Appl Intell
DOI 10.1007/s10489-010-0229-0

A large-scale distributed framework for information retrieval in large dynamic search spaces

Eugene Santos, Jr. · Eunice E. Santos · Hien Nguyen · Long Pan · John Korah

Abstract One of the main problems facing human analysts dealing with large amounts of dynamic data is that important information may not be assessed in time to aid the decision making process. We present a novel distributed processing framework called Intelligent Foraging, Gathering and Matching (I-FGM) that addresses this problem by concentrating on resource allocation and adapting to computational needs in real-time. It serves as an umbrella framework in which the various tools and techniques available in information retrieval can be used effectively and efficiently. We implement a prototype of I-FGM and validate it through both empirical studies and theoretical performance analysis.

E. Santos, Jr.
Thayer School of Engineering, Dartmouth College,
8000 Cummings Hall, Hanover, NH 03755, USA
e-mail: Eugene.Santos.Jr@dartmouth.edu

E.E. Santos · J. Korah
Department of Computer Science, University of Texas at El Paso,
Room 234, Computer Science Building, 500 West University
Avenue, El Paso, TX 79968-0518, USA

E.E. Santos
e-mail: eesantos@utep.edu

J. Korah
e-mail: jkorah@utep.edu

H. Nguyen (✉)
Mathematical and Computer Sciences Department, University of
Wisconsin, 800 W. Main street, Whitewater, WI 53190, USA
e-mail: nguyenh@uww.edu

L. Pan
Department of Computer Science, Virginia Polytechnic Institute
& State University, 660 McBryde Hall,
Blacksburg, VA 24060, USA
e-mail: panl@vt.edu

Keywords Information search and retrieval · Distributed processing · Multi-agent architecture · Dynamic anytime processing · Content analysis and indexing

1 Introduction

One of the challenging problems that people face with today's information technology is the difficulty of retrieving *timely* relevant information from a huge amount of information that is constantly and rapidly changing. Due to the revolutionary advances in communication systems, database connectivity, and the Internet, we can have instant access to enormous and diverse sources of information. This has changed the way a user accesses and uses information in his/her personal and professional life. Most users now are accessing databases that are large and dynamic in nature. Recently, the Internet has not only seen a tremendous growth, but also a dramatic change, in content (e.g. as cited in [1], the surface Web is growing at a rate of 7.5 million documents per day). The ascendency of new trends such as Web 2.0 and e-commerce has led to the establishment of a multitude of businesses involved with new media (e.g. blogs, RSS, social networking, iReport[1]), auctions, financial markets, and so forth. This has, in turn, led to a dramatic increase in the rate at which data is being added to the Internet, making it an extremely dynamic place. Additionally, users have come to depend on the Internet for their up-to-date information needs such as news, stock prices, sports, and entertainment, which has created the need for "real time search" capabilities. This is especially true for professionals and analysts working in critical areas such as disaster

[1]http://www.ireport.com/

relief and homeland security. For example, analysts working on natural disasters such as the 2010 earth quake in Chile, or the 2004 Tsunami in Asia, would need to get access to all relevant reports from all available channels such as network news, formal government announcements, and personal blogs as fast as possible. The information needs to be *continuously* updated on a *very short-time interval* (say, 5 minute intervals). The existing techniques in information retrieval and Web 2.0 could not meet this demand because they only either offer one time retrieval or provide feeds from other web sites based on static, pre-defined keywords [34]. We need to have a more sophisticated tool that can dynamically change over time and offer the timely updates of the content automatically. The urgent need for sophisticated software tools to help human analysts was also highlighted by the 9/11 commission that blamed the poor information technology infrastructure available to analysts including the unavailability of large amount of data collected by intelligence agencies [39].

Even though researchers from various disciplines (Computer Science, Information and Library Sciences, etc.) have tried to tackle this problem by addressing the individual factors that contribute to the common problem, as yet, a unified framework that allows different methods and mechanisms to work together effectively is still missing. Therefore, wheels are being reinvented and the results of such research still have not fully addressed the problem. For example, the information retrieval community has focused on improving the retrieval methods on large, heterogeneous data sources [12] while the parallel and distributed computing research has focused on different techniques to optimize the use of resources [15]. Unfortunately, we have not been able to find any work that bridges these two fields. This gap is a stumbling block to achieving a solution for real-time retrieval because we have to simultaneously deal with large amounts of fast changing data and the time constraints imposed by a user's needs. Current methodologies have tried to solve this by indiscriminately adding computational resources leading to the creation of large data centers and supercomputing facilities. However, the growth rate of the Internet and private databases continue to outstrip resources. Therefore, we need to integrate smart resource allocation algorithms that scale well with ever expanding search spaces. To summarize, there is a need for a well-defined cross-disciplinary framework that is capable of adapting, in real-time, to dynamic information content while addressing the problem of computational resource allocation.

In this paper, we present such a large-scale distributed real-time framework, called the anytime Intelligent Foraging, Gathering, and Matching (I-FGM) system [27–29, 31]. We integrate the knowledge representation techniques from Artificial Intelligence (AI) and algorithms from Distributed and Parallel computing to efficiently allocate and deploy computation resources in a unified framework. The central goals of this framework are quick retrieval of relevant results in large dynamic search spaces, easy incorporation of new retrieval technologies, and capability for employing multiple retrieval algorithms in tandem. To build such a framework, we need to address two main issues which are efficient resource allocation and unified representation of documents from different resources. We employ efficient resource allocation strategies within I-FGM by using a multi-agent architecture with a modular structure and a partial information processing strategy to reduce the time taken to retrieve information. In order to deal with data heterogeneity, we use a common unifying knowledge representation in the form of concept graphs to represent documents retrieved from different sources. The multi-agent technology allows rapid allocation and deployment of resources in a parallel distributed environment. It also enables us to employ sophisticated resource allocation principles. We believe that the Achilles' heel in current retrieval technologies is its requirement to fully process a document before its relevancy can be measured. We term such systems as full-processing systems. In the full-processing systems, the resource allocation strategies that can be applied is severely limited. We propose a paradigm shift in the way documents are processed by introducing partial processing—incrementally processing documents in multiple steps and calculating their partial relevancy or approximation of the final relevancy after each step. This partial relevancy helps prioritizing the allocation of computational resources to documents that have 'potential'. This strategy of prioritizing documents helps in the efficient use of limited computational resources. As we shall see, this strategy is crucial to dealing with a rapidly changing space of information in real-time. It also helps in delivering a rough set of results to the user quickly and gradually refines these results. Our experimental section shows that a dynamic resource allocation strategy based on partial processing is more effective than current strategies that range from random allocation to static resource allocation. We further bolster these results with a theoretical analysis of the full and partial processing systems.

Furthermore, the multiagent architecture has a natural modularity that allows for the rapid assimilation of third party tools required for accessing and processing information from different sources in different ways. This modularity enables deployment on a wide variety of grid computing platforms/architectures [11].

Finally, the common knowledge representation in I-FGM captures document information retrieved by any third party tool based on the representation of *document graphs* [24–26] representation. Document graphs are graph based structures where nodes and edges represent the concepts in a document and the relationships among them. Having a common representation in our framework allows for the processing and display of results from diverse search techniques on a unified ranking system. Although this paper deals with text retrieval, I-FGM and document graphs have also been successfully used with other types of data such as images [31].

In order to validate I-FGM, a prototype has been built with the Internet as the target search space with various search engines based on third party tools. In order to duplicate the dynamic conditions that some users like intelligence analysts often face, we provide a dynamic simulation environment where documents enter I-FGM at different rates. The validation is provided through comparison with control systems that reflect the current paradigms in information retrieval. Additionally, we show through analytical modeling that our resource allocation strategy based on the partial processing paradigm is better than conventional methods.

In what follows, we describe the related work and point out the challenges we faced while designing I-FGM. We also provide background on document representation within our I-FGM framework. Next, we present the framework in detail, followed by a description and analysis of our validation experiments. We then provide theoretical analyses of system performance for I-FGM to better explain the observed performance. Finally, we present our conclusions and discuss future work.

2 Related work and background

2.1 Related work

In this section, we review the existing approaches to retrieving information in a distributed setting and the existing architectures for distributed information retrieval (distributed IR).

We are primarily interested in being able to employ and adapt existing state-of-the-art information retrieval technique into our framework to enable retrieval in large dynamic search spaces. As such, we are not focusing on developing a new retrieval method. While most typical document retrieval techniques (such as Boolean, keyword-based matching, and probabilistic information retrieval, to name a few) have relatively low computational requirements [40, 41], they still do not address the problems arising from the inherent dynamics of the information space. The existing approaches to retrieving information such as filtering [13, 21], taxonomies [4, 5, 36], user modeling [6], and document clustering [2, 37], cannot achieve our objectives for quickly obtaining accurate results in dynamic situations. The current search engines use pre-processing techniques such as indexing web pages to achieve quick returns to the queries. Indexing is only done periodically and typically not in real-time. This may work for predominantly static information but will fail in situations such as real-time intelligence or disaster relief operations where most of the information is being quickly produced and changing rapidly, e.g., streaming data [7]. Since such databases are dynamic with new documents coming in and older documents being updated by new real-time information, the indexing cycle must be continuous.

Distributed IR is often considered a solution for retrieval in large databases (see [8] for more discussion on distributed IR). Resource allocation is typically treated as the simple “distribution of labor” or “partitioning of indexes” methods. Unfortunately, dynamic databases are changing at a much faster rate than processor speeds are capable of handling. One straightforward solution (which incidentally is the most popular method in distributed retrieval methods) is to consider a document to be an indivisible unit and completely process it before moving on to the next. This limits the effectiveness of potential methods in dynamic search spaces and thus, does not adequately address our challenge.

Although a number of architectures have been suggested for distributed IR, there has been little work in anytime information retrieval architectures for dynamic databases and information spaces. The Harvest system [3], for example, is an architecture suggested for distributed web searching. The main difference between I-FGM and Harvest is that the Harvest system does not have an intelligent resource allocation strategy which can improve system efficiency over time and imbue the system with the ability to bring new relevant information to the attention of the analyst in a timely manner. ACQUIRE [9] is a powerful system designed for retrieving heterogeneous data from distributed databases. The system can dynamically change its retrieval strategy according to retrieval factors such as network speed, computational capabilities. However, it does not take the dynamism of data into account.

2.2 Background

In I-FGM, we represent a document as a document graph (DG), instead of a typical vector [22]. *DG* is a directed acyclic graph that consists of concepts nodes and relation nodes. The main difference between the *DG* representation and a vector space model is that we further explore the syntactic structure of sentences in a document to establish the relationships among the concepts. The *DG* representation has been successfully used in the development of a cognitive user modeling technology to enhance the performance of an IR system [24–26]. This cognitive user modeling technology has been shown to help the target IR system to retrieve more relevant documents earlier when compared to the traditional vector space model approach in evaluations with testbed collections found in the IR community such as CRANFIELD, CACM, and MEDLINE [18–20]. More importantly, it also helps human intelligence analysts to retrieve more *uniquely* relevant documents as compared to keyword-based retrieval systems as demonstrated in an experimental evaluation with human analysts at the National Institute of Standard and Technology [30]. Given the above user modeling technology for information retrieval, the I-FGM framework is a natural complement.

In I-FGM, to make comparisons between a query and a document possible, we represent queries also as graphs and we refer to them as query graphs (*QGs)* in this paper. Basically, a QG has similar structure as a DG and is constructed from a user' query in the same way that a DG is constructed from a document. The construction of a *DG* is an automated process as follows:

(1) decomposing a document from plain text format into sentences;
(2) parsing each sentence using Link Parser [33];
(3) extracting noun phrases (NPs) from the parsing results; and,
(4) generating relations between concepts/entities based on heuristic rules (see [23] for details).

The most computationally costly step in this process is parsing the sentence, with a complexity of $O(m^3)$ where m is the number of words in a sentence [33]. A simple query and its corresponding *DG* are shown below in Fig. 1. The query "*Asian Tsunami disaster in South East Asia*" contains two main NPs and one prepositional phrase (PP). The "related-to" link between the node "Asian Tsunami disaster" and the node "South East Asia" is generated using a prepositional phrase heuristic while the remaining "is-a" links between nodes are generated from a noun phrase heuristic [23].

A *match* (*or similarity measure*) between a query q and a document d is defined as:

$$\mathrm{sim}(q,d) = \frac{n}{2*N} + \frac{m}{2*M} \tag{1}$$

where n, m are the number of concept and relation nodes of q matched in d, respectively, and N, M are the total number of concept and relation nodes in q. If a labeled concept node c occurs in both q and d, then c is called a match. Two labeled relation nodes are said to be matched if and only if at least one of their concept parents and one of their concept children are matched and they share the same relation label. This measure is modified from [17]. We can compare two *DG*s that are significantly different in size (for example, a *DG* representing an entire document and another *DG* representing a user's query) by using the number of concept/entity nodes and relation nodes in the larger *DG* instead of the total number of nodes in both *DG*s.

Thus, for comparisons, we note that we are working strictly with labeled graphs, as opposed to general graph isomorphism, which avoids computational complexity. I-FGM

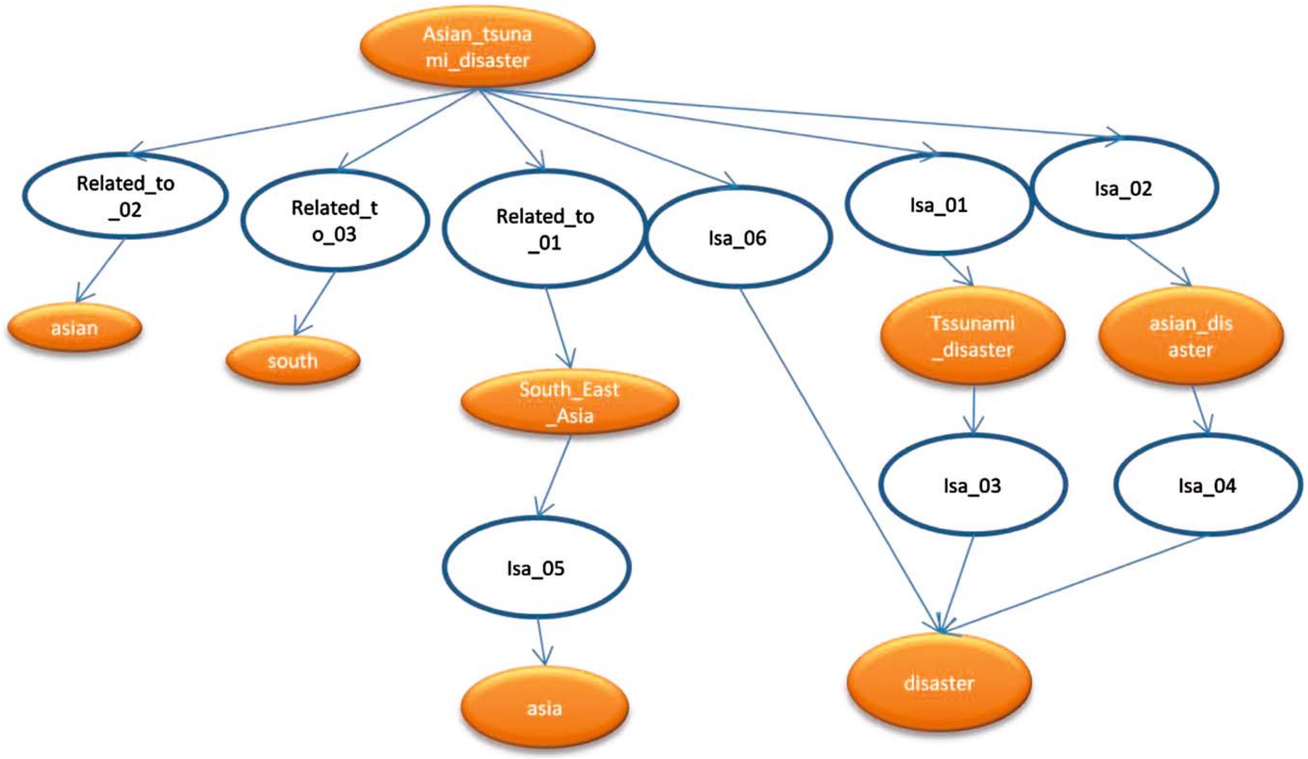

Fig. 1 Example of a document graph (DG) for the query "Asian Tsunami disaster in South East Asia"

uses DGs as the scheme for representing semantic information given their effectiveness. To re-iterate though, I-FGM allows for any modular method to be used in place of DGs.

3 I-FGM framework and system architecture

The overall goals of the I-FGM framework are to enable easy incorporation of the existing and new retrieval technologies, to allow for multiple retrieval algorithms to work in tandem, and ultimately, to quickly retrieve relevant results in large dynamic search spaces. We use popular search engines such as Google or Yahoo to do a quick, coarse-grained exploration of the search space and filter out the most non-relevant documents. The filtered search space is still large and dynamic enough to create problems with existing methodologies. The difference between our I-Foragers with popular meta-search engines [14–16, 32] is that their goals focus primarily on data coverage while our goal is the rapid retrieval of relevant documents from a dynamic information space. The documents in the search spaces are then chosen to process incrementally by an intelligence resource allocation strategy and the results are shown to the users in real-time.

The I-FGM framework consists of five major components: *I-Foragers*, *IGSoup*, *gIG-Builders*, *I-Matchers*, and the *Blackboard*. The I-FGM framework and system architecture is shown in Fig. 2.

I-Foragers (*Information Foragers*) are tasked with foraging for data from various (massive and dynamic) sources. The user's modified query is sent to the *I-Foragers*, each of which analyzes the query and selects appropriate sources/databases to collect information. *I-Foragers* gather lots of data from the Internet by taking advantages of existing search tools such as search engines. Different retrieval engines use different techniques which work better for some queries over others. To deal with these differences, we first measure the *reliability* of each engine for different types of queries and use them as guiding metrics for integrating results from the different engines. We denote A as the set of the relevant documents for a query. We denote B as a subset of A which is the set of relevant document returned by the *I-Forager*. Then reliability is calculated based on results from a number of test runs using the following equation:

$$\text{reliability} = \alpha_1 \beta \frac{n}{k} + \alpha_2 s + \alpha_3 d \qquad (2)$$

where s—average similarity measurement of documents in B, d—average difference between rank of documents of B within A and the rank returned by the I-Forager, β—ratio of the maximum number of documents gathered by any *I-Forager* to the number of documents gathered by this *I-Forager*, n—order of B, k—order of A, and α_1, α_2, α_3—scaling factors.

The reliability function measures how reliable a search engine is at any given time. It may be noted that the reliability function used here is heuristic in nature, however, we also note that the choice of a reliability function or any of the metrics listed in this paper is not fixed and can be adapted based on the designers' choice. Our design choice reflects the metrics most useful from our previous experience in information retrieval research.

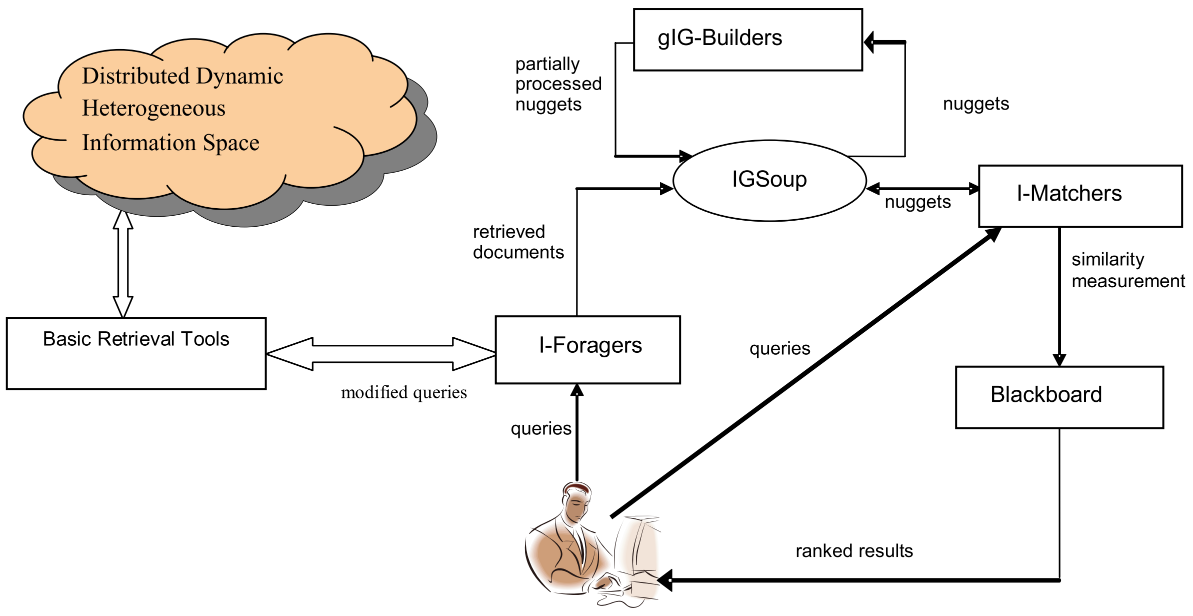

Fig. 2 I-FGM system architecture

With reliability, we can now determine our expected first-order similarity for a new document which is defined as the similarity measure that can be calculated from the *I-Forager* measurements. For each downloaded document, it is a summation of the product of the scaled reliability of the *I-Forager* and *I-Forager* measurement, which is the inverse of its rank in the result that the *I-Forager* returned, over all *I-Foragers*. For example, if document x is returned by engines a, b and c which have reliability values y_a, y_b and y_c respectively. Let z_a, z_b and z_c be the rank of the document x as given by engines a, b and c respectively. Then, the first order similarity of document x is calculated as:

$$\left(y_a * \frac{1}{z_a}\right) + \left(y_b * \frac{1}{z_b}\right) + \left(y_c * \frac{1}{z_c}\right).$$

I-Foragers perform "quick and dirty" or roughly adequate retrieval to continuously populate the common repository called *IGSoup*. The *IGSoup (Information Graph Soup)*, serves as the repository for retrieved data in which more sophisticated analyses will be performed to determine final relevancy with respect to the target query. For our testbed, as we mentioned earlier, we simply employed existing search engines to populate our space of documents. We will provide simulations with respect to our goal of working with dynamic search spaces. In particular, our target I-Foragers would be those attached to streams, blogs, etc. that rapidly come into existence, change, and can even disappear in a short amount of time.

gIG-Builders (geospatial Information Graph-Builders) are processes that extract and partially analyze information/knowledge nuggets (such as *DGs*) from documents chosen in the *IGSoup*. In this paper, we refer to knowledge nuggets as a partial piece of information. *gIG-Builders* together with *I-Matchers* are at the heart of the I-FGM framework in dealing with search space dynamics and massive size. In order to be able to partially process a document, we need a resource allocation strategy that determines the candidate documents to choose from the IGSoup. We identified two methods for *gIG-Builders* to select documents for processing. The simple way is to select the document with the highest *priority*. Each document in the *IGSoup* has an associated priority value that dynamically changes over time. Intuitively, this priority value reflects the partial relevancy value computed thus far (such as the current partial similarity/match value for *DGs*) as well as an indicator on how promising or how likely this document is among the most relevant. Hence, the priority value is a parameter used by the *gIG-Builders* in determining the candidates for future allocation of computational resources for further semantic analyses. The priority of a document d is computed as follows by the *I-Matchers* (as we shall discuss later):

$$\begin{aligned} &P_k = \beta_1 q_1 + \beta_2 q_2 \\ &q_1 = \delta P_{k-1} + (1-\delta)\mathrm{Sim}_k \\ &q_2 = \begin{cases} \Delta_k & \Delta_k > 0 \\ -\Delta_0 \frac{t_k}{t_0} & \Delta_k = 0 \end{cases} \\ &\Delta_k = \mathrm{Sim}_k - \mathrm{Sim}_{k-1} \end{aligned} \tag{3}$$

where P_k represents the priority of a document d at step k. It is computed from the document priority at step $(k-1)$, P_{k-1}, and its current similarity, Sim_k. Δ_0 is the average expected similarity for the maximum parsing time. t_k is the parsing time allocated for the document at step k. δ, β_1, and β_2 are the scaling factors.

The purpose for employing the simple function is to provide a basis of comparison with our desired second method. The second method of selection is done using a probabilistic function, biased towards higher priority documents. This function is called a *biased randomized function* and formulated as:

$$P_i = \frac{\mathrm{pri}_i}{\sum_{j=1}^{k} \mathrm{pri}_j} \tag{4}$$

where P_i is the probability for selecting document i, pri_i is the priority value of document i, and k is the number of documents contained in the selection pool. In our current implementation, k is equal to 10. Within the top k yet-to-be-finished documents according to priority, *gIG-Builders* use the *biased randomized function* (biased selection) to choose the document that will be processed. This type of selection guarantees that documents with lower priority have some chance of being selected. This is essential to handling dynamic search spaces since new documents will have had very little processing done on them and potentially have an initial low priority. Since the difference in priorities of top documents may be small, and priorities may fluctuate, the biased selection also provides a smoothing function. Specifically, for our target testbed, the *gIG-Builders* partially and incrementally build the *DG* for each document. The promising document is chosen and processing time is set based on its priority. After working on the document for the processing time assigned, they store the updated *DG* in the *IGSoup* and then choose other promising documents to work on.

For a newly arriving document, we can then set its initial priority according to two factors, one is its first order similarity described above and the other is the priority of its neighborhood. For finding the neighborhood of a new document, we rank the documents in the *IGSoup* that is brought in by the same engine as new documents according to this retrieval engine's measurement. The documents surrounding the new document in this rank order are termed as its neighborhood. The size of the neighborhood that should be considered while calculating the priority will depend on the reliability of the *I-Foragers*. For this, we rely on the following principle: the higher the reliability is, the smaller the size of the neighborhood should be. We use the following formula to calculate the initial priority of the new document:

$$P_{\text{init}} = \max(\text{first order similarity}, \text{average of neighborhood priority}) \tag{5}$$

Now on to our fourth component, *I-Matchers (Information Matchers)* compare the documents that have been partially processed by the *gIG-Builders* with the target query to obtain a similarity measure for the document and update the document rankings and priority values in the *IGSoup*. In essence, the *I-Matchers* update the current similarity value when the *gIG-Builders* continue with content analyses. In our target testbed, the *gIG-Builders* continuously extend the DGs while the *I-Matchers* update the similarity value between the *DGs* and the target query. The new priority values are then calculated by the *I-Matchers*.

The last component, the *Blackboard* continuously displays the current top results to the user/analyst based on similarity/match value. These results change over time as the processing of the documents in the system proceeds.

4 Empirical evaluation

4.1 Overview of the evaluation

The goal of I-FGM is to retrieve relevant information for the analyst fast in a dynamic environment. In order to validate this goal, we focus on evaluating how the efficiency and effectiveness of current relevance-based retrieval methods can be improved through use of I-FGM. We compare its performance to control systems that represent the current methods for dealing with large dynamic data. Our evaluation procedure also differs from standard procedures used in information retrieval systems such as TREC testbeds [38]. This is because we are not evaluating the effectiveness of our retrieval algorithm (the effectiveness of document graphs have already been discussed in earlier sections).

In our evaluation, we designed two intuitive control systems that have similar system architectures to I-FGM but differ in the manner in which they choose a document to process and consequently how the resources are allocated. Each of the control systems uses the same retrieval technique. Our task is to determine whether the I-FGM system alerts the analyst to relevant documents in a more timely fashion than the control systems. Furthermore, since the control systems use relatively straightforward approaches for resource allocation, the goal here is to determine whether the resource allocation used in I-FGM is beneficial. Thus, for our purposes, it is sufficient to choose four queries from a scenario that an analyst might encounter for our study. Experimental runs are conducted with these queries and the results are collected. Since we are concerned with getting the relevant documents quickly, we record the recall of the systems at regular intervals. Recall is defined as the ratio between the number of retrieved relevant documents over the number of relevant documents [22]. In essence, since each system will ultimately, in time, return the same collection of documents in the same ranking order, the system that returns all the relevant documents the earliest is the best.

For the purposes of our experiments, we used the Internet (World-Wide Web) as the basis for constructing our dynamic testbed. Clearly, the Internet is large and dynamic, and in addition to the proliferation of web pages, there is a substantial variance in type and scope of content. We realize that the dynamism in the Internet does not mirror the extreme conditions that a human analyst in the field must face. However, we can artificially create a highly dynamic search space by controlling the arrival of different sets of documents into the system. Since our testbed focus is on text-based information, we decided to select five popular search engines to retrieve quick and raw results as our I-Foragers: Google, Yahoo, MSN, LookSmart, and Teoma. Thus, we have five *I-Foragers*, one for each search engine.

4.2 Control systems

The two control systems used to evaluate the performance of I-FGM are the Baseline system (B system) and Partial-Intelligent system (PI system). These control systems were carefully selected to represent the conventional resource allocation strategies. It may be noted that all these control systems incrementally process the documents even when they do not use the partial relevancy values. The control systems differ from the I-FGM (or known as the fully intelligent) system mainly in the way the resources are allocated. We do not use a control system that does full-processing because the list of relevant documents in such systems can only be known after all the documents have been processed and this would unfairly penalize the full-processing systems. The Baseline and Partial-Intelligent control system utilize some of the resource allocation used in the full-processing systems. The Baseline system does a random selection of the documents and allocates it a constant amount of resources. Thus the baseline system represents a system that does not employ any resource allocation strategy. Partial-Intelligent system uses the first-order similarity or the initial ranking given by the search engine or other document gathering algorithms, to allocate resources. This system uses a static resource allocation that does not take into account the current state of the search space. The experimental runs are conducted with the control systems using the various queries and performance statistics collected. In order to accurately compare performance, these three systems use the same information resources and the same set of queries. By comparing the performance of the control systems with the I-FGM system, we demonstrate how the partial knowledge in an information nugget can be used to achieve efficient resource allocation to get better and faster results. We will also show that depending only on the first-order similarity of the search engines is insufficient.

4.3 Method and testbed

We evaluate I-FGM by comparing its performance with the control systems described above. The systems are run with four queries and results are collected and analyzed. We use recall to evaluate the performance of an information retrieval system. One of the goals of I-FGM is to retrieve the relevant set of documents quickly. We validate this result, by collecting the recall values of I-FGM at regular intervals throughout the duration of the experiment. Time taken to retrieve individual relevant documents is also noted, as this gives us insights on factors that influence the performance of I-FGM.

The queries that we have used for the simulation studies are based on the Asian Tsunami disaster of 2004 and are as follows:

1. Asian Tsunami disaster in South East Asia
2. Tsunami survivors in Indonesian islands
3. Damages caused by tsunami in Phuket beach, Thailand
4. Tsunami victims in Phi Phi Island

In creating the testbed for each query, we attempted to select the set of 10 documents with the highest similarity values (based on document graphs as described earlier). However, multiple documents can have the same similarity value depending on the measure used. For example, there might be 10 documents with the same similarity value, say for the 9th best position. Choosing all of them as relevant documents can make the relevant set large and not very useful when analyzing the performance difference between I-FGM and other control systems. On the other hand, choosing only some of these documents will skew the experimental results. For example, let doc_1, doc_2, and doc_3 be three documents of equal similarity. They have the 10th highest similarity in the testbed of a particular query. In order to maintain the size of the relevant set at 10, we would need to select one of the documents to be in the set. This can certainly skew the results. For example, let's select doc_3 to be the 10th document in the relevant set. While running the control systems with this query, the I-FGM system, hypothetically speaking, may find doc_1 at time $t = 1$, doc_2 at time $t = 2$, and lastly, doc_3 at time $t = 3$. Even though it has found equally relevant docs at time $t = 1$ and $t = 2$, its recall is incremented only at $t = 3$. Meanwhile some other control system such as Baseline system may select doc_3 at $t = 2$ and gets its recall value incremented before I-FGM. Due to this, the analysis of simulation results can lead to a false conclusion that Baseline system performs better. In other words, the performance of the systems cannot be compared with such a relevant set. As we will see below, we limit our relevant set to include all documents with the same similarity but try to avoid going much beyond 10.

For our experiments, we used a total of 19 machines. Each machine has a Pentium III 1 GHz processor and runs Red Hat 8.0 operating system. The machines are interconnected with a 100 Mbps Ethernet network. Of the 19 machines, 12 are used as *gIG-Builders,* 5 as *I-Foragers*, 1 as an *I-Matcher*, and 1 as *IGSoup* and *Blackboard.* Our testbed is created by sending the query for the current run to the *I-Forager* search engines. The top 50 results from each *I-Forager* are collected and combined to form the testbed. In order to identify the relevant document set, we process the documents using the document graph-based ranking method as described in Sect. 2.2. The control systems are then run using the documents in the testbed as the search space. In addition, to simulate a highly dynamic search space, we conduct a second experiment where the relevant documents are added to the search space at different instants in time. Thus, the time it takes for these documents to appear on the blackboard is an important indicator of system performance and is duly recorded.

4.4 Procedure for evaluating retrieval performance

We conducted three sets of experiments. The first set of experimental runs is static, i.e., all documents in the search space are available to the retrieval system from the beginning of the experiment. We use all 4 queries for each run. For each query, we run the I-FGM system and its two control systems with the query until all documents in the testbed for the particular query have been completely processed. For each run, we record the following data:

- Blackboard content: We record the contents of the blackboard at each interval time t. To determine t, multiple experimental runs were conducted to determine the optimum interval and 30 seconds was chosen because it is not so big that important trends are missed and not so small to put a strain on the database by constantly querying it.
- Data used for computing reliability for each *I-Forager:* The number of documents retrieved by a particular *I-Forager* that appears in the final blackboard content is used in computing a reliability rating for that *I-Forager* and its associated search engine. This rating is then used as part of the first-order similarity function.

We use the same procedure for Baseline, Partial-Intelligent and I-FGM systems. The neighborhood priority is not used to calculate the initial priority of the documents. Initial priority of a new document is equal to first order similarity. This first step helps to quickly validate I-FGM with a simple experimental setup.

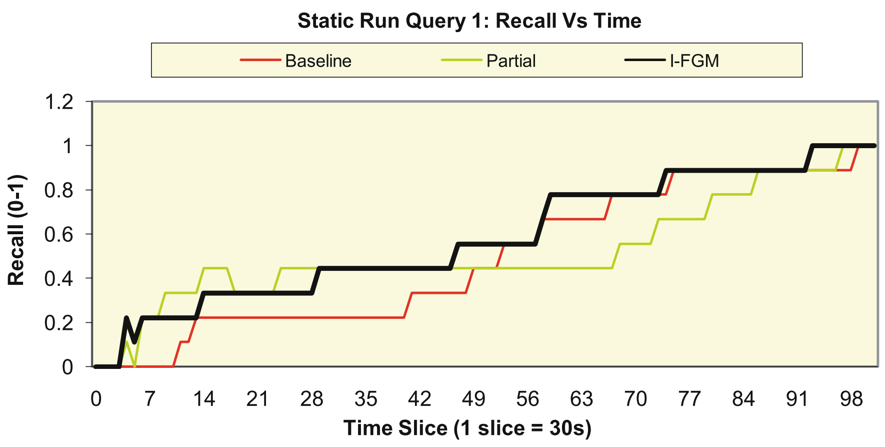

Fig. 3 Static run query 1: Retrieval vs time

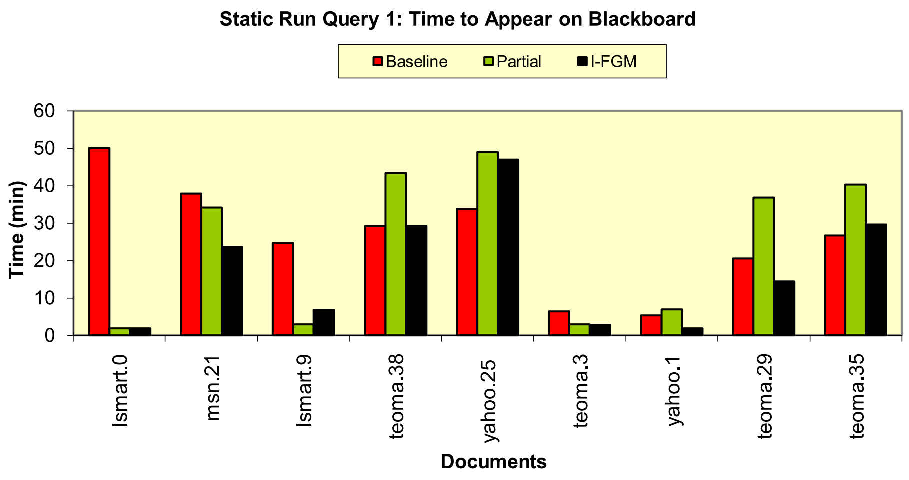

Fig. 4 Static run query 1: Time to appear on the blackboard

In the second set of experiments, we simulate dynamic databases by inserting relevant documents into the search space at different instants during the simulation. As in the first experimental set, the time taken for documents to appear on the blackboard, and the recall values are noted. The performance of the systems is compared and the results are analyzed to bring out their salient features.

I-FGM uses an equal number of highest priority selection and biased selection *gIG-Builders*. In order to justify this, we conduct the third set of experiments to compare the performance of I-FGM with two intermediate I-FGM systems, which have *gIG-Builders* with highest priority selection and biased selection respectively. They are called all-top I-FGM and all-biased I-FGM, respectively.

From the test results obtained from the experimental runs, we calculate the average time spent by a relevant document in each of the component of I-FGM. This gives us an insight into main bottlenecks of the system with respect to processing. This will help us to formulate a policy to allocate optimum number of *gIG-Builders*, *I-Matchers*, according to the processing load on the machines.

5 Experimental results and analyses

5.1 Experiment one—static runs

For brevity, and since this static experiment is identical to the one in [27], except for using different queries; we provide the results and analysis only for query 1. From the recall vs. time graph in Fig. 3, I-FGM has better recall in the 0.4–1 range. It reaches these recall levels faster than the other systems. Another point we observed is the surprisingly good performance of the Baseline system when compared to the Partial-Intelligent system. In similar experiments in [27], we see that on average, Baseline performed worse than the Partial-Intelligent system. This can be attributed to the fact that the performance of the Baseline system increases when the ratio of the number of document in the search space to the number of *gIG-Builders* decreases. In [27], we used only 5 *gIG-Builders* compared to the 12 *gIG-Builders* being used in this experiment, while the total number of documents in the search space remained about the same. Figure 4 is the histogram comparing the time it takes for the relevant documents to appear in the Blackboard for each of the three

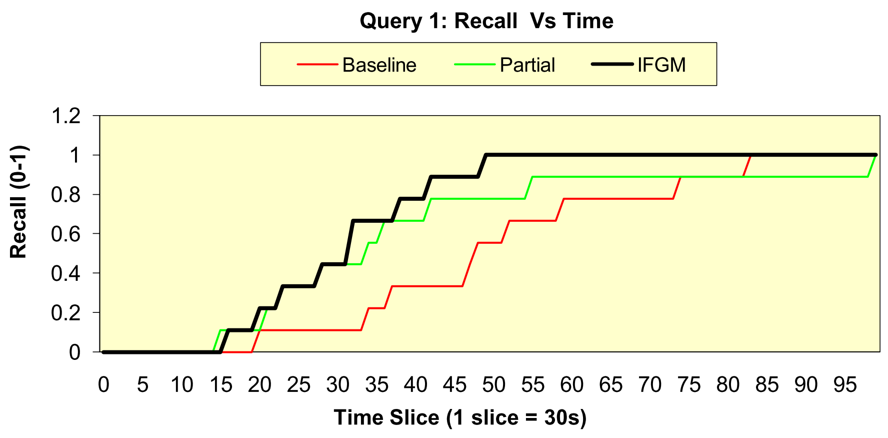

Fig. 5 Dynamic run query 1: Recall vs time

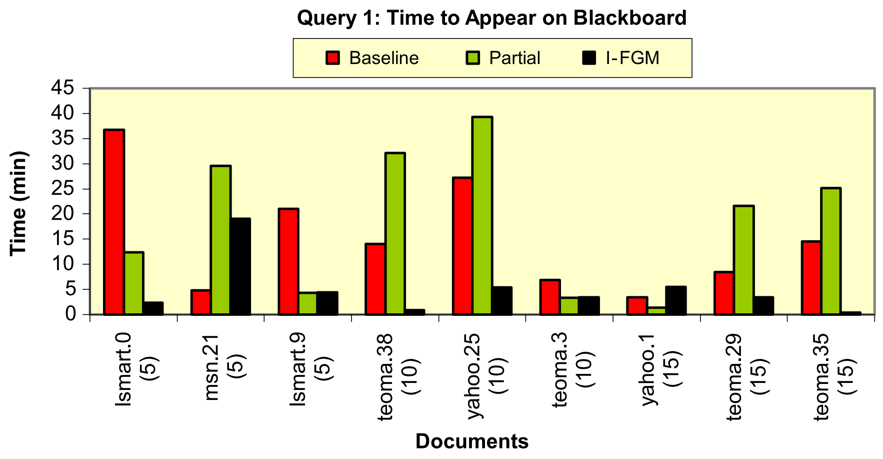

Fig. 6 Dynamic run query 1: Time to appear on the blackboard

systems. The relevant documents are listed on the x-axis in the form $x.y$ where x is the search engine that downloaded the document and y is the rank returned by the search engine. I-FGM performed best by retrieving 6 documents out of a total 9, faster than the control systems.

5.2 Experiment two—dynamic runs

In order to simulate a dynamic search space, we conducted a second set of experiments - the dynamic runs. The relevant documents are held back from the *I-Foragers* and are inserted into the search space at different time instants during the simulation. The instants at which they are inserted are identical for all the three systems but were chosen randomly. For all the queries, I-FGM typically has better recall time than other control systems. Results for queries 1, 2, 3, and 4 are provided in Figs. 5–12. Figures 5, 7, 9, and 11 show the relationship between recall and time for these 4 queries. The y-axis is the recall value which is between 0 and 1 while the x-axis represents time-slices starting from 0. For all the queries, I-FGM has better recall than the other control systems for most of the simulation time. For queries 1 and 3, I-FGM has highest recall throughout the experiment.

For query 2, Partial-Intelligent has better recall than I-FGM in 0.1–0.2 range. This is because the relevant set consists of looksmart.1 and looksmart.12, which are already highly ranked by Looksmart (#1 and #12, respectively) and are inserted into the search space early on in the simulation. Thus, Partial-Intelligent is able to retrieve them both quickly and reach a 0.2 recall faster than I-FGM. For query 4, in the recall region 0–0.4, the Partial-Intelligent system is better than I-FGM. The reason is that a number of relevant documents are short documents and are highly ranked by the search engines. Since the Partial-Intelligent system depends only on the measure returned by the *I-Foragers*, these short relevant documents are quickly displayed to the analyst. Hence, we see the spurt in recall during the first part of the simulation. The performance of I-FGM is marginally inferior for these documents. The other documents in the relevant set are ranked low by the *I-Foragers*. The Partial-Intelligent system takes a long time to process them. Since I-FGM uses refined priority measures, it picks up these documents much faster and consequently has better performance. Though Partial-Intelligent may win over I-FGM in queries 2 and 4, they are for only short durations. On the other hand, in all the queries, I-FGM reaches the full recall

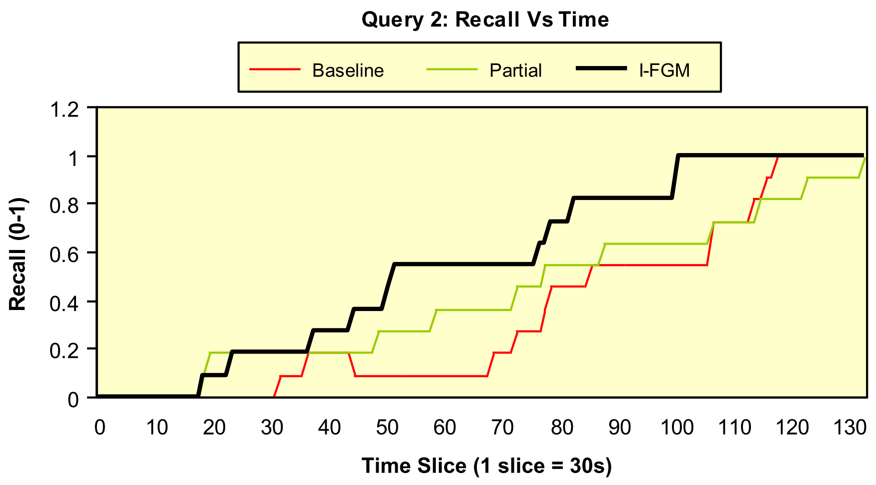

Fig. 7 Dynamic run query 2: Recall vs time

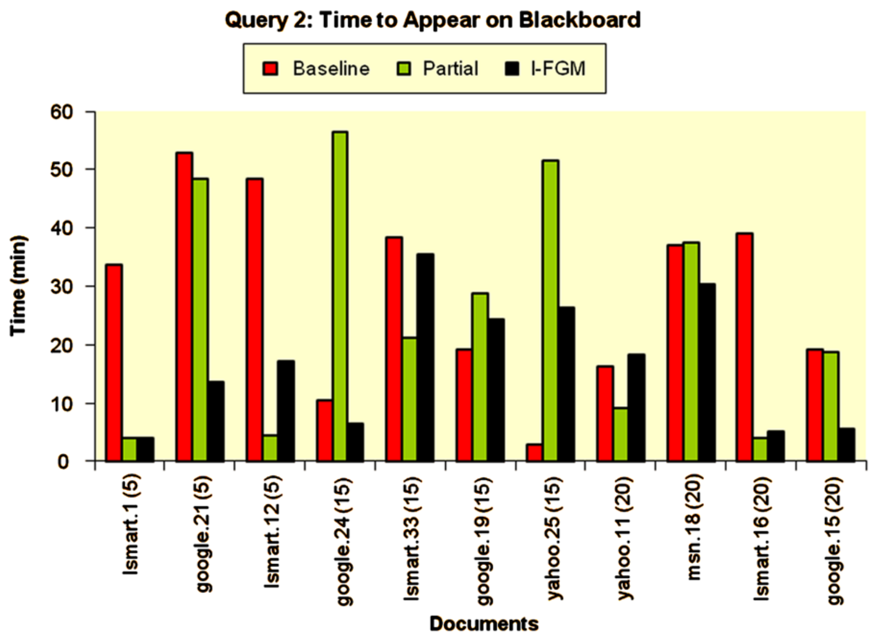

Fig. 8 Dynamic run query 2: Time to appear on the blackboard

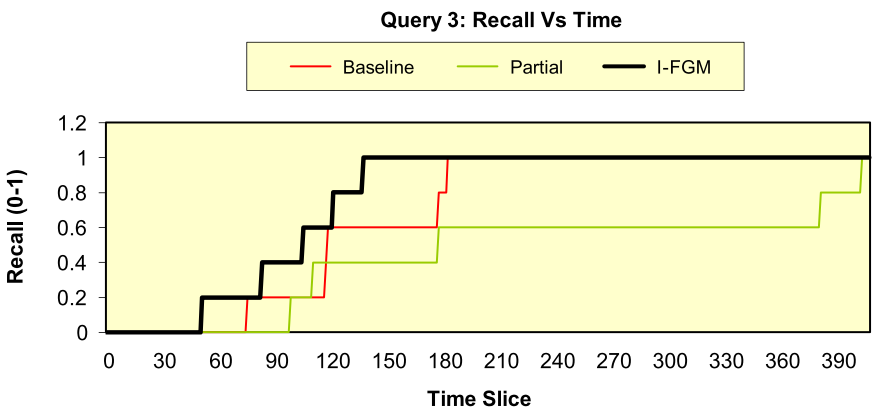

Fig. 9 Dynamic run query 3: Recall vs time

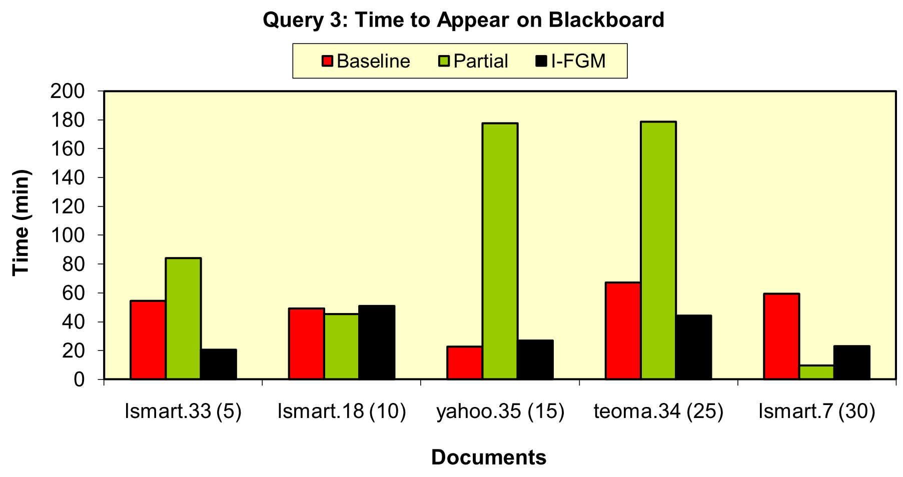

Fig. 10 Dynamic run query 3: Time to appear on the blackboard

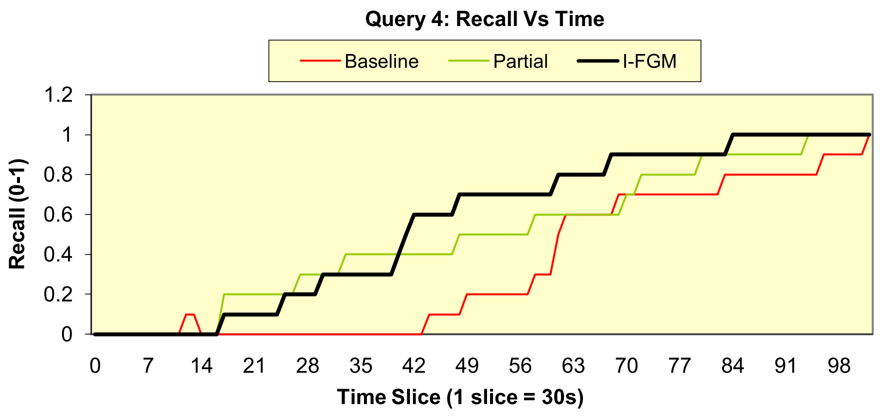

Fig. 11 Dynamic run query 4: Recall vs time

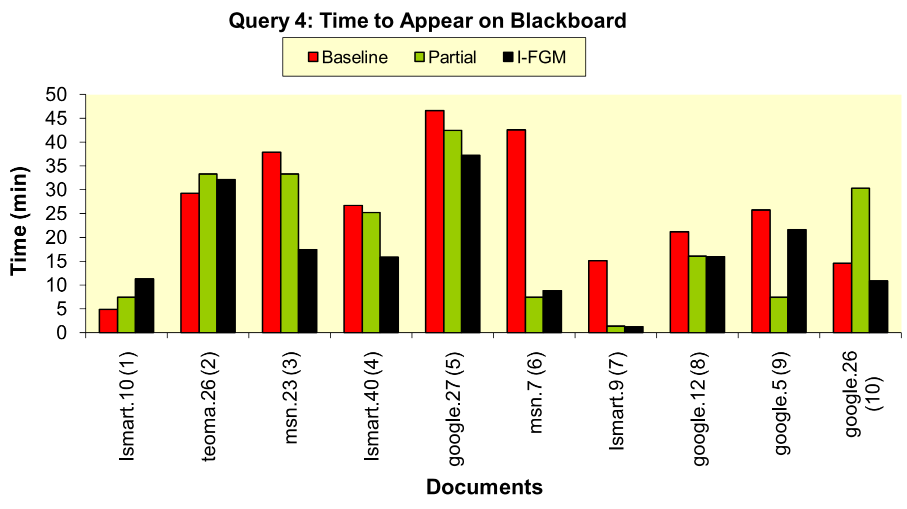

Fig. 12 Dynamic run query 4: Time to appear on the blackboard

the fastest. In fact for queries 1, 2, and 3, I-FGM reaches full recall 15 or more minutes faster than Partial-Intelligent or Baseline systems.

In all the queries, I-FGM discovers the maximum number of relevant documents faster than the other control systems. The average time for the documents to appear on the Blackboard is also the lowest in I-FGM. Figures 6, 8, 10, and 12 are the histograms to depict the time it takes for relevant documents to appear in the Blackboard for each system during the dynamic runs. The y-axis is the time in minutes and the x-axis has the documents that are inserted dynamically. The documents are named using the form $x.y(z)$,

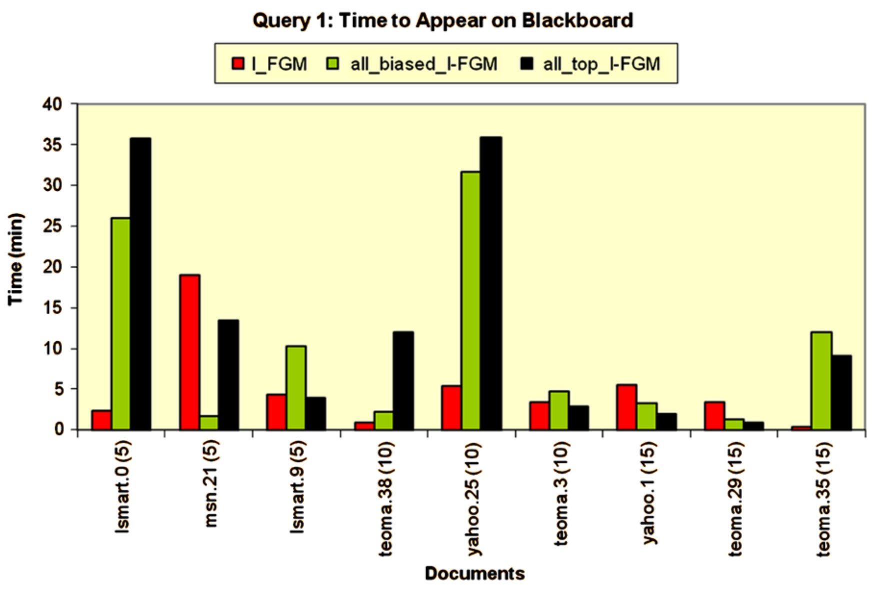

Fig. 13 Comparison of gIG-builder document selection criterion—time to appear on the blackboard

where x is the search engine that found the document, y is the rank returned by the search engine and z is the time (in minutes) when the document was added to the search space. For query 1, out of 9 documents, I-FGM retrieves 5 documents faster than other systems. It ties with Partial-Intelligent for 2 other documents. Similarly, the dominance of I-FGM is clear for queries 2 and 4 with retrievals of 6 out of 11, and 6 out of 10, respectively. For query 3, I-FGM ties with Partial-Intelligent by getting 2 out of 5 documents. In order to understand why this happens, we have to analyze the factors that affect the performance of the Partial-Intelligent and I-FGM system. For the Partial-Intelligent system, the factors are the rank (returned by the search engine) of the document and the time that it was inserted. If a is the number of *gIG-Builders* used, then the Partial-Intelligent system processes the documents in sets of size a. For example, a document ranked $(a + 3)$ cannot be processed until at least 3 documents in the top a documents have been completely processed. When a relevant document ranked low by the search engine is inserted at the beginning of the simulation, the Partial-Intelligent system will not discover the document until the documents ranked above it have been processed. By the same reasoning, it is easy to see that relevant documents ranked high by the search engine are quickly discovered by the Partial-Intelligent system. For the I-FGM system, the factors are the rank (according to priority) of the document and layout of the document. By layout of the document, we refer to the distribution of the relevant parts within the document. Since the I-FGM reduces the priority of the document if it does not find anything relevant during one cycle of parsing, documents that have its relevant portion in the beginning or uniformly distributed have a better chance of being discovered quickly. Since the priority function that we use is not rigid, future work would be to fine tune it for better performance.

5.3 Experiment three—selection functions

As we described earlier, the *gIG-Builders* use two types of document selection strategies: (i) highest priority selection and (ii) biased selection. It is our hypothesis that each of these strategies can work efficiently only in some scenarios and that using a combination of these two types of *gIG-Builders* is the best approach. This issue is critical as the processing in the *gIG-Builders* is the main computational bottleneck. Any performance advantage that we gain by using the right kind of *gIG-Builders* will have significant impacts on the overall performance of I-FGM. In the third set of experimental runs, we compare the performance of I-FGM systems with the other intermediate I-FGM systems: all-top I-FGM and all-biased I-FGM. From the recall graph in Fig. 14, we see that the I-FGM system has better recall over time. The I-FGM system reaches full recall 7 minutes before the other two intermediate systems, which is a significant performance difference. Also, the average time for relevant documents to appear on the Blackboard is lowest in the I-FGM system compared to all-top or all-biased intermediate systems. From Fig. 13, we compare the time taken for relevant documents to be displayed on the Blackboard for the three systems. We see that I-FGM performs better than the other two intermediate I-FGM systems. Out of 9 documents, the I-FGM system is a clear winner in 4 and ties with all-top system for 2 other documents. The other two systems are clear winners for only one document each. Each of them wins for 4 documents and ties for one when we compare the results for just all-top and all-biased I-FGM systems. This provides the intuition that I-FGM should employ a mix with

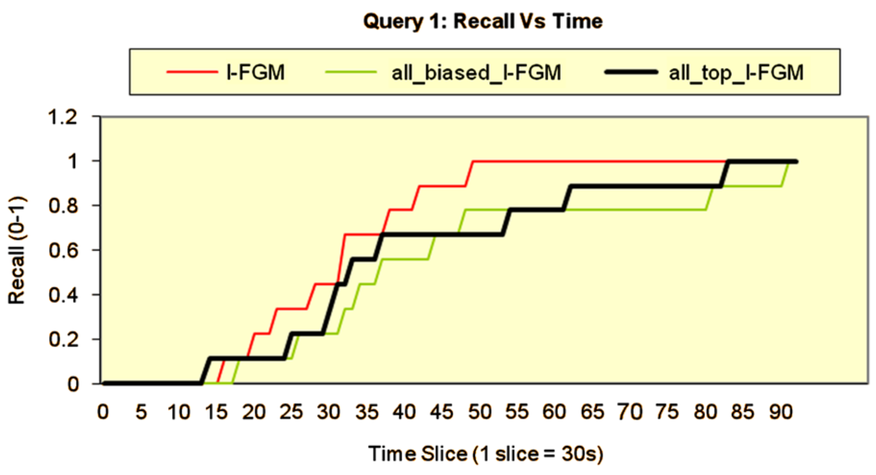

Fig. 14 Comparison of gIG-builder document selection criterion—recall vs time

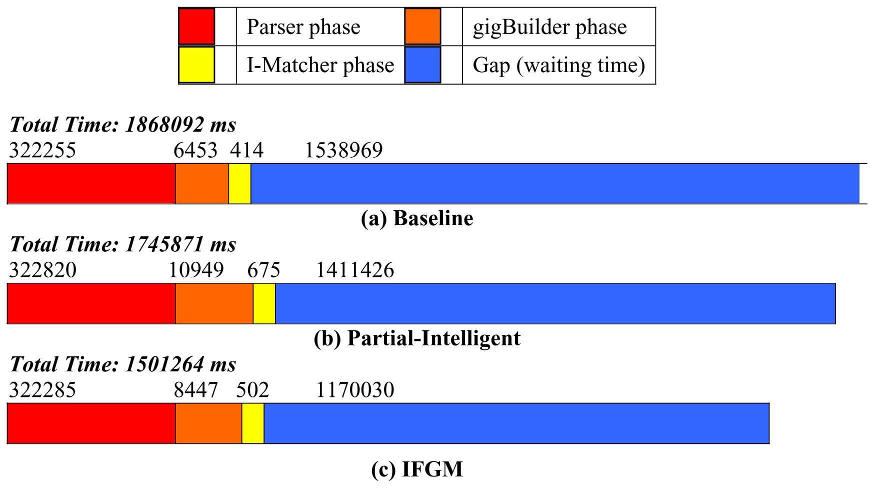

Fig. 15 Process pipeline for control systems

half of the *gIG-Builders* using the highest priority selection method and the other half using the biased selection.

5.4 Computational costs

We tabulated the time spent on the documents in the different components and prepared a process pipeline representing the average processing time for the top relevant documents for each query. Given the limitation of space, we present only the results for query 1 which is representative of all the queries in Fig. 15. From this figure, we see that the main processing is done in the link parser phase which is a part of the *gIG-Builders*. This provides guidance on initially allocating the computational resources among *gIG-Builders*, *I-Foragers* and *I-Matchers* (represented by their numbers). This figure also shows that the waiting time is the least for I-FGM, which is indicative of the effectiveness of our resource allocation strategy. This will also be useful for future work on dynamically changing agents from *gIG-Builders* to *I-Matchers* and so on to further improve performance.

5.5 Scaling up I-FGM

From the experimental results for static and dynamic simulations, we see that I-FGM has achieved better performance than the Baseline and Partially-intelligent systems. But one drawback shown in the results is that the average time to "discover" a relevant document is longer with respect to the analysts' requirement. The analysts deal with critical, real time events and the average discovery times that we obtained in our experimental results, for example, 33 minutes for query 3 and 17 minutes for query 4, are not acceptable. We have already mentioned that the I-FGM architecture is easily scalable and more nodes can be included to provide better performance. But this aspect of I-FGM should not be confused with brute force methodology wherein the naïve solution is to just increase the computational resources. We show in a succeeding section on system analysis, that I-FGM has a better chance of discovering relevant documents earlier because of its smart resource allocation strategy. By increasing the number of nodes, we reduce the contention for resources and this combined with our resource alloca-

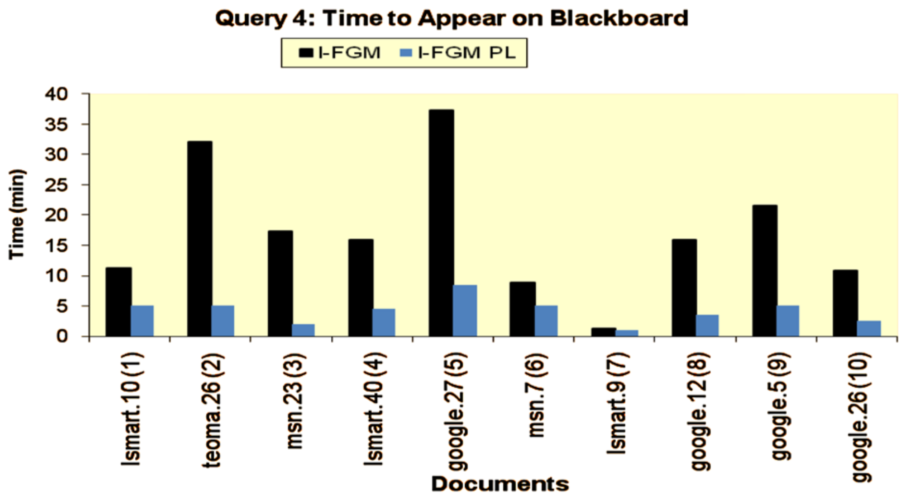

Fig. 16 Discovery time for relevant document between I-FGM and I-FGM PL

tion should improve the odds of retrieving the relevant documents in I-FGM.

In order to demonstrate this, we implemented the I-FGM system using 42 nodes, which included 36 *gIG-Builders* and 1 *I-Matcher*. The nodes are high end Intel Pentium-D 2.8 GHz 64-Bit dual processor servers connected by 1 GBPS Ethernet links. We term this production level I-FGM prototype as I-FGM PL. We repeated the previously dynamic simulations on this system and analyzed the results. We saw that average discovery time were 4 times faster than the previous simulations. For lack of space, we do not provide the complete simulation results. In Fig. 16, we compare I-FGM and I-FGM PL prototypes using the discovery time for relevant documents in query 4. We see I-FGM PL performs better than I-FGM with its average discovery time 4 times smaller than I-FGM (17 min for I-FGM and 4 min for I-FGM PL). It may be argued that even 4 min is unacceptable. But this delay is mainly caused by the text parsing in gIG Builders and as such is unavoidable.

5.6 Summary

In short, after we analyze the simulation results, for both recall and discovery time, we see that I-FGM is a clear winner. All these results show that I-FGM is a viable way to get relevant results quickly even with minimum computation resources in a large and dynamic search space. Also, we can readily infer that the elapsed times for I-FGM can be significantly decreased as the number of *gIG-Builders* and other processes increases.

6 System performance analyses

In previous sections, we describe the initial design, implementation as well as demonstrate the effectiveness of our I-FGM system for both static and dynamic search spaces. In order to make our results more convincing, we provide rigorous theoretical analyses of system performance, validate the effectiveness and efficiency of our system, and show under what conditions and to what extent the I-FGM system will outperform current technologies and its control systems. In this section, we focus on static and homogeneous search spaces, and provide our theoretical analyses of the performance of information retrieval systems including the Baseline (B), the Partial-Intelligent (PI), and the I-FGM system as well as the systems without partial processing mechanisms.

With the continuing proliferation of digitalized information, a great amount of information retrieval technologies and systems have been developed based on various models, such as vector space model [40], probabilistic model [40], and language model [35]. Although models are different, most of these techniques, if not all, share a common feature that the similarity result of a document can be obtained only *after* the full processing of this document. We call the systems based on this type of techniques as Full-Processing (FP) systems. For FP systems, document processing cannot be interrupted. The similarity value of a document is either zero or the final value. Although FP systems are the most popular systems employed in current IR tasks, they waste time on processing documents that are ultimately irrelevant to users.

6.1 Performance metric and frequently used parameters

We assume that the retrieval technique which uses DG representation employed in the I-FGM can accurately measure the true similarity between any document contained in the search space and the user's query. In order to identify the documents which the user is interested in, we assume that the documents ranked in the top s based on their similarity values are treated as relevant to the user's query. For Partial Processing (PP) systems including the Baseline (B) system, the Partially Intelligent (PI) system, and the I-FGM system, the similarity value of a document is partially and incrementally built. Thus, a relevant document may be treated as relevant (or not) alternatively during system processing. When more and more parts of the document are processed, the document's rank according to the similarity value will become more and more stable. A relevant document is called *stably ranked as relevant* when this document is contained in the *relevant set*, a set of documents which are treated as relevant to the user's query, and never falls out of it. Thus, we decide to use the probability that at time t a relevant document d_i is *stably ranked as relevant* to indicate the system performance. This is depicted as:

$$\mathrm{perf}_X = P_X(t|d_i) \tag{6}$$

where $P_X(t|d_i)$ represents the average probability for a relevant document d_i to be *stably ranked as relevant* at time t.

In our theoretical analyses of system performance, we use the following assumptions in order to help bring to light significant performance insights of these systems:

(i) All documents in a search space have the same size. They are evenly partitioned into K slices in both the B system and the PI system.
(ii) Slices of a document are independent of each other.
(iii) Ignoring the partial processing overhead such as the cost for communication and synchronization, the cost for loading documents or document graphs into *gIG-Builders*.

Based on the first assumption, all documents are the same except that they have different similarity values. From the second assumption, we can see that the sequence for processing document slices does not impact the B or PI system performance. Obviously, the processing sequence of document slices does not affect the document's final similarity result. For the B system, at any given time, each un-processed document slice has the same possibility to be processed. Therefore, the B system can process document slices in arbitrary order without affecting its performance. In the preliminary analyses of system performance, we want to focus our sights on studying system computation load. Thus, based on the third assumption, performance analysis results of our distributed system can be achieved by scaling up the analysis results of systems with only one processor. Thus, our theoretical analyses of system performance can be initially focused on systems with one processor.

In order to facilitate easy understanding of the theoretical analyses contained in this section, we list definitions/notations of frequently used parameters in Table 1.

Table 1 System parameters

Parameter	Definition
d_i	Document i which is relevant to the user's query
n	The total number of documents contained in the database (*IGSoup*)
m	The number of documents a system has processed
s	The number of relevant documents contained in the database (*IGSoup*)
K	The number of slices that a document is partitioned into in Partial Processing systems (including Baseline, Partial-Intelligent, and I-FGM systems)
T_d	The time for a system to fully process a document
T_{slice}	The time for a system to process one slice of a document. $T_d = KT_{\mathrm{slice}}$
a_i	The minimum number of processed slices for a relevant document d_i to be stably ranked as relevant, $a_i \leq K$
$P_X(t\|d_i)$	The probability for a relevant document d_i to be stably ranked as relevant at time t by system X
$p_X(d_i)$	The probability for a relevant document to be processed at any time in the system X

6.2 Full-processing (FP) system

All documents contained in the search space are treated as the same in Full-Processing (FP) systems. These systems randomly select a document and wholly process it to obtain its similarity value. After processing, a document's similarity value will not change. Thus, in static search spaces, as soon as a FP system finishes processing a relevant document, this document will always be ranked and remain as relevant. It is easy to see that system retrieval results (i.e. ranks of documents) may change only after the system finishes processing a whole document. We therefore analyze the FP system performance at the time when a document is fully processed. At time $t = mT_d$, exactly m documents have been fully processed. The number of total candidates of these m documents is

$$\binom{n}{m}$$

The number of candidates which contains a relevant document d_i is:

$$\binom{n-1}{m-1}$$

Since all of the documents are uniformly considered by a FP system, the chance for each candidate being selected is the same. Thus, for a FP system, the probability P_{FP} that a relevant document d_i is contained in these m handled documents can be defined as:

$$P_{\mathrm{FP}}(mT_d|d_i) = \binom{n-1}{m-1} \bigg/ \binom{n}{m} = m/n \tag{7}$$

6.3 Baseline (B) system

In this section, we analyze the characteristics of a B system. In order to demonstrate the advantages of the B system, we compare the performance of B system with FP system and determine under what conditions and to what extent B system will outperform FP system.

6.3.1 B system performance

Based on the design of the B system, we can see that as soon as the system finishes its current process, it randomly picks up one un-processed document slice to handle. Since this system treats each document slice uniformly, the probability p_B that after each partial process the B system selects a slice of document d_i to process can be formed as below:

$$p_B(d_i) = 1/n \tag{8}$$

To compare the B system and FP systems, we check the performance of the B system at time $t = mT_d$. At this time, we know there are mK document slices processed because $T_d = KT_{\text{slice}}$. In the search space (*IGSoup*), there are n documents which indicate that the total number of document slices is Kn. Since document slices can be processed in arbitrary order, the total number of possible candidates of these mK document slices at time t is:

$$\binom{nK}{mK}$$

In order for a relevant document d_i to be *stably ranked as relevant*, we assume that the system should process at least a_i slices of this document. The total number of possible candidates that the B system has processed at least a_i slices of d_i at time $t = mT_d$ can be approximated as:

$$\binom{K}{a_i} \cdot \binom{nK - a_i}{mK - a_i}$$

Since each candidate has the equal chance to happen, the probability that in the B system a relevant document d_i is *stably ranked as relevant* at time $t = mT_d$ can be formed as:

$$P_B(mT_d|d_i) = P_B(mKT_s|d_i) = \frac{\binom{K}{a_i} \cdot \binom{nK-a_i}{mK-a_i}}{\binom{nK}{mK}} \tag{9}$$

From (9), we can see that a_i is an important factor which significantly affects the system performance on document d_i. It is not hard to imagine that in many cases the document does not need to be fully processed before it is *stably ranked as relevant*. Intuitively, a large a_i indicates that the system should put more effort to distinguish this relevant document from not-relevant ones, and vice versa. As K changes, the value of a_i itself does not determine the system performance. Instead, the relationship between K and a_i does. For a fixed a_i, larger K indicates less effort to find relevant documents. To maintain a specific performance, K and a_i should be in an inverse relationship. We define r_i to indicate this relationship:

$$r_i = a_i/K \tag{10}$$

6.3.2 Comparison between B system and FP system

One common question is that when and under what conditions Partial-Processing systems perform better than Full-Processing systems. However, it is hard to guarantee any advantage of Partial-Processing systems since how a Partial-Processing system performs is really dependent on the nature of the data contained in the search space. As we mentioned in the previous sub-section, B system's performance is mainly affected by r_i which is determined by the nature of document d_i. In order to provide further analyses of (9), and a deeper understanding of how r_i affects system performance, we compare the FP system with the B system on their performance metric $P_X(t|d_i)$ under various system configurations (different values of K and n).

The comparison can be accomplished by analyzing the following inequality:

$$P_B(t|d_i) > P_{\text{PF}}(t|d_i) \tag{11}$$

If we can solve this inequality, we can know that under what condition (values of r_i) and to what extent the B system will outperform a FP system. The extent of B system's performance advantage can be measured as the percentage of the time range during which the inequality (11) is satisfied. We define this parameter as the percentage of time intervals in which the B system outperforms the FP system and use Δ to represent it. However, it is difficult to mathematically solve Δ from (11). The value of Δ may be affected by different parameters, such as r_i, K, and n. Thus, we use numerical methods to try to find when and how Δ changes with the system configuration (n and K) and r_i. The sets of system configurations used in our numerical computation are shown in Table 2. We constrained the minimum number of documents contained in the search space to 100 since our system is designed for dealing with large search spaces.

In our computation, we draw two curves described by (7) and (9) respectively, and find the time intervals which satisfies (11). Based on the size of these time intervals, we calculate the value of Δ. The results are shown in Fig. 17.

From the computational results we can discover the following facts:

1. The value of n does not affect Δ.
2. The value of Δ has an inverse relationship with r_i.

Table 2 System configuration parameters

n	K	r_i
100	5, 10, 20, 40, 60, 80, 100	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1
1000	5, 10, 20, 40, 60, 80, 100	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1
10000	5, 10, 20, 40, 60, 80, 100	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1
100000	5, 10, 20, 40, 60, 80, 100	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1

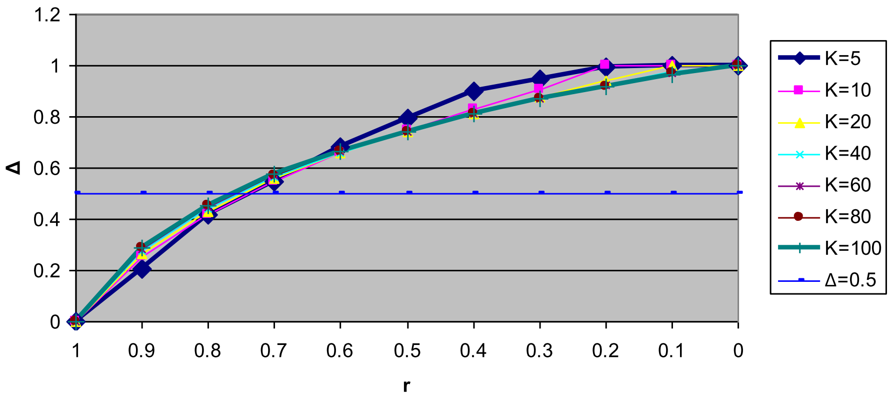

Fig. 17 Percentage of time intervals in which the B system outperforms the FP system with respect to r and K for all values of n

3. Different values of K generate different curves of Δ versus r_i. However, all these curves are contained in a narrow band.
4. The B system will always outperforms FP systems ($\Delta > 0.5$) when r_i is less than 0.7.
5. When r_i is larger than 0.3, the curves of $K = 10$ and $K = 20$ are quite close to each other, even superposed.

Based on our experience from our experiments on Partial-Processing systems, a reasonable value of K for real search spaces is between 10 and 20. Based on fact 5, we decide to use the curve of $K = 10$ as an approximation of the B system performance. From Fig. 17, we also can see that the curve of $K = 10$ is approximately linear during the range from $a_i = 0.9$ to $a_i = 0.3$.

The parameter a_i represents the property of a specific document d_i. It is obvious that documents are quite different from each other so that the B system performance on each relevant document may be quite dissimilar. Therefore, we focus on average performance of systems. Based o n the linear relationship between system performance and a_i, we employ the average value of a_i which is denoted as a_B:

$$a_B = \sum_{i=1}^{s} a_i / s \tag{12}$$

to indicate the average performance of the B system which is defined as r_B in (13):

$$r_B = a_B / K \tag{13}$$

The parameter, r_B, is determined by the query and the nature of real search spaces. Its value will be measured in our system performance experiments.

6.4 Partial-intelligent (PI) system

In the PI system, we rank documents by *a priori* ordering of documents based on FOS (*First-Order Similarity*). The PI system processes documents in the decrease order of FOS. Although it partitions documents into slices and processes documents in the unit of document slice which makes us to label it as a Partial-Processing system, in fact, it acts more like Full-Processing system. When it begins to process a document, it will not process other documents until it finishes all slices of the current document. The performance of the PI system is totally determined by how well the FOS indicates the document's relevance to the user's query. This property of FOS can be represented by a function $\Gamma(t|d_i)$ which represent the probability for a relevant document d_i to be *stably ranked as relevant* at the time t. Thus, the system performance P_{PI} is:

$$P_{\mathrm{PI}}(t|d_i) = \Gamma(t|d_i) \tag{14}$$

In the preliminary analyses of the PI system performance, the function $\Gamma(t|d_i)$ is defined in the following way. Assume that all relevant documents are uniformly treated in the PI system. Also, assume that the top m documents ranked by FOS contain a subset $R(m)$ of relevant documents. We denote the number of documents contained in $R(m)$ as $|R(m)|$. The probability of a relevant document d_i to be contained in $R(m)$ can be formulated as $|R(m)|/s$. Thus, the PI system performance metric, the probability P_{PI} that a relevant document d_i is *stably ranked as relevant* at the time $t = mT_d$ can be formed as:

$$P_{\mathrm{PI}}(mT_d|d_i) = \Gamma(mT_d|d_i) = |R(m)|/s \tag{15}$$

6.5 I-FGM system

In the I-FGM system, we assign initial values (based on FOS) for document priorities and refine them based on the analysis of partial results obtained during system processing. The refinement of priorities imbues the I-FGM system with the ability to intelligently allocate its computational resources to achieve better efficiency. Comparing with the B system, the advantage of I-FGM can be represented as the average number of slices required for stably ranking a relevant document as relevant is less than a_B. Comparing with the PI system, the advantage of I-FGM can be shown as the fact that when the FOS is not well defined, the refined document priorities can identify the promising documents and intelligently reallocate the resources accordingly.

I-FGM system is quite complicated due to the nature of documents and the complexity of priority function. Therefore, we simplify the I-FGM system in the following way:

1. In our original design, for each partial processing step, the amount of data to be processed changes with the document priority. In our simplified model, we fix this amount as the value used in the B and the PI systems. We believe that, with a good priority function, our original system would need to process lesser number of document slices, than the simplified model, to stably rank relevant documents.
2. In our simplified model, we use the obtained document similarity as the document priority, instead of considering a lot of other factors, such as the history of the document's similarity value, the increment of the document similarity value, etc. Therefore, $\mathrm{pri}_{di}(t) = \mathrm{sim}_{di}(t)$, where pri represents document priority value and sim represents document similarity value at time t.
3. In our original design, the initial value of document priority is calculated from the FOS. In our simplified model, we set the same initial value of all document priorities.
4. In this paper, the I-FGM system uses a hybrid mechanism to select a document to be processed at each partial step. Processors are partitioned into two sets. One set of processors select the document strictly according to the rank on document priorities. The other set of processors select the document based on a probability which is proportional to the value of documents priorities. This probability can be formed as

$$p_{\text{I-FGM}}(t|d_i) = \mathrm{pri}_{d_i}(t) \Big/ \sum_{j=1}^{n} \mathrm{pri}_{d_j}(t)$$
$$= \mathrm{sim}_{d_i}(t) \Big/ \sum_{j=1}^{n} \mathrm{sim}_{d_j}(t) \tag{16}$$

where $p_{\text{I-FGM}}(t|d_i)$ represents the probability that the I-FGM system selects document d_i to process at time t. In our simplified model, all processors employ the second mechanism. We believe that with a good priority function, our original design would perform better than the simplified model.

Under these simplifications, the I-FGM system performance is determined by the nature of search spaces which mainly depends on two kinds of distributions. One is the distribution of *nuggets* (elements which are relevant to the user's query) in a document. The other one is the distribution of documents according to their similarity values.

The *nuggets* distribution within a document can be arbitrary. Studying this distribution is a research topic which requires a great amount of statistical work. However, if we can assume this distribution is uniform, then based on (16), we can get that on average, the probability that the I-FGM system selects a relevant document d_i to process at a partial step can be formulated as

$$p_{\text{I-FGM}}(d_i) \geq \mathrm{sim}_{d_i} \Big/ \sum_{j=1}^{n} \mathrm{sim}_{d_j} \tag{17}$$

This is because, at the beginning, all documents have the same possibility to be processed. After I-FGM partially processes a document, the document's priority will be proportional to the similarity value contained in the processed document slices. Thus, in the initial stage, the probability will be close to the value on the right hand side of inequality (17). As the I-FGM system processes more document slices, there seems to be a positive feedback for relevant documents. The high similarity value introduces more chances for a relevant document to be processed which leads to higher similarity value. Thus, we say that the probability to select a relevant document to be processed is bounded by the right hand side part in (17). In our preliminary analysis of the I-FGM system performance, we approximate the probability $p_{\text{I-FGM}}$ as:

$$p_{\text{I-FGM}}(d_i) = \mathrm{sim}_{d_i} \Big/ \sum_{j=1}^{n} \mathrm{sim}_{d_j} \tag{18}$$

The distribution of documents according to their similarity values can also be arbitrary. To make our analysis easier, we assume this distribution to be uniform too. If we take top λ percent documents as relevant, the average similarity value for relevant documents will be $(1 + 1 - \lambda)/2$. The average probability for I-FGM system to choose a relevant document d_i at a partial step is:

$$p_{\text{I-FGM}}(d_i) = \frac{2-\lambda}{2} \cdot \frac{1}{\sum_{j=1}^{n} \text{sim}_{d_j}} \tag{19}$$

From the distribution of documents similarity values, we know that:

$$\sum_{j=1}^{n} \text{sim}_{d_j} = (1+0) \cdot n/2 = 0.5n \tag{20}$$

Therefore, we can get that

$$\begin{aligned} p_{\text{I-FGM}}(d_i) &= \frac{(2-\lambda)}{2} \cdot \frac{1}{\sum_{j=1}^{n} \text{sim}_{d_j}} \\ &= (2-\lambda)\frac{1}{n} = (2-\lambda) p_B(d_i) \end{aligned} \tag{21}$$

We use c to present the performance comparison between the B system and the I-FGM, and define it as:

$$c = \frac{p_{\text{I-FGM}}(d_i)}{p_B(d_i)} = (2-\lambda) \tag{22}$$

Currently, in our I-FGM system, we set λ as 10%. Thus, $c = 1.9$ and $p_{\text{I-FGM}}$ is 1.9 times of p_B. This means that the I-FGM system would retrieve the relevant documents at least 1.9 times faster than the Baseline system. It also indicates that at a given time, the average number of processed slices of relevant documents in the I-FGM system is about 1.9 times of the corresponding number in the B system. Therefore, we can obtain that

$$a_{\text{I-FGM}} = a_B/c = a_B/1.9 \tag{23}$$

6.6 Experimental results and analysis

In this section, we present the measurement of system performance on selected queries "Asian Tsunami disaster in South East Asia" (Query 1), "Tsunami survivors in Indonesian islands" (Query 2), "Damages caused by tsunami in Phuket beach, Thailand" (Query 3), and "Tsunami victims in Phi Phi Island" (Query 4). As we mentioned before, the performance of the PI system completely depends on the design of FOS. Therefore, the experimental validation of the PI system performance is not included in this paper. We will provide the PI system performance results and analysis in our future work on studying the FOS design.

Table 3 Measurement of a_B and r_B

Query	K	a_B	r_B
Query1	6.450	4.091	0.634
Query2	11.537	4.545	0.394
Query3	20.754	11.2	0.540
Query4	5.870	4.3	0.733
Average	*11.15*	*6.034*	*0.575*

6.6.1 Measurement of a_B (comparison between the B system and FP system)

As we mentioned, the performance comparison between the B and FP system can be indicated by the value of r_B, which can be measured by a_B and K. In the following table, we present the measurements of the B system on 4 queries. From Table 3 we get that the average r_B is 0.575. Based on Fig. 17, we can see that the B system outperforms FP systems and Δ is about 70%.

6.6.2 I-FGM system performance measurement (comparison between I-FGM system and B system)

As we discussed in the previous section, the value of $a_{\text{I-FGM}}$ can be estimated from the value of a_B and the comparison between the I-FGM system and the B system. The comparison can be achieved based on comparing $p_{\text{I-FGM}}$ and p_B.

When a relevant document d_i is initially *stably ranked as relevant*, the number of slices processed are represented as z_i. The average probability for a system to process this document d_i can be formed as:

$$p(d_i) = z_i/m_i = z_i/t_i/T_s = z_i T_s/t_i \tag{24}$$

where m_i is the number of slices processed by the system at the time t_i. The average probability to choose a relevant document is

$$\bar{p}(d) = \sum_{i=1}^{s} p_X(d_i)/s = \sum_{i=1}^{s} \frac{1}{s} \cdot \frac{z_i T_s}{t_i} \tag{25}$$

Therefore, the comparison c can be estimated as

$$\begin{aligned} c &= \frac{\bar{p}_{\text{I-FGM}}(d)}{\bar{p}_B(d)} \\ &= \frac{\sum_{i=1}^{s} z_{\text{I-FGM}i} \cdot T_s/(t_{\text{I-FGM}i} \cdot s)}{\sum_{i=1}^{s} z_{Bi} \cdot T_s/(t_{Bi} \cdot s)} \\ &= \frac{\sum_{i=1}^{s} z_{\text{I-FGM}i}/t_{\text{I-FGM}i}}{\sum_{i=1}^{s} z_{Bi}/t_{Bi}} \end{aligned} \tag{26}$$

In addition to recall and the time to appear on blackboard, we need more detailed information in order to evaluate I-FGM performance. Unfortunately, we did not capture

Table 4 Measurement of z and t for query 1 and 2

Query 1					Query 2				
Relevant document	Baseline		I-FGM		Relevant document	Baseline		I-FGM	
	Slices	Time	Slices	Time		Slices	Time	Slices	Time
Teoma.38	1	11	1	32	Looksmart.16	5	77	4	29
Teoma.35	1	5	1	4	Google.24	2	27	2	89
Looksmart.0	11	91	11	91	Google.15	3	47	3	50
Looksmart.9	3	36	3	22	Yahoo.11	3	55	3	68
Teoma.3	1	37	1	2	Looksmart.12	2	42	2	69
Yahoo.25	11	91	11	91	Looksmart.1	3	34	1	2
Msn.21	1	14	1	4	Google.21	4	48	4	50
Teoma.29	1	15	1	54	Looksmart.33	10	100	11	100
Yahoo.1	1	11	1	5	Msn.18	10	118	10	89
Looksmart.39	7	74	7	91	Google.19	3	30	3	81
Looksmart.40	7	52	7	94	Yahoo.25	5	62	2	20

Table 5 Measurement of z and t for query 3 and 4

Query 3					Query 4				
Relevant document	Baseline		I-FGM		Relevant document	Baseline		I-FGM	
	Slices	Time	Slices	Time		Slices	Time	Slices	Time
Looksmart.18	7	68	12	110	Looksmart.10	6	69	6	50
Teoma.34	23	255	22	161	Looksmart.40	3	65	3	54
Looksmart.7	22	184	12	114	Google.27	8	95	8	99
Looksmart.33	2	29	2	15	Msn.7	7	101	6	44
Yahoo.35	2	18	2	19	Looksmart.9	2	31	2	23
					Google.12	4	75	4	31
					Google.5	7	70	6	76
					Google.26	2	27	2	18
					Msn.23	1	59	1	14
					Teoma.26	3	29	3	50

Table 6 Comparison between the B system and the I-FGM system

	Query1	Query2	Query3	Query4	Average
c	1.615	1.465	1.196	1.33	1.40

this information in [27]. Thus, we added some logs in our code and re-ran the experiments. The measured values of z and time t for every relevant document for each query are shown in the following three tables. The Time columns in Tables 4 and 5 represent the index of interval time in the Blackboard. Note that we record the contents of the Blackboard at each interval time (which is equal to *30 seconds* in this experiment).

From Table 6, we can see that on average the I-FGM system can retrieve the relevant documents 1.4 times faster than the B system. The value of c we measured in experimental results is different from the estimated value from our theoretical analysis. This is mainly due to the fact that in our real search space, the distribution of *nuggets* in a document is not a uniform distribution. In our experiments we found that many documents will achieve their final similarity value within the first (or second) partial processing step. This means that in many documents the *nuggets* are contained in the front of the document. This reduces the advantage of refining document priorities in I-FGM. The other extreme case, i.e. if all documents can obtain their final similarity values at their first partial processing step, also holds less advantage for I-FGM.

7 Conclusion

In this paper, we considered the problem of extracting relevant data from a large and dynamic search space and the challenges faced in providing analysts such data effectively and efficiently. We proposed a solution to this problem through a novel system called I-FGM. We presented the results for the experiments run on a prototype of I-FGM which used the World-Wide Web as the search space focusing on text-retrieval.

By analyzing the results, we have validated the I-FGM system. We compare its performance with the control systems which represent the current paradigms in retrieval systems, and show that I-FGM readily dominates in the majority of cases. The experiments have also shed light on some of the issues we continue to face and have shown that there is room for improvement especially in the design of reliability measures. The theoretical analyses of system performance demonstrate under what conditions and to what extent the partially-and-incrementally processing mechanism will outperform fully processing system. In this paper, we successfully validated the basic tenets of our system both experimentally and theoretically. Most importantly, we have shown that incremental processing of data reinforced by modular system architecture is an effective way to get quick relevant results when dealing with large and dynamic search spaces. Finally, given the massive number of documents that need to be processed for any retrieval algorithm, allocating large numbers of processors in a simple parallel fashion without the intelligent resource allocation will meet with limited success. I-FGM provides the scalability needed to address the modern large-scale dynamic databases that intelligence analysts must work with.

In this work, we have used Document Graphs (DGs) to represent the information in the documents. One advantage of using DGs is that the conversion procedure of text to DGs lends itself favorably to the partial processing paradigm in I-FGM. Another more important reason is that DGs can be used to represent information found in varied heterogeneous documents. We have demonstrated in [26] a methodology of representing the visual information (in images) as concept graphs. This common representation methodology of information found in various types of documents, leads to a unified ranking system in I-FGM. This will be tackled in the next phase of our research.

Acknowledgements The work presented in this paper was supported in part by the National Geospatial Intelligence Agency Grant No. HM1582-04-1-2027. Some preliminary results in this paper can be found in [28, 29]. Also, thanks to Morgan Pittkin for his assistance in this project.

References

1. Bergman MK (2001) White paper: the deep web: surfacing hidden value. J Electron Publ 7(1) doi:10.3998/3336451.0007.104
2. Bhatia SK, Deogun JS (1998) Conceptual clustering in information retrieval. IEEE Trans Syst Man Cybern B 28(3):427–436
3. Bowman CM, Danzig PB, Hard DR, Manber U, Schwartz MF (1995) The harvest information discovery and access system. Comput Netw ISDN Syst 28(1–2):119–125
4. Chen SM, Horng YJ (1999) Fuzzy query processing for document retrieval based on extended fuzzy concept networks. IEEE Trans Syst Man Cybern B 29(1):96–104
5. Chen SM, Horng YJ, Lee CH (2001) Document retrieval using fuzzy-valued concept networks. IEEE Trans Syst Man Cybern B 31(1):111–118
6. Cheng J, Emami R, Kerschberg L, Santos E Jr, Zhao Q, Nguyen H, Wang H, Huhns MN, Valtorta M, Dang J, Goradia HJ, Huang J, Xi S (2005) OmniSeer: a cognitive framework for user modeling, reuse of prior and tacit knowledge, and collaborative knowledge services. In: Proceedings of the 38th Hawaii international conference on system sciences
7. Coden AR, Brown EW (2006) Automatic search from streaming data. Inf Retr 9(1):95–109
8. Craswell N (2000) Methods for distributed information retrieval. PhD thesis, The Australian Nation University
9. Das S, Shuster K, Wu C, Levit I (2005) Mobile agents for distributed and heterogeneous information retrieval. Inf Retr 8(3):383–416
10. Dhyani D, Ng WK, Bhowmick SVS (2002) A survey of web metrics. ACM Comput Surv 34(4):469–503
11. Foster I, Kesselman C, Tuecke S (2001) The anatomy of the grid: enabling scalable virtual organizations. Int J High Perform Comput Appl 15(3):200–222
12. Grossman DA, Frieder O (2004) Information retrieval: algorithms and heuristics. The Kluwer international series on information retrieval. Kluwer Academic, Dordrecht
13. Herlocker JL, Konstan JA, Terveen LG, Riedl JT (2004) Evaluating collaborative filtering recommender systems. ACM Trans Inf Syst 22(1):5–53
14. Hu WC, Chen Y, Schmalz MS, Ritter GX (2001) An overview of world wide web search technologies. In: Proceedings of the fifth world multi conference on system, cybernetics and informatics, pp 356–361
15. Kshemkalyani AD, Singhal M (2008) Distributed computing: principles, algorithms, and systems. Cambridge University Press, Cambridge
16. Meng WY, Yu C, Liu K-L (2002) Building efficient and effective metasearch engines. ACM Comput Surv 34(1):48–89
17. Montes-y-Gómez M, Gelbukh A, Lópes-López A (2000) Comparison of conceptual graphs. In: Proceeding of MICAI-2000—1st Mexican international conference on artificial intelligence. Acapulco, Mexico
18. Nguyen H, Santos E Jr (2007) Effects of prior knowledge on the effectiveness of a hybrid user model for information retrieval. In: Proceedings of the SPIE: defense & security symposium, vol 6536, Orlando, FL
19. Nguyen H, Santos E Jr, Zhao Q, Lee C (2004) Evaluation of effects on retrieval performance for an adaptive user model. In: Adaptive Hypermedia 2004: workshop proceedings—part I, Eindhoven, The Netherlands, pp 193–202

20. Nguyen H, Santos E Jr, Zhao Q, Wang H (2004) Capturing user intent for information retrieval. In: Proceedings of the 48th annual meeting of the human factors and ergonomics society (HFES 2004), New Orleans, LA, pp 371–375
21. Pazzani M, Nguyen L, Mantik S (1995) Learning from hotlists and coldlists: towards a WWW information filtering and seeking agent. In: Proceedings of the IEEE international conference on tools with AI, pp 39–46
22. Salton G, McGill M (1983) Introduction to modern information retrieval. McGraw-Hill Book, New York
23. Santos E Jr, Mohamed A, Zhao Q (2004) Automatic evaluation of summaries using document graphs. In: Proceedings of the 42nd annual meeting of the association for computational linguistics (ACL 2004) workshop on text summarization branches out, Barcelona, Spain, pp 66–73
24. Santos E Jr, Nguyen H, Brown SM (2001) Kavanah: an active user interface information retrieval application. In: Proceedings of the 2nd Asia-pacific conference on intelligent agent technology, pp 412–423
25. Santos E Jr, Nguyen H, Zhao Q, Pukinskis E, (2003) Empirical evaluation of adaptive user modeling in a medical information retrieval application. In: Brusilovsky P, Corbett A, de Rosis F. (eds) Lecture notes in artificial intelligence. User Modeling 2003, vol 2702. Springer, Berlin, pp 292–296
26. Santos E Jr, Nguyen H, Zhao Q, Wang H (2003) User modeling for intent prediction in information analysis. In: Proceedings of the 47th annual meeting for the human factors and ergonomics society (HFES-03), Denver, CO, pp 1034–1038
27. Santos E Jr, Santos EE, Nguyen H, Pan L, Korah J (2005) Large-scale distributed foraging, gathering, and matching for information retrieval: assisting the geospatial intelligent analyst. In: Proceedings of the SPIE: defense & security symposium, vol 5803, pp 66–77
28. Santos E Jr, Santos EE, Nguyen H, Pan L, Korah J, Zhao Q, Pittkin M (2006) Information retrieval in highly dynamic search spaces. In: Proceedings of the SPIE: defense & security symposium, Orlando, FL, vol 6229, pp 1–12
29. Santos E Jr, Santos EE, Nguyen H, Pan L, Korah J, Zhao Q, Xia H (2007) Applying I-FGM to image retrieval and an I-FGM system performance analyses. In: Proceedings of the SPIE: defense & security symposium, vol 6560
30. Santos E Jr, Zhao Q, Nguyen H, Wang H (2005) Impacts of user modeling on personalization of information retrieval: an evaluation with human intelligence analysts. In: Weibelzahl S, Paramythis A, Masthoff J (eds) Proceedings of the fourth workshop on the evaluation of adaptive systems (held in conjunction with the 10th International Conference on User Modeling (UM-05)), Edinburgh, UK, pp 27–36
31. Santos E Jr, Santos E, Nguyen H, Pan L, Korah J, Xia H (2008) I-FGM as a real time information retrieval tool for E-governance. Int J Electr Governm Res 4(1):14–25. Special issue: E-government technologies for managing national security and defense
32. Selberg E, Etzioni O (1995) Multi-service search and comparison using the MetaCrawler. In: Proceedings of the fourth world wide web conference, pp 195–208
33. Sleator DD, Temperley D (1993) Parsing English with a link grammar. In: Proceedings of the 3rd international workshop on parsing technologies, pp 277–292
34. Segaran T (2007) Programming collective intelligence. Building Smart Web 2.0 Applications. O'Reilly Media
35. Song F, Croft WB (1999) A general language model for information retrieval. In: Proceedings of eighth international conference on information and knowledge management, pp 279–280
36. Suan NM (2004) Semi-automatic taxonomy for efficient information searching. In: Proceedings second international conference information technology for application
37. Tanaka H, Kumano T, Uratani N, Ehara T (1999) An efficient document clustering algorithm and its application to a document browser. Inf Process Manag 35:541–557
38. Text REtrieval Conference (TREC) see http://trec.nist.gov/overview.html
39. Verton D (2003) IT deficiencies blamed in part for Pre-9/11 intelligence failure. Computerworld 37(30):12
40. Yates RB, Neto BR (1999) Modern information retrieval. Addison Wesley, Reading
41. Zobel J, Moffat A (2006) Inverted files for text search engines. ACM Comput Surv 38(2). doi:10.1145/1132956.1132959

Eugene Santos, Jr. received his B.S. (1985) in Mathematics and Computer Science from Youngstown State University, a M.S. (1986) in Mathematics (specializing in Numerical Analysis) from Youngstown State University, as well as Sc.M. (1988) and Ph.D. (1992) degrees in Computer Science from Brown University. He is currently Professor of Engineering in the Thayer School of Engineering at Dartmouth College, Hanover, NH. His areas of research interest include artificial intelligence, intent inferencing, social and cultural modeling, computational social science, automated reasoning, decision science, adversarial reasoning, user modeling, natural language processing, probabilistic reasoning, and knowledge engineering, verification and validation, protein folding, virtual reality, and active user interfaces. He has served on many major conference program committees from intelligent agents to evolutionary computing. He is currently Editor-in-Chief for the IEEE Transactions on Systems, Man, and Cybernetics: Part B, an associate editor for the International Journal of Image and Graphics, and is also on the editorial advisor board for System and Information Sciences Notes and on the editorial boards for Journal of Intelligent Information Systems and Journal of Experimental and Theoretical Artificial Intelligence.

Eunice E. Santos received her Ph.D. in Computer Science from the University of California, Berkeley in 1995. She has also received B.S. and M.S. degrees in both Mathematics and Computer Science. She is currently Professor and Chair of the Department of Computer Science at the University of Texas, El Paso. She is also the Director of the National Center for Border Security and Immigration, and the Director of the Center for Defense Systems Research. Her areas of research interest include parallel and distributed processing, modeling and simulation, complex systems, computational biology, computational social science, and socio-cultural modeling. Dr. Santos has received numerous awards, including a National Science Foundation Career Award, the Robinson Faculty Award and the IEEE-CS Technical Achievement Award.

Hien Nguyen is currently Assistant Professor in the Department of Mathematical and Computer Sciences at the University of Wisconsin-Whitewater. She received her Ph.D. in Computer Science from the University of Connecticut in 2005. Her research interests include user modeling, information retrieval, collaborative information retrieval, recommender systems, intent inferencing, and text summarization with a current focus on hybrid user model for improving a user's performance in information retrieval. Professional services and committee work include program committees for 2010 User Modeling Personalization and Adaptation (UMAP) conference, 2008–2010 FLAIRS Conference, 2011, 2007 and 2006 IEEE International Conference on Systems, Man, and Cybernetics. She also is a member of *User Modeling—User Adapted Interactions* journal Special Reviewers Board. She actively involves in supervising undergraduate research at the University of Wisconsin–Whitewater.

Long Pan is a senior software engineer in Microstrategy Inc. He served as a postdoctoral associate in the Laboratory for Computation, Information & Distributed Processing (LCID) at the Virginia Polytechnic Institute & State University for 8 months. He received his Ph.D. degree from Computer Science department of Virginia Tech. He got his B.S. and M.S. degrees from the Automation Department of the Tsinghua University. His research interests are focused on Social Network Analysis (SNA) on large and dynamic networks, SNA with culture infusion, parallel/distributed computing, data mining, and information retrieval. He served as a reviewer for Journal of Supercomputing, and the International Conference on Parallel Processing (2007).

John Korah received his bachelor degree in Electronics and Instrumentation Engineering from the Govt. College of Technology, Coimbatore, India (2000), Masters degree in Electrical Engg (2002) and Ph.D. in Computer Science (2010), both from Virginia Tech. He is currently employed as a Research Assistant Professor in Computer Science, University of Texas at El Paso. His research interests are focused on parallel/distributed computing, parallel numerical algorithms, large scale network modeling/simulation, social networks and information retrieval. He also serves as a reviewer for the Journal of Supercomputing and IEEE Systems, Man and Cybernetics.