MAAY: a decentralized personalized search system

Frédéric Dang Ngoc[*◇], Joaquín Keller[◇], Gwendal Simon[◇]
∗ : *LRI - Laboratoire de Recherche en Informatique, Université de Paris Sud*
◇ : *France Telecom - R&D Division*
38 rue du Général Leclerc, 92794 Issy-Moulineaux Cedex, FRANCE
{*frederic.dangngoc, joaquin.keller, gwendal.simon*}*@rd.francetelecom.com*

Abstract

This paper focuses on decentralized personalized search engines. It is composed of three parts. Firstly, we formulate the problem and we propose a graph-based measure of quality of a document given a user and a word. This quality gives a formal measure to test the personalized relevance of answers provided by search engines. Secondly, we present a decentralized system namely MAAY *which aims to provide personalized results to its users. Users can share, search and retrieve documents from others. Nodes locally learn others' profile from previous interactions and use feedbacks to raise words characterizing a document. By querying preferentially nodes with similar profile and by ranking documents according to the requester profile, nodes may find documents which are relevant for them. Finally, we propose a framework for experimental studies of such system. Then, we use it to show that the main principles of* MAAY *could benefits to users.*

1. Introduction

1.1. Motivations

A search engine is a tool which returns a set of documents among an information space to a word-based query initiated by a user. Nowadays, the size of the information space available on the Internet is so large that users pay more attention to the *quality* of responses which highly depends on the *ranking* accuracy of the search engine.

Most popular search engines [2, 14] rely on the hyperlink-based structure of the Web. Not only as an efficient way to crawl it but also to produce a semantically relevant criterion for ranking documents. But, even if search engines satisfy most users, they admit three main shortcomings. Firstly, the web space is dynamic, so documents may change since the last crawl. As a consequence, some files may be missing or wrongly appear in responses. Secondly, the centralization of the system raises some issues, especially on dependability, scalability and privacy. Finally, responses are generic and impersonal, so their relevance is only good in average.

Some distributed search engines have been developed on top of peer-to-peer systems. These systems introduce new information spaces in which the functionalities of the search engine are provided by the peers themselves. In early systems, nodes are connected with a bounded number of randomly chosen peers. It results in a totally unstructured information space. Document searches use flooding-based mechanisms: requests are forwarded until they reach a peer having a file matching the query. As the scalability issue imposes to restrict the number of forwards, the lookup is not exhaustive nor ranked, but it provides up-to-date results in a fully distributed fashion.

The next generation of peer-to-peer systems relies on the Distributed Hash Table abstraction [17, 6, 21, 16]. A well-defined structure is used for routing queries, thus all files related to one word may be easily discovered. However, this implies several drawbacks. First of all, maintaining the overlay may be expensive in a dynamic context. Moreover, the task of publication requires to reach as many peers as the number of words contained in the published document. A request involving several words produces tremendous traffic.

In this paper, we focus on *decentralized personalized* search engines. A *personalized search engine* is a tool which ranks results by taking into consideration the requester profile. Although some studies have already addressed this issue, the solutions rely on centralized systems in which a server knows all users and the whole document space. Rather, we are interested with a search engine that could be used on top of both the whole web and the files owned by users. Decentralized solutions appear as an appealing way to ensure the scalability of the engine we envision.

1.2. Paper Contributions

We first formally define the problem. We propose to use a model based on two bipartite graphs document-user and word-document to measure the quality of a document for a request emitted by a user. This leads to a natural and simple definition of the personalized search engine.

Following, we propose a decentralized solution based on a peer-to-peer network, namely MAAY. Peers tend to aggregate taking into account common interests and thematic similarities. Maay aims to learn the complex graph of relationships between users, documents and words. The self-adaptive network brings requested documents closer to requesters achieving reactivity and scalability.

Finally, we design a framework for experimental studies of such personalized search engines. It is based on some recent observations on the human language and user behavior. We experiment Maay under this framework and we show some promising first results.

2. Problem Formulation

2.1. Accurate Search Engines

An information space $\mathcal{D}$ is a set of documents d characterized by some words $W(d)$ in a global lexicon $\mathcal{L}$ and some owners $O(d)$ among a set of users $\mathcal{V}$. A user u only knows a part of the whole lexicon $L(u) \subset \mathcal{L}$.

The search engine is invoked by a *request* r emitted by a user $u \in \mathcal{V}$ and containing a set of words $W(r) = \{w_1, w_2, \dots w_n : \forall i, w_i \in L(u)\}$. Note that we assume that users choose the request words in their lexicon. Let $D(r)$ be the set of all documents $d \in \mathcal{D}$ such that all words in $W(r)$ are contained by d. Formally, $D(r) = \{d \in \mathcal{D} : \forall w \in W(r), w \in W(d)\}$. We note $R(r) = \{d_1, d_2, \dots d_n\}$ the set of documents returned by the search engine to the request r. The size of $R(r)$ should be fixed (typically less than 20) and all documents in $R(r)$ should also belong to $D(r)$.

We measure the efficiency of a search engine by its ability to return the documents which are the most relevant to the request and the user. We assume that there exists a function $q(d, u, w)$ which associates a document d, a user u and a word w with some value from an ordered domain which represents the *relevance* of d. For simplicity, we assume in the following that a request only contains one word. An accurate search engine should be able to ensure that the quality of a document $d \in R(r)$ is greater than any other documents $d' \in D(r) \setminus R(r)$.

2.2. Quality Approximation

We attempt now to approximate the quality of a document for a request by a graph-based approach. An information space $\mathcal{D}$ can be modeled by two bipartite graphs as illustrated in Figure 1. Let G_{SOC} be the graph linking two documents sharing one common user and G_{SEM} be the graph linking two documents sharing one common word. Formally:

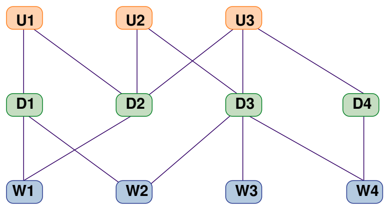

Figure 1: user-doc and doc-word bipartite graphs

$$\exists(d, d') \in E_{SOC} \Leftrightarrow O(d) \cap O(d') \neq \emptyset$$
$$\exists(d, d') \in E_{SEM} \Leftrightarrow W(d) \cap W(d') \neq \emptyset$$

An obvious way to compute the *absolute distance* between two documents d and d' consists of computing the total number of paths of length j in a graph G between d and d'. We note this absolute distance $npath_G(d, d', j)$. This distance has no meaning if it is not related to the total number of paths of length j from node d to the others nodes e in G. So, we define the *semantic distance* $SEM(d, d')$ and the *social distance* $SOC(d, d')$ between two documents d and d' as follows:

$$SEM(d, d') = \sum_{i=1}^{x} \frac{npath_{G_{SEM}}(d, d', i)}{\sum_e npath_{G_{SEM}}(d, e, i)}$$
$$SOC(d, d') = \sum_{i=1}^{x} \frac{npath_{G_{SOC}}(d, d', i)}{\sum_e npath_{G_{SOC}}(d, e, i)}$$

Behind the computation of these distances, we aim to define a global relation linking one user, one word and one document. We argue that the semantic meaning of a word w for a user u can benefit from the fact that u owns at least one document d' containing w. If d is semantically close to d', the document is certainly semantically relevant for u. Moreover, if the users which are socially close to u own this document d, there is less doubt that d is a document of interest for u. That is, we define the quality of a document d for a user u and a word w as:

$$q(d, u, w) = \sum_{d' \in \mathcal{D}(u) : w \in d'} (SEM(d, d') + SOC(d, d'))$$

To our knowledge, the quality of a document for a user and a word has never been presented with such a graph-based definition. As well, we have no knowledge of any formal definition of a *personalized* search engine.

3. Maay System

We describe in the following the Maay System, a decentralized personalized search engine. To provide personalized answers, Maay nodes learn locally from their previous interactions, the profile of others nodes and raise words characterizing documents. Using these profiles, requesters can choose nodes to query and rank documents according to the requester profile. So each user can have personalized results with local criteria of relevance.

We first review some studies related to Maay, then we present the Maay protocol.

3.1. Related Works

Several approaches have been recently proposed to improve search efficiency in unstructured information spaces [20, 3, 13, 23]. The idea consists of choosing neighbors which frequently answer to requests instead of picking them randomly. In this partially semantically structured networks, the probability to discover a file matching a query substantially increases. Although these techniques succeed in improving search efficiency, no response ranking is provided to users.

Some distributed systems [4, 21, 22, 15] implement document ranking using automatic weight computation of words in document. Unfortunately, they assume that the relevance of a document is only based on words frequency.

Another system [15] goes further by using aggressive caching of results to make nodes specialize by gathering document descriptions and nodes that are close to their topics of interest. Search requests are routed to nodes specialized in the words of the query and results are ranked on words frequency basis.

Other systems use feedbacks of users to compute the relevance of documents. For instance, the Neurogrid bookmarking system [12] uses implicit recommendations of users to semantically associate neighbors and URLs with a set of weighted words. These associations are then used to semantically route requests and rank responses.

However all the relevance criteria defined in these systems do not take into consideration personalization, in other words, including the user characteristics in the ranking process. In an information space as large as the web, providing personalized ranking of responses to each user is clearly an issue. And this is currently the problem MAAY tries to address.

3.2. Model and Definition

A MAAY *node* is a process having communication capabilities and being associated with a user. Nodes can search, download and publish documents. A node b is an *acquaintance* (or friend or neighbor) of a node a when a is able to send a message to b. We note $\mathcal{N}(a)$ the set of acquaintances of node a.

A document has a unique identifier generated by applying a hash function on its content. A node stores the documents it has published or downloaded and make them retrievable by others. Downloaded documents can come from others nodes or from web pages visited by the user through its browser. We note $\mathcal{D}(a)$ the documents stored by a node a. These documents are indexed by words and match a query if they contain all the words of the query.

In MAAY, each node maintains a database updated at every event it is aware of. Thus, a message transiting or issued by a node may add, modify or discard an entry in the database. Although, memory space is not an issue, we limit the knowledge of a node, so only fresh information which are relevant for the user are kept.

3.3. Protocol

The MAAY protocol contains two messages devoted to search documents. A `request` message contains a `query id`, the query composed of a set of words and some parameters to control the propagation of the request. A response of a request is a `response` message. This message contains a `query id` and gives meta-data (title, excerpt, ...) of documents matching the query, the *scores* of words describing the document profile and the document providers. We detail further the concept of word scores.

Each search is uniquely identified by a `query id` and all request messages and responses messages issued from the same search have the same `query id`.

A node initiating a request is called an *initiator*. From the point of view of a node receiving a request, the request emitter is a *requester*. This requester is either the initiator itself, or a node which forwarded an incoming request.

The *propagation control parameters* aim to avoid the well-known broadcast storm effect. The `ttl` parameter counts the number of remaining hops of the message. It is decremented after each hop and the message is not forwarded when equal to 0. The `fnc` parameter sets the number of neighbors to forward the request. It is a not-null integer halved after each hop.

Upon reception of a `request` message, the node a first scans the set of documents it has published, downloaded or just observed in previous search responses, then extracts the list of documents which are the most relevant for the requester and the queried words. The selection of these documents are detailled in Section 3.5. The node a stores this list in a buffer $\mathcal{B}$ and sends it back to the requester in a "`response`" message. Each document in $\mathcal{B}$ contains the meta-data of the matching documents, scores describing the document profiles and the providers of the documents.

If the `ttl` is not null, the node a should forward the request. The node stores information related to the search for further use: `query id`, requester, $\mathcal{B}$. Then it selects a list of neighbors to forward the request. This list is chosen according to their profile similarities with the requester and to their interests for the queried words, this will be detailed in Section 3.4.

Upon reception of a `response` message from the node b, the node a first retrieves all information previously stored that is related to the search identified by the `query id` : requester, $\mathcal{B}$. Then, it merges $\mathcal{B}$ with the responses contained in the response from b. As a may forward the request message to several neighbors, it may receive several `response` messages. It should obviously not forward all these messages, but only "complete" the response it previously sent by forwarding to the requester only new relevant documents.

Responses appear for the initiator as a sorted list of document descriptions with the scores describing the documents and some links toward document providers.

3.4. Using Neighbor Profiles to Route Requests

The idea behind the profile-based routing is to select, among the numerous acquaintances, a small set of nodes having both similarities with the requester and interest for the queried words.

A node a has four ways to notice that a neighbor b *claims* its interest for a word w: (i) b sends a search request on w, (ii) b sends back a search response on w, (iii) b is a document provider for a search on w, and finally (iv) node a provides to b a document on w. We note $claim_a(b,w)$ the number of claims for word w expressed by the node b and noticed by a.

We use the claims to determine the *semantic profile* of a node b, from a's point of view. Let $\mathcal{W}(a)$ be the set of words known by the node a. This set contains the words occurring in the documents known by a and the words for which a has a knowledge through previous incoming messages. We note the semantic profile of b, for a, as $\overrightarrow{b_a} = \{claim_a(b,w) : w \in \mathcal{W}(a)\}$. The vector $\overrightarrow{b_a}$ has n dimensions where n is the number of words in $\mathcal{W}(a)$. Note that nodes do not treat the same messages, nor use the same lexicon, so two nodes a and c might have a totally different semantic profile for a common neighbor b.

A node a measures the profile similarities between two nodes $b, c \in \mathcal{N}(a)$ by computing the *affinity* based on the cosine similarity between their vectors.

$$AFF_a(b,c) = \frac{\overrightarrow{b_a} \cdot \overrightarrow{c_a}}{\|\overrightarrow{b_a} \cdot \overrightarrow{c_a}\|}$$

The profile-based routing should also take into account node interests for the queried words. We distinguish two notions. From a's point of view, the *specialization* of b for a word w, denoted $SP_a(b,w)$, quantifies its part of interest for w compared to its interest for the other words in $\mathcal{W}(a)$. The *expertise* of b for w, noted $XP_a(b,w)$, measures the part of claims expressed by b for w compared to the other neighbors in $\mathcal{N}(a)$.

$$SP_a(b,w) = \frac{claim_a(b,w)}{\sum_{w' \in \mathcal{W}(a)} claim_a(b,w')}$$
$$XP_a(b,w) = \frac{claim_a(b,w)}{\sum_{n \in \mathcal{N}(a)} claim_a(n,w)}$$

When a node joins the system or has a brutal interest for a new topic, it might not know document matching its queries nor neighbors interested by the queried words. A solution consists of routing queries toward neighbors interested with words that are lexically close to the queried words. The lexical proximity of w and w' is usually measured by the Levenshtein distance $d(w,w')$ *i.e.* by counting the number of insertions, deletions or replacements of characters to change w into w'. Thus, the distance between a singular word and its plural form or words with the same etymology root is small. Therefore, nodes tend to specialize on words that are lexically close to their most interesting words. It results that these nodes acts as relationship enablers between nodes that did not previously know each other.

It is now possible to define the *routing score* of a node b for a request from z containing the word w. This routing score depends on the node a computing it :

$$NRS_a(z \rightarrow b, w) = \sum_{w' \in \mathcal{W}(a)} \frac{SP_a(b,w') * XP_a(b,w')}{2^{d(w,w')}} * AFF_a(z,b)$$

The search request is forwarded to `fnc` neighbors, most of them are chosen among the top ranked neighbors and the remaining part is chosen randomly to give the opportunity for new nodes to get known. The `fnc` factor is halved at each step, so that most queried nodes are in the close neighborhood - and so have close profile - of the requester.

The task of computing the NRS and the $affinity$ may be very costly. However, this can be reduced using three optimizations. First, we can only consider words having high value of specialization for the neighbor or the node. Second, by periodically evicting neighbors which have less affinity with the node and/or using a *Least Recently used* policy. Third, by only considering words close to w for the computation of NRS.

3.5. Using Document Profiles to Rank Results

We describe now the mechanisms to rank documents according to their relevance to a query. A node a is able to establish the profile of a document $d \in \mathcal{D}(a)$ by measuring the number of times d has been *downloaded*, and by observing the words associated with the download requests. We note $vote_a(d,w)$ the number of download requests received by a for the document d and the word w.

A document d is *relevant* to a word w if most of the votes for d are associated with w. A node a quantifies the relevance $REL_a(d,w)$ of w for d as the ratio of the votes for w to the votes for other words. On the other side, a document d is *popular* for a word w if most of the received votes for w are associated with d. The popularity $POP_a(d,w)$ is computed with the ratio of votes for d to the votes for another documents.

But, as pointed out in [12], a document receiving numerous votes has a more accurate measure of relevance than another which receives few votes. To prevent such bias, we use the Hoeffding Bound [8] to compute the maximum deviation $\epsilon(n)$ of the ratio from its expected value for n votes. So, it is possible to compute the pessimistic value of the relevance and the pessimistic value of the popularity.

$$REL_a(d,w) = \frac{vote_a(d,w)}{\sum_{w' \in \mathcal{W}(a)} vote_a(d,w')} - \epsilon(\sum_{w' \in \mathcal{W}(a)} vote_a(d,w'))$$

$$POP_a(d,w) = \frac{vote_a(d,w)}{\sum_{d' \in \mathcal{D}(a)} vote_a(d',w)} - \epsilon(\sum_{d' \in \mathcal{D}(a)} vote_a(d',w))$$

As the popularity of old documents increase, newly published documents may never be found. So, we implement an aging mechanism by periodically multiplying the number of votes for a document by an aging factor (*i.e.* 0.99).

The request initiator should also be considered. If a word w is relevant to a document d and if the user z is a specialist of w, we guess that the document d will be of interest for z. So we obtain the following computation of the *matching* between a user and a document:

$$matching_a(z,d) = \sum_{w \in \mathcal{W}(a)} REL_a(d,w) * SP_a(z,w)$$

Finally, the node a is able to compute the *document ranking score* of a document d for a queried word w and a requester z as:

$$DRS_a(z,d,w) = REL_a(d,w) {*} POP_a(d,w) {*} matching_a(z,d)$$

Nodes also use informations contained in the search responses to establish a partial profile of some known but not stored documents. The accuracy of this partial profile depends on the amount of informations in the search response. There is a trade-off between the compactness of the search responses and the accuracy of the profiles.

Using the document ranking score, a node can select, from its point of view, the most relevant document for the requester. The selection is done among the documents published, cached or known through previously received search responses.

4. Experimental Study

Two reasons lead us to propose a framework for an experimental study rather than relying on real-trace simulations. At first, most of the available real traces are collected from peer-to-peer systems. Yet, the information space of these systems are mostly related to multimedia files, while we rather focus on rich queries in full-text. Secondly, observations give us the conviction that the studies of user behavior in centralized systems [19, 11] are biased by the system itself. Indeed, a European fan of soccer modifies its query because he *knows* that typing "football" in a centralized system results in web pages on NFL.

We would naturally prefer to provide proofs of Maay correctness. But, the quality of results only depends on the approximation of the initiator profile computed from observation of the history of all transactions occurring in the network. We guess that simulations will provide better hints of the validity of the concepts behind Maay.

Our approach is straightforward. We first present a framework that can be used in the future by other similar systems. We argue that this system provides an accurate model of users requesting and downloading documents in an information space. Then, we study the behavior of Maay.

4.1. A Framework for Experimental Studies

The model of users is based on two bipartite graphs (see Figure 1). The first one links users with documents, while the second one connects documents and words. The construction relies on two assumptions.

The first one, related to the human language, has been partially established by several studies [1,24] since the pioneering work of [9]. The graph of word interactions is constructed by linking two words when they co-occur in a sentence (*a fortiori* in a file). The study of the properties of these graphs shows some features recently found in numerous complex systems. They first exhibit the *small world* effect, characterized by a high clustering and a small diameter. Moreover, these graphs are *scale free*. Their distribution of degrees follows a power law: few words are linked to a large number of words while most of them have very few connections.

The second assumption follows recent observations on several peer-to-peer systems [10, 5]. The data-sharing graph is built by linking two users when they share a same file. Observations on several real traces show that this graph also exhibits the small-world effect and the scale-free distribution of degrees.

As both bipartite graph should exhibit the same properties, we use the same construction based on [7]. The principle is simple: each user and each document chooses its degree at random with respect to a power law distribution. Then, connections between documents and users are randomly chosen until their degrees match.

However, a uniform choice of links between documents and users results that no set of documents co-occurs more frequently than others. To solve this problem, we artificially introduce *groups*, which can be seen as cultural, linguistic or geographic clusters.

In the web, there are far more documents written in English than in Spanish or in French. The same fashion is observed for Internet users. So, we choose to create groups with different sizes. However, we assume that the number of words commonly used in each group are roughly the same. These groups are not exclusive and some users, documents and words might belong to several groups. Yet, the most popular group is still more attractive.

Then, the user behavior should be modeled. We assume that users do not choose the words of their queries at random in an universal lexicon. Rather, we expect that queried words depend on user profile. So, queries are constructed according to word occurrence in both stored documents and in the whole set of documents. The number of words by query is chosen following observations of [19].

The system initially contains only two nodes and two documents, but it grows. The inter-arrival period is inversely proportional to the number of nodes already connected. Therefore, the number of nodes within the system grows exponentially, which is conform to the assumption that the utility of a network scales with its size [18]. The execution of a simulation consists of consecutive ticks. At each tick, the nodes process incoming messages. They periodically emit a request and choose a document to download among the responses they receive.

4.2. Maay First Results

We use the framework for experimenting Maay.

Parameters of the simulation (see Table 1).

users	1500
documents	3000
words	500
groups	5 (A, B, C, D, E)
users per group	A:500, B:400, C:300, D:200, E:100
docs. per group	A:1000, B:800, C:600, D:400, E:200
words per group	A:100, B:100, C:100, D:100, E:100
users per doc.	1-200 (avg:2.5) [power-law dist.]
words per doc.	5
docs. per user	1-40 (avg:5) [power-law dist.]
docs. per word	1-1000 (avg: 30) [power-law dist.]
initial `ttl`	2
initial `fnc`	4
nei. profile handled	50
cached hit stored	20
downloaded doc. stored	20
words used by query	1-3 [34%: 1, 41%: 2, 25%: 3]
query rate	1 / 100 ticks / user
publication rate	1 / 25000 ticks / user
node join rate	1 / 50000 ticks / user

Table 1: Parameters of the Experimental Study

Nodes join the network and documents get published until the system reaches 1,500 users and 3,000 documents. Each search reaches a maximum of 12 nodes and has in average 36 matching documents. The number of documents in responses is restricted to 5. Nodes only know a small part of the network: each node handles 50 neighbor profiles and 40 document profiles. The eviction policy is based on affinity and matching with the user.

Node aggregation by common profiles. A graphical overview of the system is obtained by using *graphopt* (`http://www.schmuhl.org/graphopt/`), a tool that puts in evidence clusters by applying physical principles of repulsion and attraction to nodes. We link each user to the five users which shares the most documents in common. As expected, Figure 2 on the left shows five clusters representing the five groups. We then compare it to the Maay network in which each node is linked to the five neighbors having maximal affinity. We infer that Maay succeeds in learning the complex graph of user relationships. Another interesting measure validates this fact. In average, each node has 9.3 neighbors which belongs to the same group among its top ten neighbors having maximal affinity.

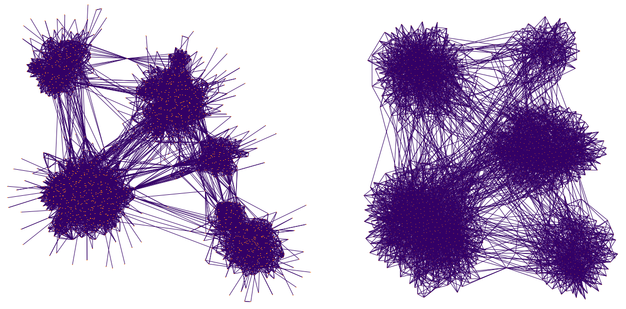

Figure 2: On the left, the graph of users sharing documents. On the right, the Maay graph at the end of the simulation. In both cases, links occur between top five neighbors.

Efficiency of routing and ranking. For each request r, we determine $B(r)$, the five documents having the best value of quality in the whole information space. Following, we compute the ratio of responses containing at least k documents belonging to $B(r)$. We study the protocol with and without the document ranking and the neighbor selection. Results are shown in Table 2.

doc. rank. pol.	nei. sel. pol.	$k = 1$	$k = 2$	gain
random	random	0.20	0.02	x 1.00
profile-based	random	0.27	0.03	x 1.35
random	profile-based	0.70	0.45	x 3.50
profile-based	profile-based	0.85	0.64	x 4.25

Table 2: Effect of the document ranking and the neighbor selection policies on result relevance

We mostly show that the search engine benefits from both policies, so it validates the interest of the approach we propose in Maay. Especially, among the five documents occurring in the response, Maay gives at least one of the five top documents in 85% of cases.

Queried word selection policy change. Finally, we simulate a user abruptly changing the set of queried words. The default policy favors words which are popular in the documents corpus of the initiator, but seldom in the whole information space. But, for instance, a user might be interested with words rare in its documents but quite popular elsewhere. Therefore, we define nine policies that considers the frequency of the queried words in the corpus of the initiator and in the whole information space. The policies can favor words which have high frequency (*popular*), low frequency (*rare*) or randomly chosen (-). We first let the system grows. Then we change the queried word selection for a small set of 15 users and we monitor the responses they receive. The responses are exhibited in Table 3. Results shows that Maay reacts quite well since for most policies, the relevance of results is higher than 0.75.

user local word	global word	$k = 1$	$k = 2$	$k = 3$
-	-	0.74	0.45	0.24
popular	-	0.76	0.42	0.22
rare	-	0.75	0.47	0.26
-	popular	0.66	0.27	0.10
-	rare	0.86	0.64	0.39
popular	popular	0.82	0.48	0.21
popular	rare	0.86	0.61	0.38
rare	popular	0.69	0.28	0.11
rare	rare	0.85	0.63	0.39

Table 3: Effects of word selection policy change on relevance of results

5. Conclusion

This paper focuses on decentralized personalized search engines. First, we formulate the problem and we propose a graph-based measure of quality of a document for a user and a word. Then we propose Maay, a decentralized system which makes use of previous interactions in order to route request messages and answer back with top-relevant documents. Finally, we propose a framework for experimental studies of such system. We use it to show that the main principles of Maay could benefit to users.

An early version of Maay is under way and might be available in open-source at the end of this year. Maay will include indexation of visited web pages by users through their browsers and tagging of documents. As the Web might be collaboratively crawled and indexed, especially sites of interests, search results might be more up-to-date and relevant than centralized search engine. In addition, deep web might be partially indexed, and searches in the past Web can become possible. Finally, the natural replication of data by downloaders increases their availability avoiding providers of popular data to be overloaded.

In parallel, we will continue to improve and enrich the user model through the observation of the behavior of real users. Confronting Maay to assorted scenarios, both simulated and real, will help in building a suitable system that could be used at Internet scale.

References

[1] A.-L. Barabási. *Linked: The New Science of Networks.* Perseus Publishing, 2002.

[2] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 1998.

[3] V. Cholvi, P. Felber, and E. W. Biersack. Efficient Search in Unstructured Peer-to-Peer Networks. In *ACM Symposium on Parallel Algorithms (SPAA'04)*, 2004.

[4] F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen. PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. *International Symposium on High-Performance Distributed Computing (HPDC-12 2003)*, pages 236–249, 2003.

[5] F. L. Fessant, S. B. Handurukande, A.-M. Kermarrec, and L. Massoulié. Clustering in Peer-to-Peer File Sharing Workloads. *3rd International Workshop, IPTPS 2004*, 2004.

[6] A. Gai and L. Viennot. Broose: A Practical Distributed Hashtable Based on the De-Bruijn Topology. In *Int. Conference on Peer-to-Peer Computing (P2P 2004)*, 2004.

[7] J.-L. Guillaume and M. Latapy. Bipartite Graphs as Models of Complex Networks. In *Workshop on Combinatorial and Algorithmic Aspects of Networking (CAAN'04)*, 2004.

[8] W. Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 301:13–30, 1963.

[9] R. F. i Cancho and R. V. Solé. The Small World of Human Language. *The Royal Society of London*, 268(1482):2261–2265, 2001.

[10] A. Iamnitchi, M. Ripeanu, and I. T. Foster. Small-World File-Sharing Communities. In *IEEE INFOCOM*, 2004.

[11] B. J. Jansen and A. Spink. An Analysis of Web Documents Retrieved and Viewed. In *International Conference on Internet Computing (IC'03)*, 2003.

[12] S. Joseph. NeuroGrid: Semantically Routing Queries in Peer-to-Peer Networks. In *Int. Workshop on Peer-to Peer Computing (IPTPS'02)*, 2002.

[13] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti. A Local Search Mechanism for Peer-to-Peer Networks. In *Int. Conf. on Information and Knowledge Management (CIKM'02)*, 2002.

[14] J. M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5):604–632, 1999.

[15] A. Z. Kronfold. *FASD: A Fault-Tolerant Adaptive Scalable Distributed Search Engine.* PhD thesis, Princeton University, 2004.

[16] J. Li, B. T. Loo, J. Hellerstein, F. Kaashoek, D. R. Karger, and R. Morris. On the Feasibility of Peer-to-Peer Web Indexing and Search. In *Int. Worskhop on Peer-to-Peer Systems (IPTPS'03)*, 2003.

[17] P. Maymounkov and D. Mazières. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Int. Workshop on Peer-to-Peer Systems (IPTPS'02)*, 2002.

[18] D. P. Reed. That Sneaky Exponential. Beyond Metcalfe's Law to the Power of Community Building. http://www.reed.com/Papers/GFN/reedslaw.html, 2000.

[19] A. Spink, D. Wolfram, B. J. Jansen, and T. Saracevic. Searching the Web: the Public and their Queries. *American Society of Info. Sci. and Tech.*, 53(2):226–234, 2001.

[20] K. Sripanidkulchai, B. M. Maggs, and H. Zhang. Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. In *IEEE INFOCOM'03*, 2003.

[21] C. Tang and S. Dwarkadas. Hybrid Global-Local Indexing for Efficient Peer-to-Peer Information Retrieval. In *Networked Systems Design and Impl. (NSDI'04)*, 2004.

[22] C. Tang, Z. Xu, and M. Mahalingam. pSearch: Information Retrieval in Structured Overlays. *Computer Communication Review*, 1:89–94, 2003.

[23] S. Voulgaris, A.-M. Kermarrec, L. Massoulié, and M. V. Steen. Exploiting Semantic Proximity in Peer-to-Peer Content Searching. In *Workshop on Future Trends of Distributed Computing Systems (FTDCS'04)*, 2004.

[24] D. J. Watts. *Six Degrees: the Science of a Connected Age.* W. W. Norton, 2003.