

I-FGM - Information Retrieval in Highly Dynamic Search Spaces

Eugene Santos, Jr.¹, Eunice E. Santos², Hien Nguyen³, Long Pan², John Korah², Qunhua Zhao¹ and Morgan Pittkin²

¹ Thayer School of Engineering, Dartmouth College, Hanover, NH, {Eugene.Santos.Jr, Qunhua.Zhao}@dartmouth.edu

² Department of Computer Science, Virginia Polytechnic Institute & State University, Blacksburg, VA, santos@cs.vt.edu, {panl, jkorah, mpittkin}@vt.edu

³ Mathematical and Computer Sciences Department, University of Wisconsin, Whitewater, WI, nguyenh@uww.edu

ABSTRACT

Intelligent foraging, gathering and matching (I-FGM) has been shown to be an effective tool for intelligence analysts who have to deal with large and dynamic search spaces. I-FGM introduced a unique resource allocation strategy based on a partial information processing paradigm which, along with a modular system architecture, makes it a truly novel and comprehensive solution to information retrieval in such search spaces. This paper provides further validation of its performance by studying its behavior while working with highly dynamic databases. Results from earlier experiments were analyzed and important changes have been made in the system parameters to deal with dynamism in the search space. These changes also help in our goal of providing relevant search results quickly and with minimum wastage of computational resources. Experiments have been conducted on I-FGM in a realistic and dynamic simulation environment, and its results are compared with two other control systems. I-FGM clearly outperforms the control systems.

Keywords: Geospatial information retrieval, dynamic information space, distributed information retrieval, parallel information retrieval, multi-agent systems, retrieval performance

1. INTRODUCTION

Over the past decade or so there has been a significant increase in the data gathering capabilities of intelligence and homeland security agencies. The data available to professionals in such organizations include satellite imagery, telephone intercepts and real time information from trouble spots. In short, there has been a quantum leap in the quantity and types of data. Although this has been a boon to the professionals in this field as they now have access to much more information on which to base their analyses, the virtual flood of information has created new problems. These include having to deal with vast amounts of frivolous and unimportant data, and a larger time delay between data reception and its eventual processing. It has also increased the possibility that crucial information might get lost or overlooked in this deluge of raw data. Also different classes of data may have special properties or requirements. For example, real time data has to be processed within a small window of time before it loses its validity. Some other forms of data might have spatial or temporal properties requiring specialized retrieval methods. To add to this already difficult problem, intelligence analysts have to retrieve the relevant information quickly as it may involve critical real-time matters. Sophisticated computing systems and software have been used to retrieve relevant data from large databases for some time now. Current space reduction techniques based on filtering¹, taxonomies² and document clustering^{3,4} do not provide adequate solutions to the problem of retrieval in large databases. It is becoming evident that the computing resources are not keeping up with increase in the amount of data available and this is unlikely to change in the future. The retrieval community has put its primary focus on developing new retrieval methods; and paid much less attention to resource allocation strategies for computing resources. This is short-sighted, especially when dealing with large and dynamic databases, as intelligent and judicious usage of computing resources are crucial in this case. It is clear that a comprehensive solution employing innovative software architecture, information processing paradigms, and resource allocation strategies is required to achieve our performance objectives.

In a previous paper⁵, we discussed how I-FGM (Intelligent Foraging, Gathering and Matching) is an effective tool for intelligence analysts who have to deal with large and dynamic search spaces. I-FGM combines the strengths of a multi-agent architecture with a new information processing paradigm. The essential idea of the I-FGM system is to partially and incrementally process data, analyze the partial results, and allocate computing resources accordingly.. The basic unit of the system is a process, and groups of processes work together to perform certain functions. The multi-agent architecture helps to facilitate timely and flexible allocation of computing resources. It is also modular which allows for the easy assimilation of third party tools required for accessing different types of data. The main aim of I-FGM is not to develop new methods of information retrieval on the lines of vector space method or language model method⁶ but rather how the effectiveness of current methods can be improved by using partial information processing. In order to provide a robust system with better performance, we refined our I-FGM system based on our prior experimental results and theoretical analysis. More importantly, we have now simulated a realistic highly dynamic search space and verified the effectiveness of our system on this testbed. In this paper, we focus on some of our key modifications and development of I-FGM for text retrieval. Complete details of I-FGM can be found in our earlier papers^{5,7}.

One of the key refinements made in this paper is of the I-FGM priority function. In the system, individual data or information nugget is assigned a priority which represents the potential that the nugget may contain information relevant to the user. I-FGM updates the priority based on its partial analysis of the nugget, and this guides further resource allocation. Estimating the priority for information nuggets is an important and difficult task which requires both careful theoretical study, and experimental analysis. Furthermore, a dynamic search space introduces additional obstacles. While the system is running, new and old documents may have been partially processed to different degrees. The priority of older documents may be more refined than newer ones and hence comparing the priority values of these two documents may be “unfair” to new documents. We need to provide a level playing field when comparing the partial results belonging to documents arriving at different times. In order to accomplish this, we use a half-life decay mechanism to calculate priority. The nugget importance will decrease as the total process time increases. Another key refinement is the modification of our reliability function. This function is used to calculate a reliability metric for each I-Forager.. This metric is required to characterize the performance of a particular I-Forager. In the earlier prototype, the reliability for all the I-Foragers were equal and kept constant. We have introduced a new reliability function that is dependant on the percentage of relevant documents downloaded by the particular I-Forager. Hence, the reliability of an I-Forager can change. Another change is the new resource allocation strategy. Previously, I-FGM picked the document with the highest priority and allocated resources to it. It was observed during our experiments that a document selection strategy that has an element of randomness performed better than always choosing the highest-priority document. In the new prototype, we select the top 10 documents and pick the document to be processed using a biased function. The last major improvement in this paper is in the range and scope of experimental runs. In the previous prototype, we provided experimental results for non-dynamic databases. Although the documents used in the previous experiments were retrieved from the internet, it does not provide us with the kind of dynamism for which I-FGM is designed. In this paper we simulate dynamic databases by providing documents to the system at different points in simulation time. We simulate two types of dynamic search spaces, termed as quasi-dynamic and fully dynamic. As the names suggest, they represent two different degrees of dynamism in search spaces. In the quasi-dynamic simulations, only the relevant documents are given to the system at different times. I-FGM has access to all other documents from the beginning. We can see this as a condition in which the load on the system is constant. In the full dynamic set of simulations, each document is given to the system at some random time. This represents a condition of highly varying dynamism.

In the following sections, we give a brief overview of the system architecture followed by the description of the control systems used in the experiments. The changes made in this prototype are explained in detail. This is followed by a description of experimental setup. The last two sections deal with the analysis of experimental results and the conclusion.

2. RELATED WORK

In this section, we describe some of the technologies that have been used in the current implementation of I-FGM. Since this prototype deals primarily with text data, we used the current web search technologies to gather information from the Internet. In particular, metasearch^{8,9} technology has been used to gather information from multiple search engines and to integrate the results into a single database. The issue of providing a common ranking to the results has been studied to some extent. Ontology networks is another area that is leveraged in I-FGM. The information nuggets are processed and

represented as a graph of nodes and edges called document graphs. The nodes represent the concepts in the nugget and the edges represent the relations between the concepts. The graphs represent the content in the nugget in a semantic form. The document graphs can be compared to calculate the relevancy. The document graphs were chosen as they lend themselves well to partial processing, and the partial relevancy can be calculated in an incremental manner. We especially use the work in ^{10,11} as the basis for the document graphs in our prototype.

Although there has been some work in retrieval architecture design for distributed databases, there is very little work in retrieval for dynamic databases. Also, the scope of current research in this field is limited in its goals. A comprehensive solution is missing in many of these works. We cite the Harvest System¹² as a system that has some architectural similarity to I-FGM. It is used to gather data from only WWW websites. The main differences between it and I-FGM are its lack of a resource allocation strategy, and its static architecture configuration.

3. SYSTEM ARCHITECTURE

3.1 Description

In this section we provide a brief overview of the I-FGM architecture. A more detailed description can be found in the previous paper⁵. In order to facilitate rapid deployment of computational resources, easy prototyping, and the partial information processing paradigm of I-FGM, a multi-agent architecture was adopted. The basic unit of the system is a process. Processes performing similar tasks are grouped together in functional components. I-FGM system has the following components:

1. I-Forager
2. gIG-Soup
3. gIG-Builder
4. I-Matcher
5. Blackboard

I-Forager: The function of I-Foragers is to retrieve promising information from the various databases available, based on the query it receives from the user. It uses widely available search technologies such as web search engines. The information thus obtained is placed in a repository called the gIG Soup. Each search technology has different levels of effectiveness, and this is quantified using a parameter called the *reliability*. The reliability is used to calculate the *first-order similarity* of the documents in the gIG Soup. The first-order similarity is used to calculate an initial ranking or priority for the document within the gIG Soup. In this prototype of I-FGM, we implemented 5 I-Foragers to retrieve documents from the WWW. Each I-forager used a different search engine. The search engines used are Google, Yahoo, MSN, Teoma, and Looksmart. The downloaded documents are first “cleaned” by removing HTML tags and other useless data, and then placed in the gIG Soup. The documents are then processed by gIG-Builders.

gIG Soup: is used to hold documents that are retrieved by I-Foragers, and the processed form of the documents produced by the gIG-Builder. In our current implementation, the processed form of a document is a document graph. Currently, the gIG Soup is implemented as a Network File System (NFS) directory that is shared by all the nodes in the system. In our current implementation, gIG Soup also contains database tables (implemented using MySQL) to store important information about the documents in gIG Soup, such as file name, URL, identification for a query, number of sentences parsed and so forth. These tables also store the partial and full relevancy measure of the documents in the gIG Soup.

gIG-Builder: Each document in the gIG Soup has a priority value. Priority of a document is used to quantify how potentially relevant it is to the given query. Each gIG-Builder uses selection criteria based on priority value to choose a document for processing. It then proceeds to process the document for a period of time called the parsing time. The parsing time is again dependant on the priority value. Thus the processing is done in an incremental manner.

In the current implementation, 12 nodes are used as gIG-Builders. The gIG-Builders continuously query the gIG Soup, specifically its MySQL database, for documents that are not fully processed. The information about the selected document such as priority and parsing time are obtained from the MySQL database. Each gIG-Builder then proceeds to

construct its document graph for the given period of time. After this time is up, the document graph is placed back into the gIG Soup. Information such as the number of lines parsed are updated in the MySQL database. Since there are multiple gIG-Builders working on the gIG Soup, appropriate flags are set to ensure that they do not interfere with each other while accessing the database and the documents from gIG Soup.

I-Matcher: The function of the I-matcher is to compare the query with the processed documents and calculate a similarity value. In this prototype, the document graph of a document that has been through a process cycle with the gIG-Builders is compared with the query graph. A similarity value is calculated using the following formula

$$sim(q, d_i) = \frac{n}{2 * N} + \frac{m}{2 * M}$$

where q is the query graph, d_i is the document graph, n and m are the number of concepts and relation nodes of the query graph found in the document graph, and N and M are the total numbers of concept and relation nodes of the query graph. Note that two relation nodes are matched if and only if at least their parent and their child are matched. The similarity value is used to update the priority value of the document. The parsing time for the next process cycle is also calculated by the I-matcher. The afore mentioned document attributes are then updated in the MySQL database.

Blackboard: The blackboard continuously displays the top ten most relevant documents. This is important as the top ten document list changes as the experiment proceeds.

3.2 Prototype Specific modifications

Based on the analysis of the experimental results presented in our previous work⁵, many important refinements were made in the system metrics, such as priority and reliability. A number of issues that degraded the system performance were noted and remedied in this paper. We now describe the changes made in the new I-FGM prototype.

Changes in Priority function: The priority function is used to determine the priority of a document and thus determine the order in which the documents will be processed. This guides the allocation of computing resources. Priority values are calculated by the I-matcher. The priority function in the previous prototype⁵ is based on its previous value, current similarity value of the documents, and the average change in the document similarity. The priority formula is as follows:

$$\begin{aligned} P_k &= \beta_1 p_1 + \beta_2 p_2 \\ p_1 &= \delta P_{k-1} + (1 - \delta) Sim_k \\ p_2 &= \begin{cases} \Delta_k & \Delta_k > 0 \\ -\Delta_0 \frac{t_k}{t_0} & \Delta_k = 0 \end{cases} \end{aligned}$$

where P_k represents the priority of a document d at time k . It is computed from the priority at time $(k-1)$ for this particular document, P_{k-1} , and the current values of similarity. Sim_k is the similarity values of the document at time k . Δ_0 is the average expected similarity for the maximum parsing time (t_0), and t_k is the parsing time allocated for the document at time k . δ, β_1, β_2 are the scaling factors.

In our previous prototype, gIG-Builders compared the priorities of the documents in the gIG Soup and selected the most promising document, i.e. the current highest-priority document. This simple approach can cause problems; the relevant documents that have been in the gIG Soup for some time have a more refined similarity value in comparison with relevant documents that have just come in. It is easy to see that the old documents will dominate the resource allocation and hinder new documents from being allocated processing time. Also it is “unfair” to the new documents since we are comparing similarities based on the evaluation of unequal portions of the document data. We seek to remedy this by introducing a weight (in the form of a half-life decay function) to the similarity value of a document, which will also take into account the total amount of time that the document has been processed. The half-life decay function has been used to calculate the similarity of a document at the k^{th} step in the following way:

$$Sim_k = Sim_k \cdot e^{-t/h}$$

So we are “decaying” the similarity value of the document according to its total processing time.

By using the decay function, we are improving the chances that newly arriving documents will be processed early. Our simulation results will show that the wait time of the document in the gIG Soup will be reduced by using the decay function. The decay function is also useful because it makes the system more responsive to sudden and new developments in the field which is a valuable asset for the intelligence analyst.

Resource allocation strategy: In the previous paper⁵ the simulation results revealed two important factors that we had overlooked:

1. Multiple documents can have the same priority value. Since the gIG-Builders selected the top document based on priority, the document (returned in alphabetic-order by the database) selected may have lower potential than the documents following it. This will affect the performance of the system.
2. Close similarity values: The priority values of the documents can be very close to each other. So the top document may win in the resource allocation competition by a very small margin. We need to give some consideration to the document that loses out in such a case.

In this paper, we use an additional document selection strategy in the gIG-Builders. In this strategy, the top 10 documents are selected and the document to be processed is picked using a biased function. This strategy is used in tandem with top priority selection. Hence we have two types of gIG-Builders. One type uses top priority selection and the other uses biased selection. We have used them in equal numbers in the experimental setup.

Metrics: In the previous prototype⁵, recall was the parameter used to measure performance. This provides a macro-level evaluation,, but gives little information of how individual documents are treated by the system. We have introduced the discovery time metric which is defined as the amount of time a document spends in the gIG soup before it is processed. IFGM tries to minimize the discovery time for relevant documents. The discovery time of the documents will be presented as bar graphs and will be compared against those of control systems.

Calculation of the Reliability function: Reliability is the metric attributed to an I-forager measuring results from the I-forager. In the previous prototype⁵ we gave the same reliability value to all I-foragers. This is a simplification that we remedy in this paper. The reliability is calculated using the following formula:

$$reliability = \alpha_1 \beta \frac{n}{k} + \alpha_2 s + \alpha_3 d$$

where A is the set of relevant documents for a query, $B \subseteq A$ is the set of relevant documents returned by the I-Forager,

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

α₁, *α₂*, *α₃*- scaling factors

The reliability is calculated by running test data on the system, and is refined after each experimental run.

In the previous prototype, when a new document enters the gIG Soup, its initial priority is calculated using the reliability. This presents a problem when we are comparing the priorities of the documents, because older documents have a more refined priority than the new documents. There is a need to scale it and this is done by using the

neighborhood priority. We are using the rank returned by the I-forager to determine the neighborhood of the document. We have the assumption, that the priority of a document is related to the priority of its neighborhood. The size of the neighborhood is dependant on the reliability of the I-forager: the lower its reliability value, the larger its neighborhood. The initial priority is calculated as

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

4. CONTROL SYSTEMS

In order to verify effectiveness, IFGM is compared with current methodologies for information retrieval in large and dynamic search spaces. We construct two control systems called the baseline and partially intelligent systems, to represent the current state of the art. They have the same components as I-FGM but differ in the manner in which they function and interact with each other. In the baseline system, the documents are randomly picked for processing for an arbitrary period of time. In other words, the potential of all documents to be relevant is considered to be equal. The partially intelligent uses the first-order similarity ranking of the documents to choose the next document for processing. This priority does not change and hence the processing is based on the relevancy measure provided by third party tools used in the I-Foragers. By comparing the performance of the control systems with I-FGM we prove that the naïve methodology of randomly picking a document or the strategy of only considering the first order similarity of the documents is not enough to achieve the best performance. The control systems of I-FGM are implemented on a set of 19 machines. Each machine has 1GHz Pentium III processor with 256 MB RAM. The machines are interconnected with 100 MBPS Ethernet network. 5 machines are used as I-Foragers, 12 as gIG-Builders, 1 machine as an I-Matcher, and the remaining 1 machine as the blackboard and gIG Soup.

5. EVALUATION

I-FGM is evaluated by comparing its performance with the control systems. A set of 5 queries are run on the systems, and results are collected and analyzed. Recall is one of the performance metric used here. Recall is a standard metric in information retrieval and is defined as the ratio of the relevant documents retrieved to the total number of relevant documents. One of the stated goals of the I-FGM is to achieve full recall quickly. In order to validate this, the recall values are collected at regular intervals during the experimental runs. Another performance metric that is unique to dynamic database retrieval is the time a relevant document waits (in the gIG Soup) before it is retrieved. I-FGM seeks to minimize this wait time for all relevant documents. We evaluate the performance of IFGM by comparing the wait times for relevant documents in the control system and I-FGM.

For the experiments, we use a set of 5 queries which are related to the recent Tsunami disaster scenario. This scenario was chosen because the information on websites related to the tsunami was changing rapidly. Moreover, there was a sudden jump in the number of websites dealing with this topic. The set of queries are:

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

A testbed was created for each query by storing the top 50 results from each of the five search engines used in the I-Foragers. In order to use recall as a performance metric, we designate the top 10 relevant documents as the target document set.

5.2 Procedure for simulating dynamism

In the previous paper⁵, we had simulation results for static data. All the documents were available to I-FGM at the beginning of the experiments. In this paper we evaluate the performance of I-FGM in dynamic search spaces. Though the WWW is dynamic in some sense, its rate of change is much slower than real time data. Hence there is a need to simulate a higher degree of dynamism in our testbed by some other means. We do this by inserting documents into the search space at different points in time. We have simulated two types of dynamic spaces termed as: quasi-dynamic

search space and fully-dynamic search space. In the quasi-dynamic search space, we introduced the top 10 documents at some pre-defined points in time. All other documents of the testbed are available to the I-FGM system at the start of the experiment. In the fully-dynamic search spaces, all the documents enter the gIG Soup at some random time. This is used to simulate a highly dynamic search space. By analyzing the results from these two sets of experiments, we will show that I-FGM is suitable for both dynamic conditions. The quasi-dynamic experiments were conducted on the I-FGM system with three different values of half-life periods in its priority function. The quasi-dynamic simulations serve two purposes: to validate the performance of I-FGM in a low dynamism search space and to select a suitable value for the half-life. This half life period is used in the fully dynamic experiments.

5.3 Procedure for evaluating retrieval performance.

For each query:

- Run all three systems until all documents have been completely processed.

For each run, record the following data:

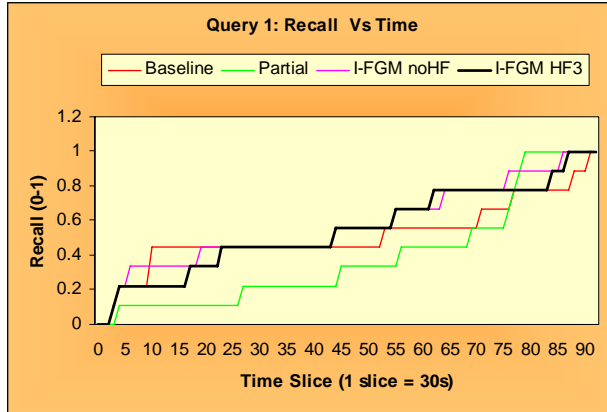
- Blackboard content: Define an interval amount of time t_i ($t_i = 30$ seconds in this experiment). We record the contents of the blackboard after each interval time t_i .
- Data used for computing reliability for each I-Forager: Number of documents, retrieved by a particular I-Forger that appeared in the final blackboard content.
- The time of arrival of each document into the gIG Soup is noted.

6. RESULTS AND ANALYSIS

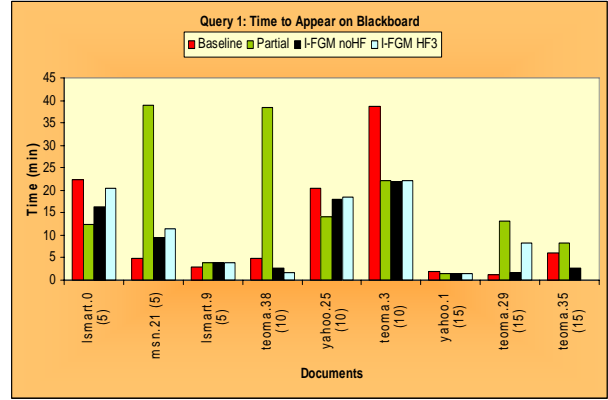
The procedure described in Section 5.2 is used to conduct the experiments. The first set of experiments is done to determine an appropriate period for the half-life function to be used in the I-FGM system. Three queries are run with baseline, partial and I-FGM variants. The variants of I-FGM used here include I-FGM noHF, which does not use a half life function, I-FGM HF1 which uses a half-life function with period 200ms, I-FGM HF2 that has a half-life function with period 857ms and I-FGM HF3 with period 514ms. These particular values of half-life period were chosen based on the average parsing time and average length of documents. By comparing the performance of these systems we can select the half-life that gives the best performance.. As mentioned before, these simulations are quasi-dynamic. Table (Fig 1) gives the number documents that each I-FGM variant discovers first for each of the query. From Fig 1, we see that I-FGM HF3 gets the most number of documents among the I-FGM variants. It also performs better than baseline and partial systems. Hence, I-FGM HF3 with half-life period=514ms has the suitable value of the half-life period. IFGM-HF1 does not perform as well because the half-life time is too small. It degrades the priority of documents quickly. Hence documents that contain relevant portions in latter part of the document are adversely affected leading to poorer performance. The half life in I-FGM HF2 is large and will make the system perform similar to that of I-FGM noHF, as seen from the experimental results. The system with HF3 is designed with an in-between half-life decay time and it discovers more documents than any of the I-FGM systems.

	I-FGM noHF	I-FGM HF1	I-FGM HF2	I-FGM HF3
Query1	5	4	5	5
Query2	2	5	4	6
Query3	2	3	2	4

Fig 1: Number of documents discovered first by I-FGM systems

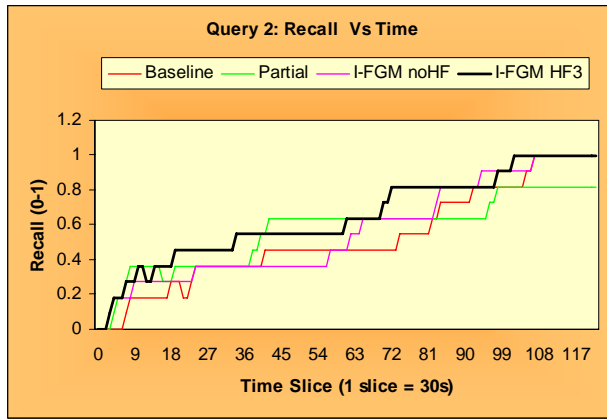


(a)

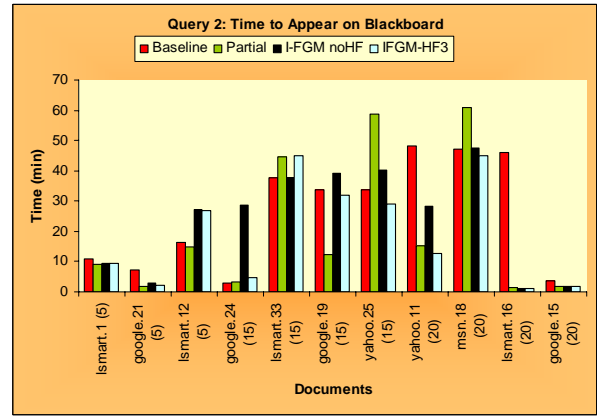


(b)

Fig 2: Query 1

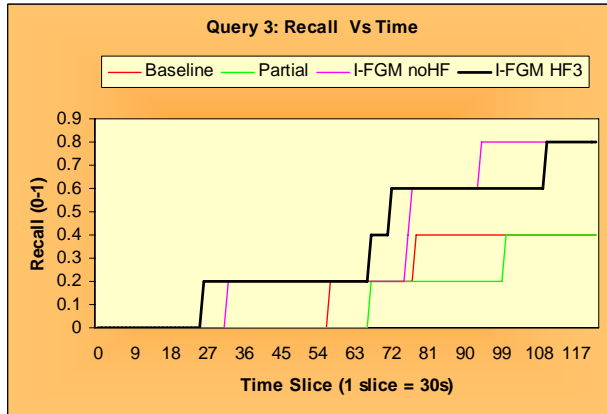


(a)

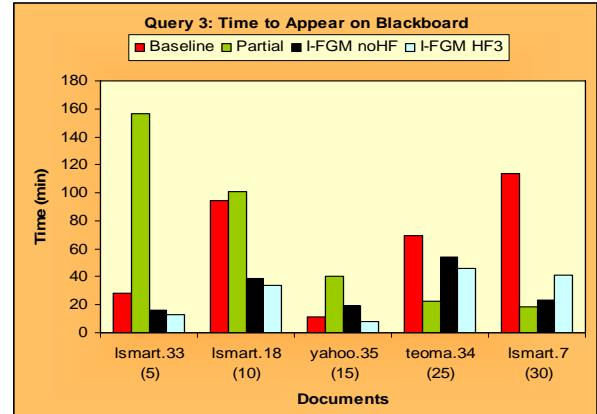


(b)

Fig 3: Query2

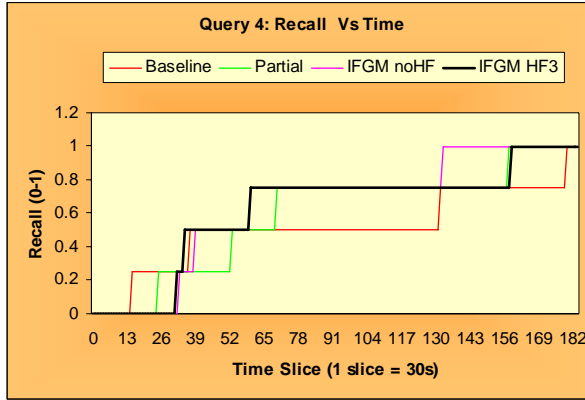


(a)

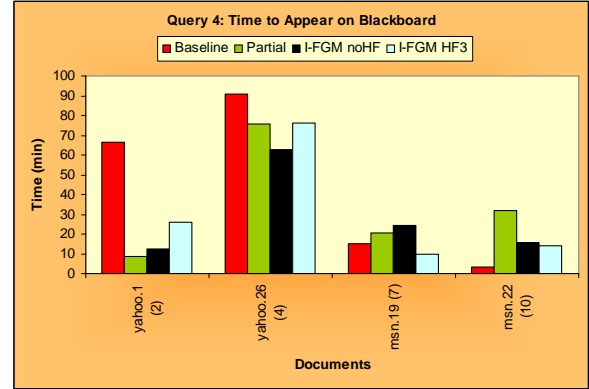


(b)

Fig 4: Query3

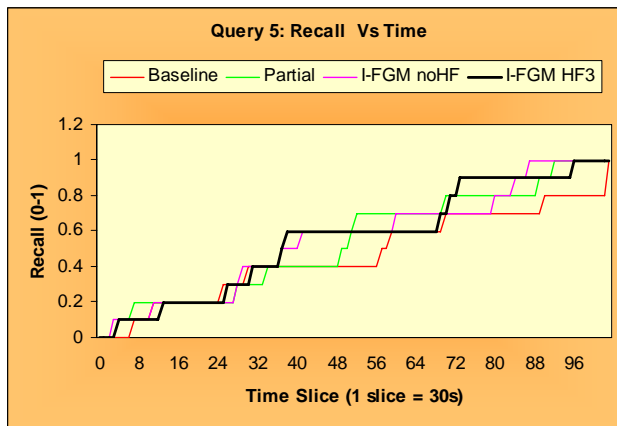


(a)

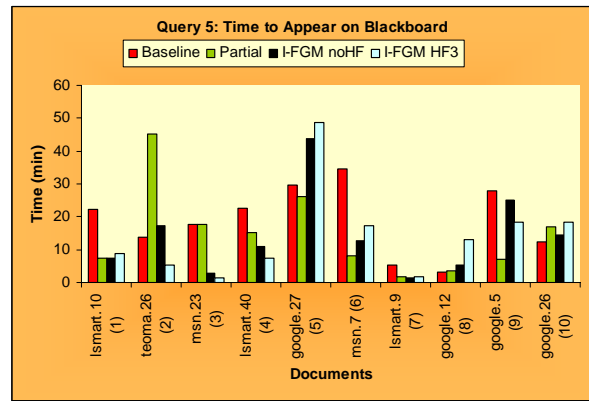


(b)

Fig 5: Query 4



(a)



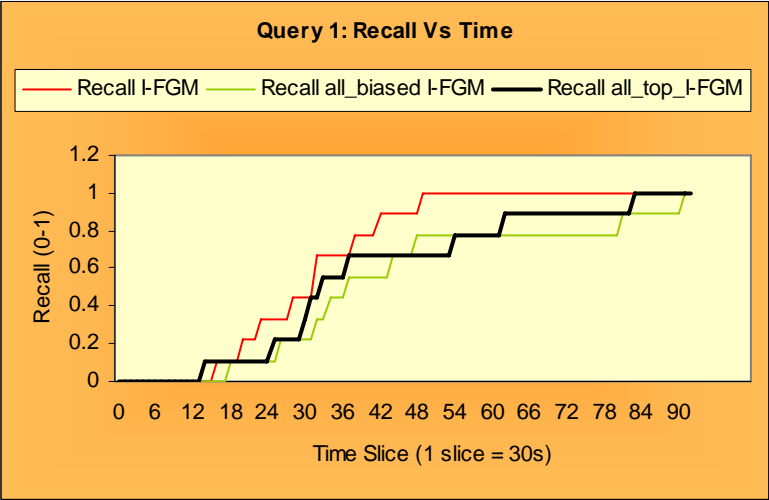
(b)

Fig 6: Query5

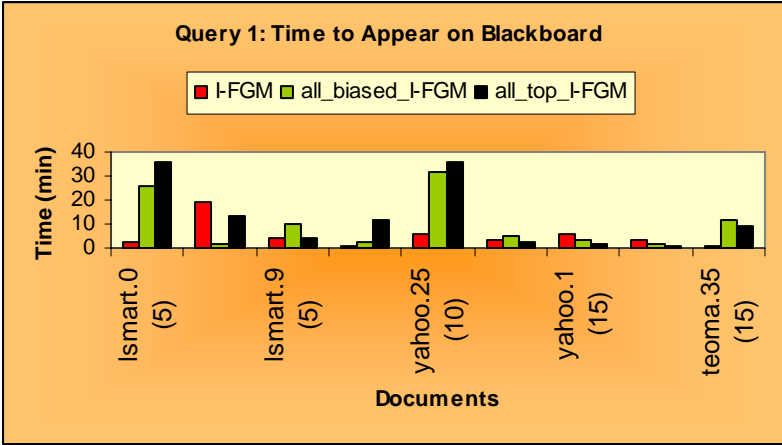
In the second set of experimental results, the 5 queries are run with the control systems, I-FGM with no half-life (I-FGM noHF), and I-FGM with half-life (I-FGM HF3). The half life period used here is 514ms. The simulations are performed in fully dynamic conditions. For each query, we generate the recall vs time graph and the discovery time bar graphs for the target document set. From the recall graphs we see that in all the queries, I-FGM has a higher rate of climb than any other system. This demonstrates the merit of employing the half life function. When we analyze the discovery time results, we see that IFGM HF3 obtains 5/9 (no. of documents discovered by I-FGM HF3/total no. of relevant docs) for query1, 7/11 for query2, 3/5 for query3, 1/4 for query4 and 4/10 for query5. In queries 1,2, and 3, IFGM HF3 gets more than 50% of the relevant documents faster than other systems. In query 4 each of the systems gets a document. But if the baseline system is discounted, then IFGM HF3 has the best performance among the rest of the control systems for query4. It may be noted that the baseline system, which depends on random selection of documents for processing, can be shown to be suitable for fast and dynamic search spaces only if it performs better than other systems in a majority of the cases. If baseline performs better only in a few cases, it can be attributed to chance and disregarded. We see that in some queries, the performance of partial equals that of IFGM HF3. This can occur since most of the relevant documents are ranked high by the search engines. In such cases, the reliability provided by the search engines is a good approximation for relevancy of the documents. Partial systems which are based on reliability work naturally in such cases.

Another major change adopted in the current prototype is the new resource allocation strategy. As mentioned before, the current prototype uses equal numbers of the two types of gIG-Builders: one type selects the document with the highest priority, and the other type uses a biased function to select the document. We have made the determination that this mix of gIG-Builders gives better performance than simply using one or the other. This was done by analyzing the data

gathered using the previous prototype. We validated this hypothesis by conducting experimental runs with the query set using three I-FGM variants. The first I-FGM variant (all_top_I-FGM) uses only the simple top document selection strategy. The second I-FGM variant (all_biased_I-FGM) uses only the biased selection function, and the third uses both these strategies. We present the results for query 1 (Fig 7) only, due to lack of space. It shows that I-FGM clearly delivers better performance than the other two. From fig 7(b), we see that out of 9 documents, the I-FGM system discovers 4 documents earlier than the other variants and ties with all-top system for 2 other documents. The all_biased and all_top systems are clear winners for only one document each. When comparing the results for just all-top and all-biased I-FGM systems, we see that each variant wins in 4 documents and ties for one. From fig 8(a), we see that I-FGM maintains better recall than the other two systems throughout the experiment. Similar conclusions are derived from the results based on other queries. Hence, we have shown that using a combination of the two selection strategies combines their strengths and gives better performance.



(a)



(b)

Fig 7. Comparison of Document Selection strategy: Query 1

7. CONCLUSION

In this paper, we present improvements resulting in the second prototype of the Intelligent Foraging, Gathering and Matching (I-FGM) system. A number of changes in the system parameters were made after analyzing the results of the

first prototype. These changes were necessary to improve the performance of the system and also to remove some of the simplifications which were made in the last prototype. The main changes were made in the priority and reliability functions of the system. The system was tested in two different conditions of dynamism quasi-dynamic and fully-dynamic. The results were analyzed, and it was shown that the changes that were made did indeed improve the system performance.

The design and implementation issues with respect to text retrieval have been completed. In the next phase, retrieval of other forms of data such as images will be incorporated. Comparing the relevancy of image and text data is another crucial problem that needs to be addressed. The performance modeling of the system is another focus area, as one of the goals of the project is to come up with performance guarantees for the system. In conclusion, the Intelligent Foraging, Gathering and Matching (I-FGM) system has been shown to work well in the highly dynamic conditions. Also the ease of changing the characteristics of the system has been demonstrated, further strengthening the case for its modular architecture.

ACKNOWLEDGEMENTS

The work presented in this paper was supported in part by the National Geospatial Intelligence Agency Grant No. HM1582-04-1-2027.

REFERENCES

- [1] M. Pazzani, L. Nguyen, and S. Mantik, "Learning from hotlists and coldlists: Towards a WWW information filtering and seeking agent," Proceedings of the Seventh International Conference on Tools with Artificial Intelligence, 1995.
- [2] S. N. M. Melvyn, "Semi-Automatic Taxonomy for Efficient Information Searching," Proceedings of the 2nd International Conference on Information Technology for Application 2004.
- [3] H. Tanaka, T. Kumano, N. Uratani, and T. Ehara, "An efficient document clustering algorithm and its application to a document browser," *Information Processing and Management* vol. 35, pp. 541-557, 1999.
- [4] W.-C. Hu, Y. Chen, M. S. Schmalz, and G. X. Ritter, "An overview of World Wide Web Search Technologies," Proceeding of SCI2001-5th World Multi-conference on System Cybernetics and Informatics, 2001.
- [5] E. Santos, Jr., E. E. Santos, H. Nguyen, Q. Zhao, L. pan, and J. Korah, "Large-scale Distributed Foraging, Gathering, and Matching for Information Retrieval: Assisting the Geospatial Intelligent Analyst," Proceedings of SPIE Defense and Security Symposium, Orlando, Florida, 2005.
- [6] C. Faloutsos and D. W. Oard, "A survey of information retrieval and filtering methods," Computer Science Technical Report Series, Vol. CS-TR-3514, University of Maryland at College Park, 1995.
- [7] E. Santos, Jr., E. E. Santos, and E. S. Santos, "Distributed Real-Time Large-Scale Information Processing and Analysis," Proceedings of SPIE Defense and Security Symposium, Vol. 5421, pp. 161-171, 2004.
- [8] W. Y. Meng, C. Yu, and K.-L. Liu, "Building Efficient and Effective Metasearch Engines," *ACM Computing Surveys*, vol. 34, pp. 48-89, 2002.
- [9] E. Selberg and O. Etzioni, "Multi-Service Search and Comparison Using the MetaCrawler," Proceedings of Fourth World Wide Web Conference, pp. 195-208, 1995.

- [10] E. Santos, Jr., H. Nguyen, and M. S. Brown, "Kavanah: an Active User Interface Information Retrieval Agent Technology," Proceedings of the Asia-Pacific Conference on Asian Intelligent Agent Technology, pp. 412-423, Oct, 2001.
- [11] E. Santos, Jr., H. Nguyen, Q. Zhao, and E. Pukinskis, "Empirical Evaluation of Adaptive User Modeling in a Medical Information Retrieval Application," Proceedings of the Ninth User Modeling Conference, pp. 292-296, 2003.
- [12] C. M. Bowman, P. B. Danzig, D. R. Hard, U. Manber, and M. F. Schwartz, "The Harvest Information Discovery and Access System," *Computer Networks and ISDN Systems*, vol. 28, pp. 119-125, 1995.