp. 13–19.

[10] T. Joachims, Optimizing search engines using clickthrough data, in: the eighth ACM SIGKDD international conference, ACM Press, New York, New York, USA, 2002, p. 133.

[11] A. M. Zareh Bidoki, N. Yazdani, DistanceRank: An intelligent ranking algorithm for web pages, Information Processing & Management 44 (2008) 877–892.

[12] C. Olston, E. H. Chi, ScentTrails: Integrating browsing and searching on the Web, ACM Transactions on Computer-Human Interaction 10 (2003) 177–197.

[13] J. Wu, K. Aberer, Swarm intelligent surfing in the web, Web Engineering (2003).

[14] Y. Freund, R. Iyer, R. E. Schapire, Y. Singer, An efficient boosting algorithm for combining preferences, Journal of Machine Learning Research 4 (2004) 933–969.

[15] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender, Learning to rank using gradient descent, in: ICML '05: Proceedings of the 22nd international conference on Machine learning, Microsoft, ACM, New York, New York, USA, 2005, pp. 89–96.

[16] Z. Zheng, K. Chen, G. Sun, H. Zha, A regression framework for learning ranking functions using relative relevance judgments, in: the 30th annual international ACM SIGIR conference, ACM Press, New York, New York, USA, 2007, p. 287.

[17] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, G. Sun, A General Boosting Method and its Application to Learning Ranking Functions for Web Search., NIPS (2007) 1697–1704.

[18] J. Xu, H. Li, AdaRank: A boosting algorithm for information retrieval, in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'07, Microsoft Research Asia, Beijing, China, pp. 391–398.

[19] P. Li, C. J. C. Burges, Q. Wu, McRank: Learning to rank using multiple classification and gradient boosting, in: Advances in Neural Information Processing Systems 20 - Proceedings of the 2007 Conference, Cornell University, Ithaca, United States.

[20] J. L. Elsas, V. R. Carvalho, J. G. Carbonell, Fast learning of document ranking functions with the committee perceptron, in: WSDM'08 - Proceedings of the 2008 International Conference on Web Search and Data Mining, Carnegie-Mellon University, Pittsburgh, United States, pp. 55–63.

[21] Z. Chen, X. Meng, B. Zhu, R. H. Fowler, WebSail: from on-line learning to Web search, in: WISE 2000: 1st International Conference on Web Information Systems Engineering, IEEE Comput. Soc, 2000, pp. 206–213.

[22] F. Radlinski, T. Joachims, Active exploration for learning rankings from clickthrough data, in: the 13th ACM SIGKDD international conference, ACM Press, New York, New York, USA, 2007, p. 570.

[23] F. Radlinski, R. Kleinberg, T. Joachims, Learning diverse rankings with multi-armed bandits, in: the 25th international conference, ACM Press, New York, New York, USA, 2008, pp. 784–791.

[24] J. Y. Yeh, J. Y. Lin, H. R. Ke, W. P. Yang, Learning to rank for information retrieval using genetic programming, in: Proceedings of SIGIR 2007: Learning to Rank for Information Retrieval.

[25] E. Diaz-Aviles, W. Nejdl, L. Schmidt-Thieme, Swarming to rank for information retrieval, in: Proceedings of the 11th Annual conference on Genetic and evolutionary computation - GECCO '09.

[26] I. Kushchu, Web-Based Evolutionary and Adaptive Information Retrieval, IEEE Transactions on Evolutionary Computation 9 (2005) 117–125.

[27] M. Dorigo, V. Maniezzo, A. Colorni, The Ant System: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics 26 (1996) 29–41.

[28] A. Engelbrecht, X. Li, M. Middendorf, L. M. Gambardella, Editorial Special Issue: Swarm Intelligence, IEEE Transactions on Evolutionary Computation 13 (2009) 677–680.

[29] T. Stützle, H. H. Hoos, MAX–MIN ant system, Future generation computer systems (2000).

[30] X.-M. Hu, J. Zhang, Y. Li, Orthogonal Methods Based Ant Colony Search for Solving Continuous Optimization Problems, Journal of Computer Science and Technology 23 (2008) 2–18.

[31] B. Bullnheimer, R. F. Hartl, C. Strauß, A New Rank Based Version of the Ant System - A Computational Study, Central European Journal for Operations Research and Economics 7 (1997) 25–38.

[32] A. Malizia, K. Olsen, Toward a New Search Paradigm-Can We Learn from Ants?, Computer 45 (2012) 89–91.

[33] T. Turchi, A. Malizia, P. Castellucci, K. Olsen, Collaborative information seeking with Ant Colony Ranking in real-time, Proceedings of the th Italian Research Conference on Digital Libraries (forthcoming).

[34] Y. Xie, D. O'Hallaron, Locality in search engine queries and its implications for caching, in: Proceedings - IEEE INFOCOM, Carnegie-Mellon University, Pittsburgh, United States, pp. 1238–1247.

[35] J.-R. Wen, H.-J. Zhang, Query Clustering in the Web Context, in: Clustering and Information Retrieval, Springer US, Boston, MA, 2004, pp. 195–225.

[36] D. A. Grossman, O. Frieder, Information Retrieval, Springer Netherlands, Dordrecht, 2004.

[37] I. Koychev, I. Schwab, Adaptation to drifting user's interests, in: ECML Workshop: Machine Learning in New Information Age, pp. 39–46.

[38] A. Malizia, K. Olsen, Emerging social search paradigms: can we learn from ants?, in: Proc. of the Italian Workshop on Artificial Life and Evolutionary Computation, pp. 1–4.

[39] D. Hawking, N. Craswell, P. Bailey, K. Griffiths, Measuring Search Engine Quality, Information Retrieval 4 (2001) 33–59.

[40] D. Gayo-Avello, D. Brenes, Making the road by searching-A search engine based on Swarm Information Foraging, arXiv.org (2009).

[41] B. Smyth, E. Balfe, J. Freyne, P. Briggs, M. Coyle, O. Boydell, Exploiting query repetition and regularity in an adaptive community-based Web search engine, User Modelling and User-Adapted Interaction 14 (2004) 383–423.

[42] S. Chakrabarti, A. Frieze, J. Vera, The influence of search engines on preferential attachment, in: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, pp. 293–300.

[43] P. S. Efraimidis, P. G. Spirakis, Weighted random sampling with a reservoir, Information Processing Letters 97 (2006).

[44] K. Church, M. T. Keane, B. Smyth, The first click is the deepest: assessing information scent predictions for a personalized search engine, Proc AH2004 Workshop (2004).

[45] T. Joachims, Evaluating Retrieval Performance Using Clickthrough Data., Text Mining (2003) 79–96.

[46] Y. Liu, Y. Fu, M. Zhang, S. Ma, L. Ru, Automatic search engine performance evaluation with click-through data analysis, in: 16th International World Wide Web Conference, WWW2007, Tsinghua University, Beijing, China, pp. 1133–1134.

[47] S. Jung, J. L. Herlocker, J. Webster, Click data as implicit relevance feedback in web search, Information Processing & Management 43 (2007) 791–807.

[48] G. Pass, A. Chowdhury, C. Torgeson, A picture of search, in: ACM International Conference Proceeding Series, America Online, United States.

[49] J. Gao, W. Yuan, X. Li, K. Deng, J.-Y. Nie, Smoothing clickthrough data for web search ranking, in: Proceedings - 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Microsoft Research, Redmond, United States, ACM Press, New York, New York, USA, 2009, pp. 355–362.

[50] T. Sakai, Alternatives to Bpref, in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'07, NewsWatch, Inc, Tokyo, Japan, pp. 71–78.

[51] K. Jarvelin, J. Kekalainen, Cumulated gain-based evaluation of IR techniques, ACM Transactions on Information Systems 20 (2002) 422–446.

[52] N. Martínez-Bazan, V. Muntés-Mulero, S. Gómez-Villamor, J. Nin, M.-A. Sánchez-Martínez, J.-L. Larriba-Pey, Dex: high-performance exploration on large graphs for information retrieval, in: CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, CSIC - Instituto de Investigacion en Inteligencia Artificial, ACM, New York, New York, USA, 2007, pp. 573–582.

[53] B. Liu, R. Jones, K. L. Klinkner, Measuring the meaning in time series clustering of text search queries, in: International Conference on Information and Knowledge Management, Proceedings, Virginia Polytechnic Institute and State University, Blacksburg, United States, pp. 836–837.

[54] C. Silverstein, H. Marais, M. Henzinger, M. Moricz, Analysis of a very large web search engine query log, ACM SIGIR Forum 33 (1999) 6–12.

[55] C. Eickhoff, K. Collins-Thompson, P. N. Bennett, S. Dumais, Personalizing atypical web search sessions, in: Proceedings of the sixth ACM international conference on Web search and data mining - WSDM '13.

[56] A. H. Awadallah, R. W. White, S. T. Dumais, Y.-M. Wang, Struggling or exploring?: disambiguating long search sessions., WSDM (2014) 53–62.