INTELLIGENT RETRIEVAL
Open Journal

Review

Large Scale Distributed Foraging, Gathering, and Matching for Information Retrieval: Assisting the Geospatial Intelligence Analyst

Eugene Santos Jr.[1*]; Eunice E. Santos[2]; Hien Nguyena[1]; Long Panb[1]; John Korahb[1]

[1]*Computer Science and Engineering Department, University of Connecticut*
[2]*Department of Computer Science, Virginia Polytechnic Institute and State University*

[*]***Corresponding author***
Eugene Santos
Computer Science and Engineering Department, University of Connecticut

Article information
Received: June 26th, 2024; **Revised:** August 20th, 2024; **Accepted:** August 31st, 2024; **Published:** September 21st, 2024

Cite this article
Santos E, Santosb EE, Nguyena H, Panb L, Korahb J. Large scale distributed foraging, gathering and matching for information retrieval: Assisting the geospatial intelligence analyst. 2024; 2(2). doi: https://doi.org/10.70705/ppp.ir.2024.v02.i02.pp75-81

ABSTRACT

There is a growing need for quick and effective data retrieval from dispersed geospatial databases due to the abundance of internet resources. The enormous and ever-changing nature of geospatial datasets is one of the main obstacles to solving this issue. To solve this issue, we create an I-FGM framework for huge and ever-changing information spaces that is distributed and designed for intelligent foraging, collecting, and matching on a wide scale. To determine how well our method works, we pit a prototype I-FGM against two control systems: one using randomized selection and the other using semi intelligent algorithms. To ensure that we were measuring each system's retrieval accuracy and recall accurately, we constructed and used a medium-sized testbed. The results demonstrate that compared to the other two control methods, I-FGM collects pertinent data at a faster rate.

Keywords
Geospatial information retrieval; Dynamic information space; Distributed information retrieval; Parallel information retrieval; Multi-agent systems; Retrieval performance.

INTRODUCTION

Due to the increasing demand for geospatial information by users like intelligence analysts—particularly in emergency situations—the limitations of current commercial search engine technologies severely limit their ability to access this data, making geospatial data retrieval from large, dynamic databases an important area of research. Geospatial data includes not only physical features like shape, distance, measures, and geophysical structure, but also demographics like population, lodging availability, school rankings, and healthcare providers. In the end, any GIS system worth its salt will quickly and accurately provide the user with all the pertinent data they need. Current technology, like general-purpose search engines, isn't suitable for this task since it would provide an excessive amount of irrelevant results, which would be overwhelming for the user. The fact that a real-time solution is necessary due to the ever-changing nature of information adds another layer of difficulty. The massive volume of data we need to go through also makes it hard to generate quick, relevant results. For large-scale, real-time geographic information retrieval, we need a method [4]. Intelligence analysts rely on this duty heavily since they must swiftly extract pertinent geographical data from enormous datasets. The capacity to quickly access pertinent data is essential for intelligence analysts to make sound decisions, which in turn impacts the success or failure of operations.

In this paper, we create a large-scale distributed real-time system that uses anytime intelligent foraging, gathering, and matching (I-FGM) to efficiently and effectively retrieve relevant information from a massive and ever-changing geospatial information space. The I-FGM technology is intended to work with many agents. Upon receiving a query, the system initiates many concurrent processes to search the geographic space for relevant data points. Each piece of knowledge or piece of partial geographic information is represented by a directed acyclic graph. Due to the enormous and ever-changing nature of the search space, the conventional approach of fully analyzing a nugget to ascertain its relevance is computationally demanding and inefficient, especially when considering the impossibility of giving real-time results employing that strategy. To get around this issue, I-FGM calculates a nugget's partial similarity measure after partly processing it. The system decides whether or not to provide this nugget additional resources based on this partial evaluation. Thus,

the distribution of the limited resources is dictated by the search space. The user may see the most relevant information nuggets as the system processes them progressively. This serves a dual purpose: first, it displays the most recent relevance metrics of the dynamic knowledge nuggets; and second, it guarantees that consumers obtain results in real time. The ability to simply "plug in" IR technologies is a key feature of IFGM's modular design. This makes it simple to incorporate new technology into the system. You can run several processes simultaneously with I-FGM since it is a multi-agent system, which makes it very scalable. This facilitates load balancing by facilitating the simple and rapid reallocation of computing resources. Although similarity measures are obviously crucial to the system, it is not easy to formulate the similarity function such that it takes into account the final relevance value of the partial knowledge nuggets when calculating the similarity measure. The dynamic nature of the data collection, intelligent selection of potential knowledge to process, partial and gradual processing of knowledge, and rapid presentation of findings to the user are the primary strengths of this technique.

By combining I-FGM with the baseline and the somewhat intelligent foraging, collecting, and matching system, we test how well our method works. To measure how well the system works, we employ the two most used metrics in the field of information retrieval: recall and precision. We calculate the efficiency by calculating the amount of labor involved and the amount of time the systems take to operate.

The structure of this article is as follows: We begin by providing a concise synopsis of relevant literature. We will now show you the I-FGM architecture. Our experimental setup, as well as the data and analysis presented, are detailed after this. Our findings and plans for the future will be detailed thereafter.

2. RELATED WORK

Researchers in the fields of image processing and information retrieval have recently concentrated on creating distributed geographic information retrieval systems that can obtain pertinent data about a specific location. Mapquest, one of the well-known industry-based conventional apps, lets users search a static database for geographic information related to a given location (http://www.mapquest.com). Also, as mentioned in [2, 3, 5, and 11], researchers have investigated both text-based and content-based information retrieval methods for picture retrieval. But all the current tech works on the premise that databases are immutable, so when you get anything from them, all the data is there and unaltered. But as the Internet continues to expand, new material is constantly being added and updated. Take the 2004 Final Four Tournament in New Orleans as an example; if we were to look for an after-game report five minutes after the game ended, we may receive the final score, but not much more about the game. But the full game report, including the score, the players, and post-game statements from couches and players, will be available fifteen minutes after the game. Worse still, information is in a perpetual state of flux, with new crucial data being generated at a dizzying pace, especially in times of crisis like September 11, 2001. Take into account the several tiers of intelligence analysts, from those working with a single source to those with access to all available sources on a national level, as well as the resulting products. At the most basic level, these analysts working with a single source are producing massive volumes of geographic data that must be made known to the ever higher-level analysts that rely on them. The static assumptions of modern search and retrieval technologies may lead them to try huge parallelism as a solution. Such an easy fix, however, is ineffective and wasteful when faced with the accompanying enormous volumes of useless data.

We base our method on the premise that data collection from a dynamic space of information must be done in real time. The hypertext metacrawler developed at the University of Washington is an early example of a similar effort [10, 12, 13]. This kind of effort is relevant to our strategy because of the way they combine the top results from major search engines. Various search engines' retrieved documents for the same query and their ranking algorithms were the subjects of tests conducted in this study. This ties into our ongoing effort to rank the search engines in order of dependability with the goal of enhancing retrieval performance. One further Microsoft study on text-based picture retrieval [1] compiles data via daily web crawling and building and updating an offline database for retrieval. This study is relevant to our work since it combines text-based image retrieval with content-based image retrieval and involves regularly traversing the information space to create a database. Eventually, we'd want to investigate this pairing.

3. SYSTEM ARCHITECTURE

3.1. Overview

Intuitively, in a general framework of a system to retrieve documents from a large, dynamic geospatial information space, we would need to select data from a geospatial information space, and then choose the most promising candidates for matching against a user's query while continuously updating the results for the user over time. Therefore, we design our I-FGM system with the following components:

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

We note that a general I-FGM framework was discussed in [9]. In this paper, we focus our discussion on I-FGM for geospatial data. Specifically, we first focus on geospatial data represented as natural language text and take advantage of the current search engines to create our database.

3.2 Components

I-Forager: This part takes the user's query as input, then retrieves "would-be" relevant data from the GIS, and finally stores it in a database known as the gIG-Soup. The first level of filtering is provided by I-Foragers using current search technologies. It is well-known that several search technologies perform better for specific queries than others. Understanding the trustworthiness of each search technology is crucial as search engine rankings provide a rough idea of how relevant a piece of information is. One way to quantify this is

by looking at the document's first-order similarity, which is measured by the dependability of the I-Forager. When the I-Foragers first prioritize a new document for download, this is known as its predicted first-order similarity. This is the total of all I-Foragers multiplied by the product of the downloaded document's scaled reliability and its I-Forager measurement, where the latter is the inverse of the former and the former is the document's rank in the search results produced by the former.

To extract useful information, we put the downloaded papers into gIG-Soup (see below). A knowledge nugget is a directed graph, which reflects the ideas and interactions between them in a geographical domain. This graph is also called a document graph. Noun phrases like "time square" and "wall street" illustrate concepts. Our knowledge nuggets include two types of relations: set-subset and related to. A knowledge nugget may have relations such as "time square" linked to "New York city" as an example. You may find additional details about this representation in the references [7, We started with five computers as I-Foragers in the implementation, as we picked five well-known search engines. Lynx-2.8.5 is used by every I-Forager node to obtain documents from the World Wide Web. A text-based browser is Lynx. I-Forager takes user queries and utilizes Lynx to get relevant document URLs from a certain search engine. The papers are thereafter downloaded by each I-Forger by means of the URLs. There is a distinct search engine that each of the I-Foragers employs. Google, Yahoo!, MSN, Teoma, and Looksmart are the five search engines that were used in our experiment. After downloading documents, gIG-Soup removes HTML tags and any other unnecessary material. Given the challenges in precisely defining an I-Forager reliability function, we simplify our analysis of our first prototype by associating equal reliability with all I-Foragers.

gIG-Soup: This component is a repository that contains documents that have been foraged and whose document graphs are under construction by the gIG-Builder. The gIG-Soup is implemented as a Network File System (NFS) directory that is shared by all the nodes in the system. A sub-component of the gIG-Soup is a database (implemented using

MySQL) that contains important information about the documents in gIG-Soup such as file name, URL, identification for a query, number of sentences parsed and so forth. It also stores the partial and full relevancy measure of the documents in the gIG-Soup.

gIG-Builder: The gIG-Builder is the process that goes through the gIG-Soup, selects the most promising documents and works on building the document graph for a period of time. The most promising documents are documents with highest priority. Priority is a parameter which tells the system how potentially relevant this document is to the given query. Priority is determined by the chosen similarity measure, the history of the similarities, the history of parsing time, and the expected first-order similarity of the document. The priority of a newly downloaded document depends primarily on the expected first-order similarity. As the document is processed by the gIG-builders, the priority is refined by the system according to the similarity measures. It is the refinement process that helps to guide the allocation of computational resources. The period of time (also called parsing time) that the gIG builder processes a document is determined by its priority and the similarity measure used.

In our implementation, 5 nodes are used as gIG-Builders for our initial setting making it equal to the number of I- Foragers in order to see how this component works with other components before planning our expansion. The gIG- Builders continuously query the gIG-Soup, specifically its MySQL database, for documents whose document graphs have to be built. The documents with the highest priority are more likely to be selected. The information about document priority and the file containing the document are obtained from the MySQL database. The gIG-builders retrieve a document from the gIG-Soup and construct its document graph for a period of time. After this process is over, the document graph is placed back into the gIG-Soup. The MySQL database is updated with information about the documents, such as the number of lines parsed. 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: Each I-Matcher generates a query graph from a user's query. Next, I-Matcher processes the nuggets in the gIG-Soup and calculates/updates their relevance measure. I-Matcher queries the MySQL database continuously for documents that have been processed by the gIG-Builder and not yet matched. The documents with highest priorities will be more likely selected by the I-Matcher and the similarity values for them will be calculated. A match between a query graph q and a document graph di is defined as

4. CONTROL SYSTEMS

Two control systems have been put in place to ensure that our I-FGM system is working properly. A somewhat intelligent system and a baseline system are these. To us, I-FGM represents the pinnacle of artificial intelligence. Both systems are quite similar to the I-FGM system in terms of its primary components, which include I-Forager, I-Matcher, gIG-Soup, gIG-Builder, and Blackboard. For control systems, however, both the individual components' functions and their interplay are distinct and significantly simplified. Furthermore, all three systems use the same data resources for the identical query, allowing for precise performance comparisons. There are two main objectives when it comes to creating control systems. Our primary goal in implementing these control systems is to enable us to compare the I-FGM system's performance to that of other systems. As a second point, the I-FGM system makes use of information regarding

piece of information to help in data retrieval. Consequently, we want to observe the impact of information nugget knowledge on retrieval by building baseline and somewhat intelligent systems. In the baseline system, we don't know anything about any information nugget, while in the somewhat intelligent system, we simply know the basics.

4.1 Baseline System

The baseline system is a simple system without any intelligence. In this system, I-Foragers download the documents for a given query and directly place them into the gIG-Soup. The gIG-Builders choose documents randomly and construct document graphs for a fixed period of time. I-Matchers randomly pick a partially or completely

parsed document graph to compare with the query graph.

4.2 Partially intelligent system

The partially intelligent system uses expected first-order similarity as the priority of the document during the whole query process. In other words, the I-Matcher will not update the document priority. The priority of the document is determined only based on the reliability of each I-Forager. Since the function for reliability will be refined as more geospatial information becomes available during the query process, this system is dynamic. Using the expected first- order similarity, it, to some extent, can dynamically and intelligently guide the finite computational resource in locating the target information. The gIG-Builders arrange the documents in the gIG-Soup according to priority and then randomly choose a document from top n documents. The parsing time is computed as the product of maximum allowed time and the priority.

4.3 I-FGM

Here, the predicted first-order similarity is computed and used to establish the initial document priority once the I-Forager uploads the document to gIG-Soup. From a set of documents ranked highest in importance, the gIG-Builders choose one at random to parse. For this implementation, we've taken a document that ranks highly in terms of priority and picked it at random from a pool of papers that ranks in the top one-third. Based on our early experiences with the somewhat intelligent system, we selected this value from among the top 25%, top 33%, and top 50%. Multiplying the maximum allowable time by the priority determines the parsing time. The priority is changed as the retrieval process advances, which is what differentiates I-FGM from somewhat intelligent systems. I-Matcher revises the relevant document's metric after comparing the query graph with the partial/complete document graph. Additionally, it revises the document entry in Blackboard. The document's priority is changed by the I-Matcher after the comparison of a partially or fully parsed document is complete.

in which case Pk stands for the document's priority at time k. You may calculate it by adding the document's priority history (Pk-1) with the present similarity values (Simk). The weight assigned to the prior similarity is shown by □ . □ 0 represents the mean anticipated similarity for the longest possible parsing duration. In our experiment, we set □ equal to 0.8 and □ 0 equal to 0.6. The results of our first trials with the baseline and AI systems informed our choice of these parameters. When determining computation priorities for the I-FGM system, we have given more weight to the past of partial similarity. Thanks to I-FGM's adaptable design, we can simply change the similarity, reliability, and priority functions.

We discover that when we compare I-FGM to a somewhat intelligent control system, the former dynamically guides the system's activity by combining the I-Forager's attribute with the document's characteristics. However, the I-Forager's traits are the only ones used by the somewhat intelligent system to direct its operations.

Each of these three systems is executed on one of twelve computers. There is 256 MB of RAM and a 1GHz Pentium III CPU in every computer, or node. There is a 100 MBPS Ethernet network connecting all of the computers. There are a total of 12 nodes in the network; 5 act as I-Foragers, 5 as gIG builders, 1 as an I-Matcher, and 1 as the server that users utilize to submit queries. In subsequent trials, this experimental arrangement of twelve machines will be expanded. Additionally, we want to enhance the RAM capacity of every node. In our future endeavors, we will use the outcomes of this assessment to determine the necessary resources and anticipated performance.

5. EVALUATION

We evaluate I-FGM by comparing it with the control systems described above. A set of five queries are given to each of the three systems and results analyzed. In order to use the recall and precision performance metrics, we have to create a test bed. The procedure for creating a test bed is given below.

5.1 Testbed

The test bed is created for evaluating the retrieval performance of the given I-FGM systems. The topic of this evaluation is "Terrorism alerts for financial institutions, military compounds in New York city, New Jersey, and Washington DC". We have a set of five queries:

1. Financial buildings in New York City
2. Tall buildings in New Jersey
3. Military compounds in Washington DC
4. Financial institutions
5. Terrorism alerts for New York City

The set of targeted documents for each query is created by a pooling approach which includes the following steps: Step 1: for each query in the query set, send the query to all available searching engines

Step 2: get top 50 results from each search engine Step 3: combine results for each query into a pool

Step 4: convert completely each document in the pool to document graph Step 5: match each completely parsed document graph to the query graph Step 6: create the list of documents with non-zero similarity measure

Step 7: consider the top 10 of these documents in this set as relevant documents

This approach to creating the testbed finds most of the relevant documents without having to assess every single document in the collection which is potentially huge in this case. Therefore, the set of targeted documents can be created with reasonable effort. We note that the set of targeted documents is highly dependent on the quality of search engines used.

5.2 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 ti (ti= 30 seconds in this experiment). We record the contents of the blackboard after each interval time ti.

- 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 control systems were carefully chosen to bring out the salient features of I-FGM. By comparing with the baseline system, we are answering the question whether the methodology and similarity measures used in I-FGM is really necessary to designing an efficient IR system. By comparing with the partially intelligent system, we seek to prove that the superior performance of I-FGM cannot be emulated by just simply using the technology found in the search engines.

5.3. Metrics

We employ the commonly used metrics from information retrieval: precision and recall [6]. Precision is the ratio between the number of retrieved relevant documents over the number of retrieved documents. Recall is the ratio between the number of retrieved relevant documents over the number of relevant documents. Since we are reporting results as content to the blackboard, which is the top 10 documents at any given time, the precision and recall will be the same for this case.

6. RESULTS AND ANALYSIS

To carry out the tests, the method outlined in Section 5.2 is followed. We gathered and examined the data. For each of the five inquiries in our scenario, we display the top ten documents' recall values (a–e) over time in Figure 1 (a–e). To emphasize the early performance disparities in the recalls for all three systems—a crucial component of quick relevant retrieval—the charts only display the recall range of 0 to 0.7 for each query. When one system reaches the recall level of 0.7 or above, we mark the data as met. On one side, we have time-slices beginning at 0, and on the other, we have the recall value, which may take on values between 0 and 1. There are two time slices spaced thirty seconds apart. When compared to the somewhat intelligent system and the baseline, the I-FGM system often has better recall for a while. However, eventually, the partially intelligent system and the baseline systems become competitive. The outcomes for inquiries 3, 4, and 5 make this quite evident. As an example, consider question 4. The baseline system failed to provide any top-tier, relevant documents for the first seven minutes, but the I-FGM managed to obtain thirty percent of the top ten in only one and a half minutes. The top ten, 30% quality relevant documents for this query are retrieved by the somewhat intelligent system in 2.5 minutes. In a similar vein, retrieving a top-ten document for query 3 takes 13 minutes for the baseline and somewhat intelligent system, but only 3 minutes for I-FGM. In response to question 5, it is evident that the I-FGM system outperforms the baseline and semi intelligent systems. In comparison to the baseline system, which takes 9 minutes to obtain a single document from the top 10, I-FGM takes half that time to retrieve the most popular papers. This was another question where the semi intelligent system failed miserably. The ability to provide funds to deserving applicants is the IFGM system's strongest point. We expect that this will allow us to get the top ten blackboard pages much more quickly than the baseline and somewhat intelligent systems. The results of questions 3, 4, and 5 corroborate this.

There is less of a recall performance gap between the baseline and I-FGM systems for queries 1 and 2. IFGM keeps rapidly achieving lower recall levels. It takes the baseline and somewhat intelligent systems the same amount of time to obtain a high-quality document from the top ten, but IFGM gets there thirty seconds faster. This is for query 1. For three and a half minutes, I-FGM outperformed the baseline system, with a recall of 0.2 compared to 0.1 for the baseline. At all other recall levels, it lagged behind the control system. In response to inquiry 2, I-FGM took 5.5 minutes to get a document from the top 10, compared to 7.5 minutes for the baseline system and 8 minutes for the moderately intelligent system. Nonetheless, over time, the baseline system outperformed I-FGM and the somewhat intelligent system in terms of recall. Both of the first two requests were underperformed by the somewhat intelligent system.

The I-FGM approach was not as effective for questions 1 and 2 as it was for queries 3, 4, and 5. This is because some of the highly rated papers by trustworthy search engines actually have very little material that is relevant to the query at hand, but they do include a lot of relevant connections. These papers were highly rated by the search engines since the engines consider the popularity of the site. The I-FGM system will allocate more time to process such a document when it is chosen. It turns out, however, that these papers really don't contain any useful information. Conversely, the baseline unintentionally sidesteps this problem by giving all documents the same amount of processing time and selecting one to execute at random. This approach to allocating resources is not intelligent, but it gets the job done under certain conditions. The fact that baseline does better is another noteworthy finding from the studies. remember as the days go by. This is due to the fact that even after rounds one through four, I-FGM will continue to allocate parsing time to documents that it has previously recognized as being around 25–40% relevant. Therefore, the allotted time for each new applicant is minimal. Background checks all papers in the search area at the same rate, so it can find the remaining relevant ones faster than I-FGM. Keeping working on a reliable function will allow us to solve this issue. All of the I-Foragers have been considered equally reliable, as we said before. One of the components used to determine dependability is the I-Forger's historical performance in relation to recall. We anticipate solving this issue with the help of additional trials, bigger testbeds, and enhanced priority and reliability mechanisms.

The time it takes for three systems to achieve a certain degree of recall is also detailed here. Low recall is represented by 0.2, medium recall by 0.5, and high recall by 0.7. Table 1 shows the time it takes for a system to attain different levels of recall. As an example, at the 9-minute point, I-FGM achieved the recall level of 0.5 for question 4. According to Table 1, the I-FGM system outperforms the baseline system in terms of early recall for queries 3, 4, and 5, but it is only effective for queries 1 at low and moderate recall points. It failed to outperform the baseline system in response to question 2. Early on in the process of question 4, the somewhat intelligent system only manages to obtain high recalls.

Table 2 shows the results of our blackboard quality assessment, which included finding the difference between the full similari-

ty score of the top 10 relevant texts and its partial similarity score within the first ten minutes of the system's runtime. We settled on this time frame since loading the top ten papers into the blackboard may be a real pain for some control systems. For each system, the average of the similarity differences is shown in Table 2. Out of the five questions we looked at, the average difference for I-FGM was the least for all except query 2. Aside from inquiry 1, the somewhat intelligent system accomplishes the second-smallest difference in four inquiries. Thus, when contrasted with the two control methods, I-FGM results in higher-quality blackboard material. To sum up, our I-FGM system outperforms the somewhat intelligent and baseline systems in terms of memory level and chalkboard quality, and it can attain greater recall levels early.

7. CONCLUSION

Here we detailed the first findings from developing, deploying, and testing the Intelligent Foraging, Gathering, and Matching (I-FGM) system for retrieving geographic data. By analyzing our brief situations, we proved that the I-FGM system could consistently outperform the control systems in retrieving high-quality documents. Essentially, we have shown the efficacy of the approach that involves progressively updating the relevance measure after partly analyzing the information in the search space. This technique allows for efficient use of computing resources and, as a consequence, fast retrieval of results. Additionally, we demonstrate that our system's retrieval performance may be significantly enhanced by using a simple heuristic for document processing prioritization.

While reviewing the I-FGM system, we came across a few problems. An example of this is the problem of determining an appropriate level of computing resource use. One computer was utilized for matching in our studies, whereas five machines were employed to generate document graphs. Our research focuses on understanding the relationship between the number of computers utilized for document graph construction and matching, the fraction of components required to avoid resource bottlenecks, and the recall outcomes for the top 10 documents. Since we want to make certain adjustments to the I-FGM components so that they can communicate with one another, these studies are crucial as well. So, in an emergency, an I-Matcher may transform into a gIG-Builder. Secondly, the priority function we are now using just considers the amount of time spent on this document and its current level of similarity. When determining processing priorities, the priority function does not consider the remaining document size. This might pave the way for more fair allocation of computing resources, which in turn could improve processing efficiency. Then, new papers will be given a greater shot at processing time allocation. In order to rank the documents, we will use a combination of machine learning methods and specific empirical metrics, including the amount of parsed sentences and how closely those phrases relate to the current question. Thirdly, the job and collection at hand determine the information retrieval method. To continue validating I-FGM and gaining new insights, we will investigate experimenting with bigger testbeds and various search areas. Finally, we want to determine the information space's dynamic nature and its impact on our retrieval performance by conducting trials at various periods and comparing the results in terms of relevant documents and retrieved documents. Crawlers may further investigate the information space in real-time, which will also aid in resolving this problem.

Finally, I-FGM, or Intelligent Foraging, Gathering and Matching, might significantly improve geographic data retrieval. The first experimental findings confirm that our I-FGM technology is effective. Improving its efficiency and accuracy may be achieved by fully using its architecture and similarity measurements.

REFERENCES

[1] The authors of "Web Mining for Web Image Retrieval" are Chen, Wenyin, Zhang, Li, and Zhang, with the last name of Zhang.

Publication Date: 2001, Volume 52, Issue 10, Pages 831-839, Journal of the American Society for Information Science.

Modeling and retrieving images based on content [2] is written by Gudivada V. N. and Raghavan V. V. Articles 427–452 published in 1997 in the journal Information Processing & Management.

Automatic Image Annotation and Retrieval using Cross-Media Relevance Models [3] Jeon J., Lavrenko V., and Manmatha R. An article published in the proceedings of the 26th annual international conference on information retrieval organized by ACM SIGIR. Chapters 119–126. 2003.

[4] Distributed Geolibraries: Spatial Information Resources, National Academy Press, Washington DC, 1999. National Research Council.

[5] Image retrieval: present methods, future directions, and unanswered questions. Y. Rui and S. T. Huang. Volume 10, Issue 1, Pages 39–62, 1999, Journal of Visual Communication and Image Representation.

The authors of "Introduction to Modern Information Retrieval" are Salaton and McGill. (1983) McGraw-Hill.

A technology for active user interface information retrieval called Kavanah was developed by Santos Jr., Nguyen, and Brown [7]. October 2001, Maebashi, Japan. Sections 412–423. The year 2001.

A Medical Information Retrieval Application's Adaptive User Modeling: An Empirical Evaluation (Santos Jr., E., Nguyen, Q., & Pukinskis, E., 2008). Page numbers 292-296 from the 9th User Modeling Conference Proceedings. In 2003

[9] Distributed Real-Time Large-Scale Information Processing and Analysis by Santos Jr., Santos, and Santos The publication is the 2004 SPIE Defense & Security Symposium Proceedings, volume 5421, pages 161–171.

The MetaCrawler Architecture for Web Resource Aggregation [10] by Selberg E. and Etzioni O.

Pages 11–14 of the 1997 January–February edition of IEEE Expert.

[11] A Geospatial Web Query System Based on Ontologies by Weigand and Zhou. The Third Annual Next Generation Geospatial Information Conference (NG2I).

Grouper: An Interactive Clustering Interface for Search Results on the Web [12] Zamir O. and Etzioni O.

Volume 31, Issues 11–16, Pages 1361–1374, 1999, Computer Networks (Amsterdam, Netherlands). 1997.
Web document clustering: a feasibility demonstration [13] by Zamir and Etzioni. This paper was presented at the 21st annual international ACM SIGIR conference on Research and Development in Information Retrieval. Chapters 46–54. In 1998.