# Cloud Computing in Trust Building Knowledge Discovery for Information Retrieval

Vishal Jain
Research Scholar,
Computer Science and Engineering Department,
Lingaya's University, Faridabad.

Mahesh Kumar Madan
Professor and HOD,
Computer Science and Engineering Department,
Lingaya's University, Faridabad,

## ABSTRACT

This paper discusses more about cloud computing in trust building knowledge discovery for informational retrieval. The paper also contains some proposed model, algorithm, and experimental results. To arrive at the conclusion of this paper statistical analysis was collected.

## KEYWORDS

Information Retrieval, Knowledge Discovery, Ontology, Cloud Computing

## 1. INTRODUCTION

Cloud computing is a term used to refer to anything which involves offering hosted services using the internet. It can be public or private. Cloud computing is becoming popular in the IT industry. Over the past few years, the supply-and-demand of this new area has been seeing a huge increase of investment in infrastructure and has been drawing broader uses in the United States.

## 2. INFORMATIONAL RETRIEVAL

Informational retrieval (IR) is concerned with the structure, analysis, organization, storage, searching, and dissemination of information. An IR system is designed to make available a given stored collection of information items to a user population desiring to obtain access. The stored information is normally assumed to consist of bibliographic items such as the books in a library or documents of many kind; by extension, an IR system may be used to access collections of drawings, films, museum artifacts, patents and so on. In each case, the IR system is designed to extract from files those items that most nearly correspond to existing user as needs as reflected in requests submitted by the user population[1].

Most operational retrieval services are implemented online using console terminal devices to introduce search queries and to obtain retrieval output. In that case, the information searches may take place interactively in such a technique that data delivered by the users during the search operation is used to obtain improved search output [2]. Furthermore, networks of information centers may be created by supplying suitable connections between individual centers, thereby affording the user population a chance to access the resource of the whole network.

When there is a large store of data like a set of customer records, it becomes expensive to search the entire store sequentially to find a particular piece of data. One would like, instead to be able to go directly to the point where the relevant data is to be found and extract it without search. A memory that allows one to do this is termed as random access. A better description for this is addressable, direct access, for there is nothing random about the way in which one approaches it. The store is to be addressable so that each record in it can be designated, or pointed to, by a symbolized address (name). It is to have direct access so that the information processer can be switched to read the desired record directly, once its name is known, without requiring a search [6] (Rouse, 2010).

## 3. RETRIEVAL OPERATIONS

In many conventional retrieval situations, a search request is constructed by choosing appropriate keywords and content terms and appropriately interconnecting them by Boolean connections ( and, or, not) to express the intent of the requestor. For example, a request covering "tissue culture studies of human breast cancer" may then be transformed into the statement shown below;

*{Breast neoplasm or carcinoma, ductal} and {human or not (any term indicating animal or disease)} and {tissue culture or culture media or chick embryo} and English*

### 3.1 Retrieval application

The most common type of retrieval situations is exemplified by a reference retrieval system performing 'on demand' searches submitted by a given user population. Normally, only the bibliographic information is stored for each item, including authors' names, titles, journals or places of publication, dates, and applicable keywords are usable for search purposes. Sometimes, the words of the documents titles can also be searched. Less commonly, more extended text portions such as abstracts, summaries, or even full texts may be stored, in which case a text search (as opposed to a simple keyword search) becomes possible.

In any case, the responses provided by the system consist of references to the bibliographic items that match the user queries. In most conventional situations, the retrieved information is submitted to the users in no particular order of importance. An ordering indecreasing query-document similarity can however be obtained in the more advanced systems, which can then be used advantageously for search negotiation and feedback purposes.

### 3.2 Algorithm

An algorithm is the precise characterization of a method of solving the problem, presented in a language understandable to the device. In certain, an algorithm has the following properties.

1. Application of the algorithm to a specific input set or problem description outcomes in a finite order of actions.
2. The order of actions has anexceptional initial action.
3. Every action in the system has an exceptional initial action.
4. The system ends with either an answer to the problem, or a report that the problem is unresolved for that set of data.

This concept can be illustrated with an example. Find the square root of the real number x. As it is stated, this problem is algorithmically either trivial or unsolvable, owing to the irrationality of most square roots. If one accepts $\sqrt{2}$ as the square root of 2, for example, the solution is trivial. The answer is the square root sign (√) concatenated with the input. In Symbol, the entire algorithm is

**OUTPUT END = $\sqrt{\phantom{x}}$ INPUT**

However, if we want a decimal expression, then the square root of 2 can never be calculated exactly. Hence, the requirement of a finite number of actions is violated.

### *3.2.1 Quality Judgments of Algorithms*

Whichever computer program is a semi-algorithm, and any problem that always halts is an algorithm (of course, it may not solve the problem for which the programmer intended it). Given a solvable problem, there are many algorithms (programs) to solve it, not all of equal equality. The primary practical criteria by which the quality of an algorithm is judged are time and memory requirements, accuracy of solutions, and generality. To cite an extreme example, since a properly defined game of chess comprises of a finite number of possible moves, there exists an algorithm to determine the "perfect" chess game. Simply examine all possible move sequences, in some specified order. Unfortunately, the time required to execute any algorithm based on the idea is measured in billions of years, even at today's computer speeds. The memory requirements for such an algorithm are similarly overbearing [4].

The accuracy of algorithm is a characteristic often more closely related to time than to memory requirements. For instance, the square root algorithm presented is not very accurate. Changing the test constant from 0.00005 to 0.00000000005 will produce 0.00000381 as the square root of zero at the cost of more iteration through the circlet of the algorithm. No further memory is needed, and the further iterations require only a small fraction of a second. Further improvement may be obtained from the corresponding algorithm in double-precision at a cost of both run time and additional memory space. In each case the basic algorithmic concept is unchanged [5].

## 4. DYNAMIC PROGRAMMING

Dynamic programming arises when the only algorithm one thinks of is enumerating all possible configurations of the given data and analyzing every one of them to see if it is a solution. An essential idea is to keep a table that contains all previously computed configurations and their outcomes. If the total number of the configurations is large, the dynamic programming algorithm will necessitate substantial time and space. However, if there are merely small amount of distinct configurations, dynamic programming avoids recomputing the solution to these problems over and over [3].

To determine if there are only a small number of distinct configurations, one needs to detect when the so-called principle of optimality holds. This principle asserts that every decision that contributes to the final solution must be optimal with respect to the initial state. When this principle holds, dynamic programming drastically reduces the amount of computation by avoiding or evading the listing of some decision systems that cannot perhaps be optimal.

As simple consider computing the n-th Fibonacci number, $F_n$ where $F_n = F_{n-1} + F_{n-2}$ and $F_0=F_1=1$. The first few elements of this famous sequence are 1, 2, 3, 5, 8, 13, 21, 34,…The obvious recursive algorithm for computing $F_n$ suffers from the fact that many values of $F_i$ are computed over and over again. However, if one follows the dynamic programming strategy and create a table that contains all values of $F_i$ as they are computed, a linear time algorithm results [5].

## 5. CONCLUSION

In conclusion, the most exact answer to this topic is based on the model theory for FOPC, (First Order Predicate Calculus). This defines the structure of a possible world and conditions under which an expression would be true in it, and takes the meaning of a logical expression to be the constraint on this structure imposed by insisting on its truth. It is used as a tool in the analysis of more complex phenomena [4].

## 6. REFERENCES

[1] Christauskas, C., &Miseviciene, R. (2012). Cloud -- Computing Based Accounting for Small to Medium Sized Business. *Engineering Economics*, *23*(1), 14-21.

[2] Han, Y. (2011). Cloud Computing: Case Studies and Total Costs of Ownership.*Information Technology & Libraries*, *30*(4), 198-206.

[3] Jia, Z., Xue, S., Zhang, D., & Li, Q. (2010). Study of Improvement on Programming Method from Cloud

[4] Computing to Grid Computing. *Proceedings of The International Symposium on Electronic Commerce & Security Workshops*, 244-248.

[5] Kumar, P., & Gupta, S. (2011). Abstract Model of Fault Tolerance Algorithm in Cloud Computing Communication Networks.*International Journal on Computer Science & Engineering*, *3*(9), 3283-3290.

[6] Lizheng, G., Shuguang, Z., Shigen, S., & Changyuan, J. (2012). Task Schedule Optimization in Cloud Computing Basing on Heuristic Algorithm.*Journal of Networks*, *7*(3), 547.

[7] Rouse, M. (2010). Cloud computing.retrieved from,http://searchcloudcomputing.techtarget.com.