# DEWS: A Decentralized Engine for Web Search

Reaz Ahmed*, Md. Faizul Bari*, Rakibul Haque*, Raouf Boutaba*, and Bertrand Mathieu†
*David R. Cheriton School of Computer Science, University of Waterloo
{r5ahmed | mfbari | m9haque | rboutaba}@uwaterloo.ca
†Orange Labs, Lannion, France
bertrand2.mathieu@orange-ftgroup.com

***Abstract*—Contemporary Web search is governed by centrally controlled search engines, which is not healthy for our online freedom and privacy. A better solution is to enable the Web to index itself in a decentralized manner. In this work we propose a decentralized Web search mechanism, named DEWS, which enables existing webservers to collaborate with each other to build a distributed index of the Web. DEWS can rank search results based on query keyword relevance and relative importance of webpages. DEWS also supports approximate matching of query keywords in web documents. Simulation results show that the ranking accuracy of DEWS is very close to the centralized case, while network overhead for collaborative search and indexing is logarithmic on network size.**

## I. Introduction

Internet is the largest knowledge base that mankind has ever created. Autonomous hosting infrastructure and voluntary contributions from millions of Internet users have given the Internet its edge. However, contemporary Web search services are governed by centrally controlled search engines, which is not healthy for our online freedom due to the following reasons. A Web search service provider can be compromised to evict certain websites from the search results, which can reduce the websites' visibility. Relative ranking of websites in search results can be biased according to the service providers' preference. Moreover, a service provider can record its users' search history for targeted advertisements or spying. For example, the recent PRISM scandal surfaced the secret role of the major service providers in continuously tracking our web search and browsing history.

A decentralized Web search service can subside these problems by distributing the control over a large number of network nodes. No single authority will control the search result. It will be computed by combining partial results from multiple nodes. Thus a large number of nodes have to be compromised to bias a search result. Moreover, a user's queries will be resolved by different nodes. All of these nodes have to be compromised to accumulate the user's search history.

A number of research works ([1], [2], [3]) and implementations (YacY$^{www.yacy.net}$, Faroo$^{www.faroo.com}$) have focused on distributed Web search and ranking in peer-to-peer (P2P) networks. These approaches have two potential problems in common: (a) *lookup overhead*: number of network messages required for index/peer lookup is much higher in P2P networks compared to a centralized alternative, (b) *churn*: maintaining a consistent index in presence of high peer churn is not feasible. Thus, those solutions have issues with performance and accuracy requirements.

In this paper we take a very different approach to decentralized web indexing and ranking. Instead of relying on an overlay of regular Internet users, we build an overlay between webservers. We exploit the stability in webserver overlay to heavily cache links (network addresses) that we use as routing shortcuts. Thus we achieve faster lookup, lower messaging overhead, and higher ranking accuracy in search results.

The rest of this paper is organized as follows. First we present the DEWS architecture in §II. Then we validate the concepts presented in this work through extensive simulations and present the results in §III. We present and compare with the related works in §IV. Finally, we conclude with future research directions in §V.

## II. System Architecture

We have used Plexus protocol [4] to build an overlay network between the webservers participating in DEWS. Since the webserver overlay is fairly stable, each webserver caches links (network addresses) to other servers. This link caching reduces network overhead during the indexing and routing processes. On top of the overlay topology we maintain two indexes for distributed ranking (§II-A1) and keyword search (§II-A2), respectively. Functionally DEWS is similar to a centralized search engine. It generates ranked results for keyword search (§II-B1). From now on we use the terms webserver and node interchangeably.

Plexus is a unique Distributed Hash Table (DHT) technique with built-in support for approximate matching, which is not easily achievable by other DHT techniques. Plexus routing scales logarithmically with network size. Plexus delivers a high level of fault-resilience by using systematic replication and redundant routing paths. Because of these advantages we have used Plexus protocol to build the webserver overlay. Here, we summarize the basic concepts in Plexus followed by the proposed extensions.

### A. Indexing Architecture

Metrics used for ranking web search results can be broadly classified into two categories: a) hyperlink structure of the webpages, and b) keyword to document relevance. Techniques from Information Retrieval (IR) literature are used for measuring relevance ranks. While link structure analysis algorithms like PageRank [7] is used for computing weights or relative significance of each URL. In the following, we present a distributed mechanism that uses both of these measures for ranking search results. The same ranking mechanism can be used to consider other factors for ranking like user feedback.

In this work we assume that each Webserver will advertise its websites to the DEWS indexing structure. Alternatively, a distributed crawler can be deployed in each webserver that will crawl the webservers not participating in DEWS.

*1) Hyperlink Index:* Algorithms for computing webpage weights based on hyperlink structure are iterative and require many iterations to converge. In each iteration webpage weights are updated and the new weights are propagated to adjacent URLs for computation in the next iteration. To implement such ranking mechanisms on websites distributed across an overlay network, we need to preserve the adjacency relationships in hyperlink graph while mapping websites to nodes. If hyperlinked websites are mapped to the same node or adjacent nodes then network overhead for computing URL weights will be significantly reduced. Unfortunately, there exists no straight forward hyperlink structure preserving mapping of the Web to an overlay network.

In DEWS, we retain the hyperlink structure as a virtual overlay on top of Plexus overlay. We use a standard shift-add hash function ($\hbar(\cdot)$) to map a website's base URL, say $u_i$, to a codeword, say $c_k = \hbar(u_i)$. Then we use Plexus routing to lookup $\beta(u_i)$, which is the node responsible for indexing codeword $c_k$ (Fig. 1(a)). For each outgoing hyperlink, say $u_{it}$, of $u_i$ we find the responsible node $\beta(u_{it})$ in a similar manner. During distributed link-structure analysis $\beta(u_i)$ has to frequently send weight update messages to $\beta(u_{it})$. Hence we cache the network address of node $\beta(u_{it})$ at node $\beta(u_i)$, which we call a soft-link. Soft-links mitigate the network overhead generated from repeated lookups during PageRank computation.

The index stored in $\beta(u_i)$ has the form $< u_i, w_i, \{< u_{it}, \beta(u_{it}) >\} >$, where $w_i$ is the PageRank weight of $u_i$. $w_i$ is computed as $w_i = (1-\eta) + \eta \sum_{t=1}^{g} \frac{w_{it}}{L(u_{it})}$. Here, $\eta$ (usually 0.85) is the damping factor for PageRank algorithm. $\{u_{it}\}$ is the set of webpages linked by $u_i$ and $L(u_{it})$ is the number of outgoing links from webpage $u_{it}$.

Each node periodically executes Algorithm 1 to maintain the PageRank weights updated in a distributed manner. To communicate PageRank information between the nodes, we use a PageRank update message containing the triplet $< u_s, u_i, \frac{w_s}{L(u_s)} >$, where node $\beta(u_s)$ sends the message to node $\beta(u_i)$, and $\frac{w_s}{L(u_s)}$ is the contribution of $u_s$ towards PageRank weight of $u_i$. Each node maintains a separate message queue ($\mathcal{Q}_{u_i}$) for each website ($u_i$) it has indexed. Incoming PageRank messages are queued for a pre-specified period of time and is used to compute the PageRank for each webpage. If the change in newly computed PageRank value is greater than a pre-defined threshold $\theta$, PageRank update messages are sent to $\beta(u_{it})$ for each hyperlinked website $u_{it}$.

*2) Keyword Index:* We use Plexus to build an inverted index on the important keywords for each webpage. This index allows us to lookup a query keyword and find all the webpages containing that keyword by forwarding the query message to a small number of nodes. Suppose, $\mathcal{K}_i^{rep} = \{k_{ij}^{rep}\}$ is the set of representative keywords for $u_i$. For each keyword $k_{ij}^{rep}$ in $\mathcal{K}_i^{rep}$, we generate $k_{ij}^{dmp}$ by applying Double Metaphone encoding [8] on $k_{ij}^{rep}$. Double Metaphone encoding attempts to detect phonetic ('sounds-alike') relationship between words. Motivation behind adapting phonetic encoding is twofold: i) any two phonetically equal keywords have no edit distance between them, ii) phonetically inequivalent keywords have less edit distance than the edit distance between the original keywords. In both cases, Hamming distance between encoded advertisement and search patterns is lesser than that of the patterns generated from original keywords. This low Hamming distance increases the percentage of common codewords computed during advertisement and search, which eventually increases the possibility of finding relevant webpages.

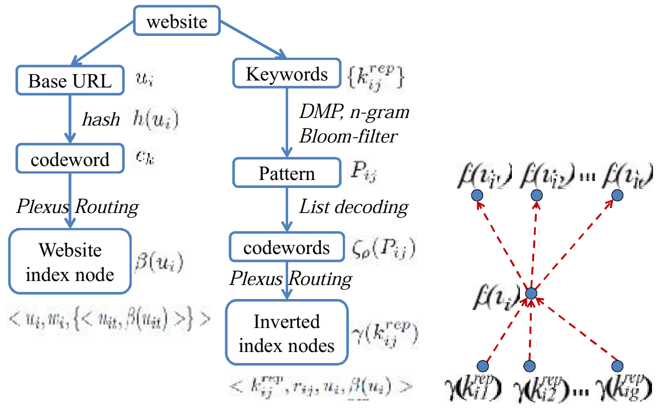
**Algorithm 1** *Update PageRank*

```
1:  Internals:
        $\mathcal{Q}_{u_i}$: PageRank message queue for $u_i$
        $L(u_i)$: Number of outlinks for $u_i$
        $w_i$: PageRank weight of $u_i$
        $\eta$: Damping factor for PageRank algorithm
        $\theta$: Update propagation threshold
2:  for all URL $u_i$ indexed in this node $\beta(u_i)$ do
3:      $temp \leftarrow 0$
4:      for all $< u_{si}, u_i, \frac{w_{si}}{L(u_{si})} > \in \mathcal{Q}_{u_i}$ do
5:          $temp \leftarrow temp + \frac{w_{si}}{L(u_{si})}$
6:      end for
7:      $w_i^{new} \leftarrow (1-\eta) + \eta * temp$
8:      if $|w_i^{new} - w_i| > \theta$ then
9:          $w_i \leftarrow w_i^{new}$
10:         for all out link $u_{it}$ from $u_i$ do
11:             send PageRank message $< u_i, u_{it}, \frac{w_i}{L(u_i)} >$ to $\beta(u_{it})$
12:         end for
13:     end if
14: end for
```

The process of generating keyword index is depicted in Fig. 1(a). To generate advertisement or query pattern $P_{ij}$ from keyword $k_{ij}^{rep}$, we fragment $k_{ij}^{rep}$ into $k$-grams ($\{k_{ij}^{rep}\}$) and encode these $k$-grams along with $k_{ij}^{dmp}$ into an $b$-bit Bloom filter. We use this Bloom filter as a pattern $P_{ij}$ in $\mathbb{F}_2^b$ and *list decode*[1] it to a set of codewords, $\zeta_\rho(P_{ij}) = \{c_k | c_k \in \mathcal{C} \wedge \delta(P_{ij}, c_k) < \rho\}$, where $\zeta_\rho(\cdot)$ is a list decoding function and $\rho$ is list decoding radius. Finally, we use Plexus routing to lookup and store the index on $k_{ij}^{rep}$ at the nodes responsible for codewords in $\zeta_\rho(P_{ij})$. The index for $k_{ij}^{rep}$ is a quadruple $< k_{ij}^{rep}, r_{ij}, u_i, \beta(u_i) >$, where $r_{ij}$ is a measure of semantic relevance of $k_{ij}^{rep}$ to $u_i$. We use $\gamma(k_{ij}^{rep})$ to represent the set of nodes responsible for $k_{ij}^{rep}$. Evidently, $\gamma(k_{ij}^{rep}) \equiv lookup(\zeta_\rho(BF(\{k_{ij}^{rep}\} \cup \{k_{ij}^{dmp}\})))$, $BF(\cdot)$ represents Bloom filter encoding function.

We use Vector Space Model (VSM) for computing relevance of keyword $k_{ij}^{rep}$ to URL $u_i$. In VSM, each URL $u_i$ is represented as a vector $\vec{\nu_i} = (r_{i1}, \ldots, r_{ig})$, where $r_{ij}$ represents the relevance of the term or keyword $k_{ij}^{rep}$ in $u_i$, and $g$ is the

[1]List decoding is the process of finding all the codewords within a given Hamming distance from a (advertisement or query) pattern



(a) Indexing process in DEWS (b) Softlink structure

Fig. 1. Softlink computation in DEWS

number of representative keywords in $u_i$. The relevance weight $r_{ij}$, of the $j^{th}$ keyword is computed as $tf(k_{ij}^{rep}) * idf(k_{ij}^{rep})$. Here, term frequency $tf(k_{ij}^{rep})$ is the number of occurrences of $k_{ij}^{rep}$ in webpage $u_i$, while inverse document frequency $idf(k_{ij}^{rep})$ is computed as $idf(k_{ij}^{rep}) = \log \frac{U}{\psi(k_{ij}^{rep})}$. Here, $U$ is the total number of webpages and $\psi(k_{ij}^{rep})$ is the number of webpages containing keyword $k_{ij}^{rep}$. $tf(k_{ij}^{rep})$ is a measure of the relevance of $k_{ij}^{rep}$ to $u_i$, while $idf(k_{ij}^{rep})$ is a measure of relative importance of $k_{ij}^{rep}$ *w.r.t.* other keywords. *idf* is used to prevent a common term from gaining higher weight and a rare term from having lower weight in a collection.

Computing $tf(k_{ij}^{rep})$ for each keyword $k_{ij}^{rep} \in \mathcal{K}_i^{rep}$ from $u_i$ is straight forward and can be done by analyzing the webpages in $u_i$. For computing $idf(k_{ij}^{rep})$ we need to know $U$ and $\psi(k_{ij}^{rep})$. Now, all webpages containing keyword $k_{ij}^{rep}$ are indexed at the same node. Hence, $\psi(k_{ij}^{rep})$ can be computed by searching the local repository of that node. However, it is not trivial to compute $U$ in a purely decentralized way. Instead we use the total number of indexed URLs in a node in place of $U$ as advocated in [9].

PageRank for URL $u_i$ is computed and maintained in node $\beta(u_i)$, while the computed PageRank value $w_i$ is used in nodes $\gamma(k_{ij}^{rep})$, where a representative keyword $k_{ij}^{rep}$ for webpage $u_i$ is indexed. The Web is continuously evolving and PageRank for the webpages are likely to change over time. As a result, storing PageRank weight, $w_i$ to the nodes in $\gamma(k_{ij}^{rep})$ will not be sufficient; we have to refresh it periodically. To reduce network overhead, softlink to $\beta(u_i)$ are stored in nodes $\gamma(k_{ij}^{rep})$. This softlink structure between nodes $\beta(u_i)$, $\beta(u_{it})$ and $\gamma(k_{ij}^{rep})$ is presented in Fig. 1(b).

*3) Advertising Websites:* The pseudocode for advertising a webpage is presented in Algorithm 2. As discussed in the previous two sections, we maintain two sets of indexes for a webpage: a) using site URL $u_i$ and b) using representative keywords $\mathcal{K}_i^{rep}$. In lines 3 to 8 of Algorithm 2, we compute the index on $u_i$, which involves computing the softlinks $(\beta(u_{it}))$ for each outgoing hyperlinks from $u_i$ and storing in node $\beta(u_i)$. In lines 9 to 18, we compute the indexes on $\mathcal{K}_i^{rep}$ and advertise the indexes to the responsible nodes.

**Algorithm 2** *Publish webpage*

```
 1: Inputs:
        $u_i$: URL of the webpage to be advertised
 2: Functions:
        $\hbar(u_i)$: hash map $u_i$ to a codeword
        $\gamma_r(P)$: $\{c_k | c_k \in \mathcal{C} \wedge \delta(P, c_k) \leq r\}$
        $lookup(c_k)$: Finds the node that stores $c_k$
 3: $\beta(u_i) \leftarrow lookup(\hbar(u_i))$
 4: for all out-link $u_{it}$ of $\{u_i\}$ do
 5:     $\beta(u_{it}) \leftarrow lookup(\hbar(u_{it}))$
 6: end for
 7: $w_i \leftarrow$ initial PageRank of $u_i$
 8: store $< u_i, w_i, \{u_{it}, \beta(u_{it})\} >$ to node $\beta(u_i)$
 9: $\mathcal{K}_i^{rep} \leftarrow$ set of representative keywords of $u_i$
10: for all $k_{ij}^{rep}$ in $\mathcal{K}_i^{rep}$ do
11:     $k_{ij}^{dmp} \leftarrow DoubleMetaphoneEncode(k_{ij}^{rep})$
12:     $P_{ij} \leftarrow BloomFilterEncode(\{k_{ij}^{rep}\} \cup \{k_{ij}^{dmp}\})$
13:     $r_{ij} \leftarrow$ relevance of $k_{ij}^{rep}$ to $u_i$
14:     for all $c_k$ in $\zeta_\rho(P_{ij})$ do
15:         $v \leftarrow lookup(c_k)$
16:         store $< k_{ij}^{rep}, r_{ij}, u_i, \beta(u_i) >$ to node $v$
17:     end for
18: end for
```

## B. Resolving Web Query

*1) Search and Ranking:* To resolve Web queries in DEWS, we breakdown the query into subqueries – each consisting of a single query keyword, say $q_l$. Similar to the keyword advertisement process explained in §II-A2, we compute the Double Metaphone (*i.e.*, $q_l^{dmp}$), $k$-gram ($\{q_l\}$), and encode them in a Bloom filter $P_l$. Then we use the Plexus protocol to find the nodes responsible for storing the keywords similar to $q_l$ and retrieve a list of triplets like $\{< u_i, w_i, r_{il} >\}$, which gives us the URLs $(u_i)$ containing query keyword $q_l$ along with the link structure weight $(w_i)$ of $u_i$, and semantic relevance of $q_l$ to $u_i$, *i.e.*, $r_{il}$. Now, the querying node computes the ranks of the extracted URLs using the following equation:

$$rank(u_i) = \sum_{q_l} \sum_{u_i} \vartheta_{il} (\mu \cdot w_i + (1 - \mu) \cdot r_{il}) \qquad (1)$$

In Equation 1, $\mu$ is a weight adjustment factor governing the relative importance of link structure weight $(w_i)$ and semantic relevance $(r_{il})$ in the rank computation process. While $\vartheta_{il}$ is a binary variable that assumes a value of one when webpage $u_i$ contains keyword $q_l$ and zero otherwise. While the implication of simply summing $w_i$ and $r_{il}$ together is not obvious, similar approaches were proposed in [10]. Although we can devise complicated ways to combine these two measures together, a simple summation suffices to achieve the desired effect. The query process in DEWS is explained in Algorithm 3.

## III. Evaluation

In this section, we present the simulation results to validate the proposed concepts. We use the following metrics for this evaluation: routing hops, indexing overhead, convergence time, network message overhead, and ranking accuracy. For our simulations we have varied the number of URLs, number of queries, number of nodes, number of keywords, edit distance between advertised and query keywords *etc.*

**Algorithm 3** *Query*

```
Input:
    $Q$: set of query keywords $\{q_l\}$
    $T$: Most relevant $T$ webpages requested
Internals:
    $\mu$: Weight adjustment on link-structure vs relevance
    $\rho$: list decoding radius
$\xi \leftarrow$ empty associative array
for all $q_l \in Q$ do
    $q_l^{dmp} \leftarrow DoubleMetaphoneEncode(q_l)$
    $P_l \leftarrow BloomFilterEncode(\{q_l\} \cup \{q_l^{dmp}\})$
    for all $c_k \in listDecode_\rho(P_l)$ do
        $n \leftarrow lookup(c_k)$
        for all $\{<u_i, w_i, r_{il}>\} \in n.retrive(q_l)$ do
            $\xi[u_i].value \leftarrow \xi[u_i].value + \mu \cdot w_i + (1-\mu) \cdot r_{il}$
        end for
    end for
end for
sort $\xi$ based on $value$
return top $T$ $u_i$ from $\xi$
```

## A. Simulation Setup

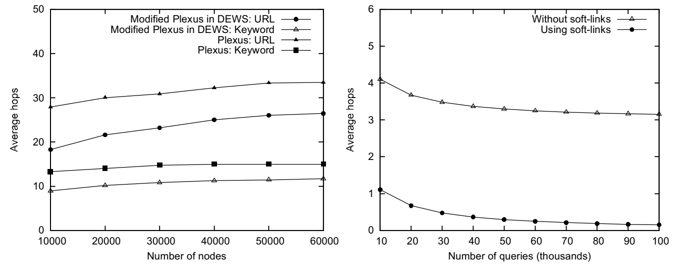
We have used the "Web Track" dataset from LETOR 3.0 [11] for our experiments. The dataset is composed of the TREC 2003 and 2004 datasets, which contain a crawl of the .gov domain done on January, 2002. There are a total of 1,053,110 webpages and 11,164,829 hyperlinks in the collection. The collection contains three search tasks: topic distillation, homepage finding, and named page finding. For computing the PageRank of the webpages in the dataset we are using the "Sitemap" of the .gov collection. Representative keywords are extracted by parsing the meta-data files associated with each query file. We have developed a cycle-driven simulator. In each cycle, each node in the simulated network processes its incoming messages and routes to the destinations specified in the messages using the Plexus routing algorithm presented. We used the second-order Reed Muller code, $RM(2,6)$ for implementing Plexus.

## B. Routing Performance

In this section we evaluate advertisement and indexing behavior of DEWS.

*1) Scalability:* Fig. 2(a) presents the average routing hops per advertisement for different network sizes. We can infer the following from this curve: firstly, average hops for advertisement do not increase significantly with increased network size. And second, URL advertisement requires more hops than keyword advertisement. The reason behind this behavior can be well-explained from Fig. 1(b). For advertising a URL, say $u_i$, we have to lookup $\beta(u_{it})$ for each out link of $u_i$. On the other hand, while advertising the keywords in $\mathcal{K}_i^{rep}$ we lookup $\beta(u_i)$ once and use it for every keyword $k_{ij}^{rep} \in \mathcal{K}_i^{rep}$. Hence the higher cost for URL advertisement.

*2) Effectiveness of Softlink:* In this experiment (Fig. 2(b)), we measure the impact of softlinks on query resolution. We conducted this experiment in a network of 50K nodes and measured average routing hops with different numbers of simultaneous queries. Average hops for resolving a query decreases whith increase query rate due to an increased level

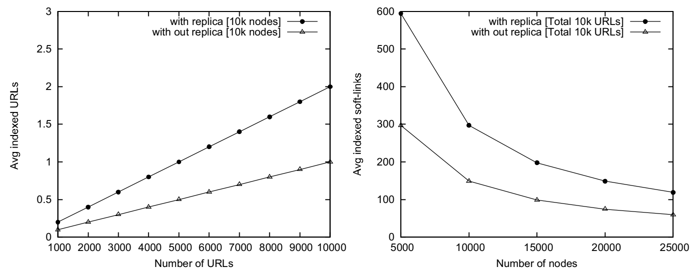


(a) Advertisement scalability (b) Query routing efficiency

Fig. 2. Routing efficiency

of message aggregation at each hop. It can also be noticed that softlinks reduces the average number of hops significantly.

*3) Index Overhead:* We present the average number of indexed URLs and softlinks per node in Figures 3(a) and 3(b), respectively. It is evident from Fig. 3(a) that the average number of indexed URLs varies linearly with the number of URL advertisements for a fixed network size. In turn, Fig. 3(b) presents that the average number of softlinks per node decreases with increased network size when the number of advertised URLs is fixed. These curves demonstrate the uniform distribution of URL indexes and softlinks over the nodes. Moreover, the number of indexed URLs and softlinks become almost double in presence of replication. The reason behind this behavior can be explained from the replication policy in Plexus, where the node responsible for codeword $c_k$ maintains a replica of its indexes to the node responsible for codeword $\overline{c_k}$. Here, $\overline{c_k}$ is the bit-wise complement of $c_k$.



(a) URLs per node (b) Softlinks per node

Fig. 3. Index overhead

## C. Ranking Performance

Here, we measure the accuracy (§III-C1) and convergence time (§III-C2) of distributed ranking as presented in §II-A.

*1) Accuracy:* We use Spearman's footrule distance[2] to measure the accuracy of distributed PageRank algorithm in DEWS. Fig. 4(a) presents Spearman's footrule distances between distributed PageRank in DEWS and centrally computed PageRank for top-20, top-100, and top-1000 results. We advertised 10K URLs on a network of 50K nodes and PageRank update

[2]For two ordered lists $\sigma_1$ and $\sigma_2$ of size $k$ each, Spearman's footrule distance is defined as $F(\sigma_1,\sigma_2) = \frac{\sum_{i=1}^{k}|\sigma_1(i)-\sigma_2(i)|}{k*k}$, where $\sigma_1(i)$ and $\sigma_2(i)$ are the positions of URL $i$ in $\sigma_1$ and $\sigma_2$, respectively. If a URL is present in one list and absent in the other, its position in the latter list is considered as $k$+1. Lower footrule distance implies better ranking accuracy.

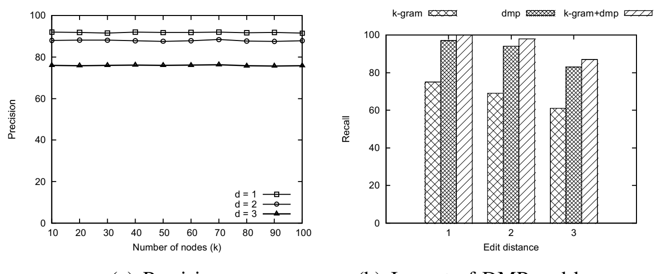
interval was set to 2 cycles. It is evident from Fig. 4(a) that Spearman's footrule distance drops significantly for the first 60 cycles due to rapid convergence in our distributed PageRank algorithm. PageRank values become almost constant after 120 cycles. It should also be noted that the Spearman's footrule distance becomes around 0.1 after 100 cycles. It indicates that the distributed PageRank weights become very close to the centrally computed PageRank weights.

*2) Convergence:* In this experiment we measure convergence time (in number of cycles) and network overhead (in number of messages) for the distributed ranking algorithm in DEWS. We performed this experiment on a network of 50K nodes and observed convergence behavior *w.r.t.* the number of advertised URLs. We used a value of 0.0001 for $\theta$ – the update propagation threshold (see Algorithm 1) and varied the update propagation interval from 1 to 3 cycles. We assume that the distributed ranking algorithm has converged when the PageRank weights converge for every URL.

It can be seen from Fig. 4(b) that the number of cycles to converge does not increase significantly as the number of advertised URL increases. On the other hand, convergence time reduces as update propagation interval is reduced from 3 cycles to 1 cycle. Obviously this reduction in convergence time is achieved at the expense of increased network overhead as can be seen in Fig. 4(c). As the update propagation interval increases, each node gets enough time to accumulate all incoming PageRank weights and the computed PageRank weight converges faster (see §II-A1).

### *D. Search Performance*

*1) Flexibility and accuracy:* DEWS provides flexible search with partially specified or misspelled keywords. We indexed 10K URLs and the associated keywords in networks of different sizes. We generated 10K queries from randomly selected indexed keywords by varying edit distances from 1 to 3. Average recall rate remains constant at 100%, 98%, and 87% for edit distances 1, 2 and 3, respectively. Recall rate is lower for higher edit distances because the Hamming distance between advertisement and query patterns is proportional to the edit distance between advertisement and query keywords.



(a) Precision (b) Impact of DMP and k-grams
Fig. 5. Search performance in DEWS

Fig. 5(a) presents the average precision of search results for 10K queries in networks of different sizes. From this graph, it is observed that precisions remain constant at 92%, 88%, and 76% for edit distances 1, 2, and 3, respectively. Precision becomes lower when edit distance increases because many irrelevant webpages are included in the search results.

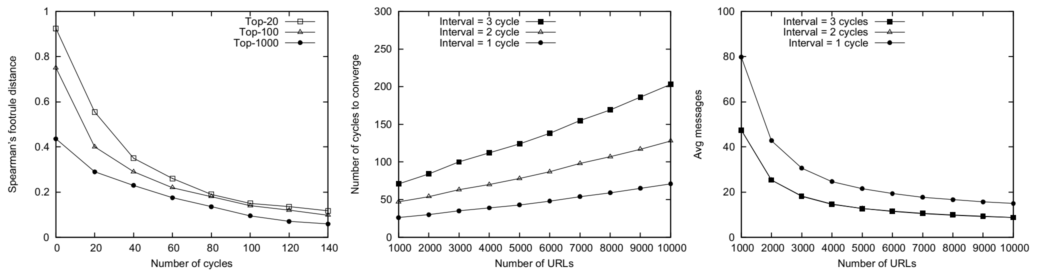
Notably, the results in Fig. 5 affirm that precision and recall are independent of network size and only depend on edit distance between advertisement and query keywords. DEWS achieves a recall rate of 98% and a precision of 88% for queries with edit distance two from the original keywords. However, recall and precision of search results drop to 87% and 76%, respectively for edit distance three. Hence, we can expect to achieve very good precision and recall for partially specified or misspelled keywords within edit distance 2.

Fig. 5(b) shows the average recall rate of the search results for 10K queries in a network of 100K nodes where the query keywords have varying edit distances from the advertised keywords. In this experiment we evaluate the impact of Double Metaphone encoding (DMP) and $k$-grams decomposition (with $k$=1) of query or advertisement keywords on recall. From this figure, it can be observed that the best recall is achieved when DMP encoding and $k$-grams decomposition are combined for generating the advertisement and query patterns.

## IV. Related Works

Integrated solution for distributed web search is not well investigated in literature, although there exists implementations like YacY and Faroo. Both of these implementations use gossip based index propagation and distributed crawlers. YacY uses distributed PageRank, while Faroo relies on user feedback to ranking search results. On the other hand, almost all research works in this domain focus on distributed ranking. These works can be classified in two broad categories: link structured based ranking and semantic relevance based ranking.

Link structure analysis is a popular technique for ranking. Google uses the PageRank [7] algorithm to compute page weights that measure its authority-ship. Bender *et al.* [1] proposed a distributed document scoring and ranking system that focuses on correlation between query keywords that appear in query logs. Sankaralingam *et al.* proposed a P2P PageRank algorithm in [2], where every peer initializes a PageRank score to its local documents and propagates update messages to adjacent peers. DynaRank [3] works in a similar manner, but only propagates update messages when the magnitude of weight change is greater than a threshold value. In JXP [12], each peer computes initial weights for their local pages using standard PageRank and introduces the notion of "external world", which is a logical node representing the outgoing and incoming hyperlinks from the webpages stored in a peer. Each time a peer meets with another peer, it updates knowledge about its external world. Wang *et al.* used two types of ranks for overall ranking: *local PageRank* computed in each peer based on the standard Pagerank and *ServerRank* computed as the highest local PageRank or sum of all the PageRanks of a web server [13]. SiteRank [14] computes the rank at the granulaity level of websites instead of web page level using PageRank [7]. Fu proposed a mechanism to retrieve top-$k$ results from an unstructured peer-to-peer network by utilizing query caching at each node in [15]. Richardson *et al.* showed



(a) Spearman's footrule distance (b) Convergence time (c) Network overhead

Fig. 4. Ranking performance

that rank accuracy can be significantly improved by replicating a document at multiple peers based on the number of times a document appears in the top-$k$ results and the frequency of queries for that document in [16], [17].

Another research trend is to use Information Retrieval techniques such as VSM (Vector Space Model), which is widely used in centralized ranking systems. However, computing global weight (inverse document frequency or $idf$) in a distributed systems is challenging. A random sampling technique is used in [18] to compute approximate value of $idf$. In a DHT-based structured network, each keyword is mapped to a particular peer and that peer can compute the approximate value of $idf$ [9]. A Gossip-based algorithm is proposed in [19] to approximate both term frequency ($tf$) and $idf$ for unstructured P2P networks.

Existing distributed web searching and ranking techniques do not use both structural and relevance ranking at the same time. Ranking in these systems is based on incomplete information. On the contrary, we use both link structure weights and keyword relevance, and compute ranks utilizing complete information. Thus, our ranking results closely match the centrally computed ones.

## V. Conclusion and future work

In this work, we have presented DEWS - a self-indexing architecture for the Web. DEWS enables webservers to collaboratively index the Web and respond to Web queries in a completely decentralized manner. In DEWS we have the provision for approximate matching on query keywords, and distributed ranking on semantic relevance and link-structure characteristics. As demonstrated by the experimental results, network and storage overheads for achieving this decentralization is not significant and the proposed framework scales well with network size and the number of indexed webpages. In addition, the ranking accuracy of DEWS is comparable to the ranking accuracy of a centralized ranking solution. Compared to a centralized solution, DEWS will incur some network overhead. In exchange DEWS will give us the freedom of searching and exploring the Web without any control or restrictions, as can be imposed by the contemporary search engines.

The concepts presented in this work have been validated with rigorous simulations and critical analysis of the simulation results. As a next step, we intend to develop a DEWS prototype that can be deployed on a real network. We also intend to develop an improved ranking system where PageRank weights will be influenced by the number of common keywords between two pages. We expect this scheme to yield semantically more accurate results.

## References

[1] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer, "P2P content search: Give the web back to the people," in *IPTPS*, 2006.

[2] K. Sankaralingam, M. Yalamanchi, S. Sethumadhavan, and J. Browne, "Pagerank computation and keyword search on distributed systems and p2p networks," *Journal of Grid Comp.*, vol. 1, no. 3, pp. 291–307, 2003.

[3] M. Kale and P. S. Thilagam, "DYNA-RANK: Efficient Calculation and Updation of PageRank," in *ICCSIT*, 2008, pp. 808–812.

[4] R. Ahmed and R. Boutaba, "Plexus: a scalable peer-to-peer protocol enabling efficient subset search," *IEEE/ACM Trans. Netw.*, vol. 17, pp. 130–143, February 2009.

[5] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, pp. 422–426, July 1970.

[6] G. Cohen, *Covering codes*. North Holland, 1997, vol. 54.

[7] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," Stanford Digital Library Technologies Project, Tech. Rep., 1999.

[8] M. R. Haque, R. Ahmed, and R. Boutaba, "Qpm: Phonetic aware p2p searching," in *IEEE Peer-to-Peer Computing*, 2009, pp. 131–134.

[9] Z. Lu, B. Ling, W. Qian, W. S. Ng, and A. Zhou, "A distributed ranking strategy in P2P based IR systems." in *APWeb*, 2004, pp. 279–284.

[10] M. Richardson and P. Domingos, "The Intelligent surfer: Probabilistic Combination of Link and Content Information in PageRank," in *NIPS*, 2001, pp. 1441–1448.

[11] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li, "LETOR: Benchmark Dataset for Research on Learning to Rank for IR," in *LR4IR*, 2007.

[12] J. X. Parreira and G. Weikum, "JXP: Global Authority Scores in a P2P Network," in *8th Int. Workshop on Web and Databases (WebDB)*, 2005.

[13] Y. Wang and D. DeWitt, "Computing pagerank in a distributed internet search system," in *Proc. VLDB*, vol. 30, 2004, pp. 420–431.

[14] J. Wu and K. Aberer, "Using siterank for decentralized computation of web document ranking," in *AH*, 2004, pp. 265–274.

[15] R. Fu, "The quality of probabilistic search in unstructured distributed information retrieval systems," Ph.D. dissertation, UCL, 2012.

[16] S. Richardson and I. Cox, "Ranked accuracy and unstructured distributed search," in *Advances in Information Retrieval*, ser. Lecture Notes in Computer Science. Springer, 2013, vol. 7814, pp. 171–182.

[17] ——, "Increasing ranked accuracy for unstructured distributed search with dynamic replication," in *IEEE P2P*, Sept 2013, pp. 1–5.

[18] V. Gopalakrishnan, R. Morselli, B. Bhattacharjee, P. Keleher, and A. Srinivasan, "Distributed ranked search," in *HiPC*, 2007, pp. 7–20.

[19] R. Neumayer, C. Doulkeridis, and K. Nørvåg, "A hybrid approach for estimating document frequencies in unstructured p2p networks," *Inf. Syst.*, vol. 36, no. 3, pp. 579–595, May 2011.