The "self-learning query" algorithm for searching scientific publications across specialized search engines

Nikolay V. Kovalev[a], Dr. Rosanna A. Esquivel[b]

[a] Vice President,Developer Soft and Technical Support Sea Inc., ,Angeles City, Philippines
[b]Assistant Dean, College of Computer Studies,Angeles University Foundation, Angeles City, Philippines
E-mail: [a]nik.nv19@gmail.com,[b]esquivel.rosanna@auf.edu.ph

Article History: Received: 10 November 2020; Revised 12 January 2021 Accepted: 27 January 2021; Published online: 5 April 2021

Abstract: Typically, a search operation on Big Data is complicated task due to different formats and different nature of the data. To simplify it the user may implement special algorithms to process different data for uncertain criteria. The "self-learning query" algorithm presented in this work allows to search in Big Data either for certain or uncertain search criteria's with minimum attention from the data programmer. It uses accumulated search statistics as the basis to make the result set more precisely according to the search criteria, so as long the user work with the system as more precise will be the results. The algorithm presented in this work allows data scientists to search for uncertain data and potentially discover results faster by offloading the burdens of data management and provenance to the expert system.

Keywords: Big Data, Search algorithm, Knowledge base, Data Mining, Logic programming, Machine learning

1. Introduction

Generally speaking, Data Mining is the process of discovering patterns in a large data sets and/or Big Data. It involves methods at the intersection of machine learning, statistics, and database systems [1], [2]. The most interesting area in the Big Data analysis is Machine Learning. It can be combined also with statistical analysis to provide the user more comprehensive results. The Machine Learning includes the next main techniques [3]:

- Supervised learning or Classification [4],
- Clustering [5],
- Outliner/anomaly detection [6],
- Filtering/parsing.

All of them can be used for search operations in Big Data especially when the search conditions are uncertain [7]. As for the search algorithm the Classification and Clustering techniques are very useful because at first the algorithm should know if the data element is somehow (approximately) can satisfy our uncertain search conditions [8]. These techniques also can be used for search together with statistics gathering and partial statistical analysis.

Since the area of this research is the search for scientific publications, it's just necessary to discuss about the existing search engines designed for these purposes. There are many search engines can allow searching for particular scientific papers, but only specific engines support searching for paper specific criteria like relation the paper to some research area, and for example affiliation the author to a specific institution.

Despite the features of the different search engines there is no one, universal world databank and typical researcher should check several engines to find papers satisfying his search criteria.

There are some popular search engines to search for scientific papers are shown and analyzed in the Table 1.

Table 1. The special search engines

Search engine	Advantage	Disadvantage
ACM Digital Library (dl.acm.org)	• Relatively free access to content (search list and item info) • Cheap access to the body of publication • Well-customized search options • Big corpus of publications	• One-time search, passive search • Non-manageable system • The service is still paid (body of publication) • Simple, list-like, non-customized report

	from different sources • Support search bots (URL substitution)	
Scopus/ Elsevier (scopus.com; elsevier.com)	• Custom reports including graphical • Well-customized search options • Convenient interface • Free access to content (in the most cases)	• One-time search, passive search, non-manageable system • Corpus limited only for Elsevier publications
Google Scholar (scholar.google.com)	• Huge corpus of publications from different sources • Free access to content (in the most cases) • Quick access to the content	• Simple, list-like, non-customized report • Various limitations in search criterias • One-time search, passive search, non-manageable system
Research Gate (researchgate.net)	• Free access to content (in the most cases)	• Simple, list-like, non-customized report • Various limitations in search criterias • Limited search base/corpus • One-time search, passive search, non-manageable system

Even the discussed search engines may satisfy the majority of its users, anyway it may have the next weakness and disadvantages:

1. Various limitations in the search criteria

Generally speaking, there is no custom search; the user can only follow the features, which the service offers to use. The search is always based on the key words. The user can not browse the corpus interactively, choosing the search domains like Philippines -> AUF -> Nikolay Kovalev, which is more preferable and effective technique.

2. Limited search base/corpus

Some services like Scopus/Elsevier can do search only in its own index of journals. If the searching article is not published in the subscribed journal it will never be shown in the search results. The solution here is to selectively combine in one report the search results from different sources, including those which are not indexed even by Google (for example local library).

3. Paid services and features

Some of the services provided by those engines are fully paid, or some features in those services are paid. The research department of A.U.F. and also other institutes may not have enough funds to pay for that.

4. One-time search, passive search, non-manageable systems

The search engines are using the principle of query-response. Even it’s good in terms of performance, for some users it’s not enough. The passive search technology, when the service only responds to the user's requests is not always effective. Furthermore, the most of the search services are cloud/internet solutions managed by big companies like Google, and it’s very hard to ask them to change something in their services.

The proposed solution to resolve this disadvantage is "active search", when the knowledge database itself is searching the web until the exact result will be found and/or on constant base reports on partial results. It’s really necessary for example, in cases when the user should monitor some frequently changed data, like news articles, stock market data and so on. One a solution for this issue can be a special data language to store the search results temporally in the local knowledge database making it persistent. The persistent search results allow the cumulative search techniques, i.e. the user can search only for new data comparing it with the stored, older results.

5. Simple, list-like, non-customized reports

The report a user can see in the search engines are typically a list of the articles found for the used search criteria. The user should scroll the list and find the desired article. There is no other navigation. I attached an example of the proposed interface of how the report could look like. If to make each element/article of such graph clickable, the information will be shown about the particular article, and the new references/links will appear and so on. At the same time the user should be allowed to choose the type of the report, it can be either list, interactive tables, graph/hierarchical trees or even 3D structures. Together with it, the general statistic should be also shown, like range diagrams for published articles by year/author/country and so on.

Anyway, despite the advantages of particular engines, there is no real global search engine, which is able to collect data not only from multiple sources, but also can apply different conditions to each of the data source. Needed to mention here also the accessibility of each engines. For users with slow and not constant internet connection may not be convenient to check multiple search engines and download full papers in order to search for specific criteria. The algorithm, which is proposed and conceptualized in this research, can also resolve almost all of the listed disadvantages.

2.Theoretical Background

At first, it's necessary to define the goal of search and articulate the terms. Given the "source data", which represent all possible Big Data what the search algorithm has access [9]. Typically, we can designate it as a set. In this set there many different data elements like article title, author, publisher, year and etc. These data elements can be designated as a_n, where n>0. Then such a_n, which satisfying our search goals can be grouped in some "desired groups" and *m* will be the total number of them.

Definition 1.*The desired group G_k, where k=1,..m is the search goal sample and a set of associated with it data elements a_n, which satisfied this search goal.*

These groups consist of the data elements same to a_n but already combined for specific purpose like all possible variants for the name of Angeles University Foundation. We can consider here such options as "A. U. F.", "AUF" and so on. However, the proposed search algorithm may not know, if which data element it puts in the desired group is correct. The goal of the algorithm here is to find connections between data elements in the desired groups and those in the source data [9].

Let's introduce here the special term of "affinity power", to numerically count such connection and to achieve the algorithm goal by reaching its specific rate. The math sense of such term is shown on the Figure 1, and described in the next definition:

Definition 2.*The affinity power $A(a_n, G_k)$, where n>0, k=1,..m between the data element a_n and the desired group G_k is an integer value showing the rate of how often a_n and G_k are appearing together in the source data.*

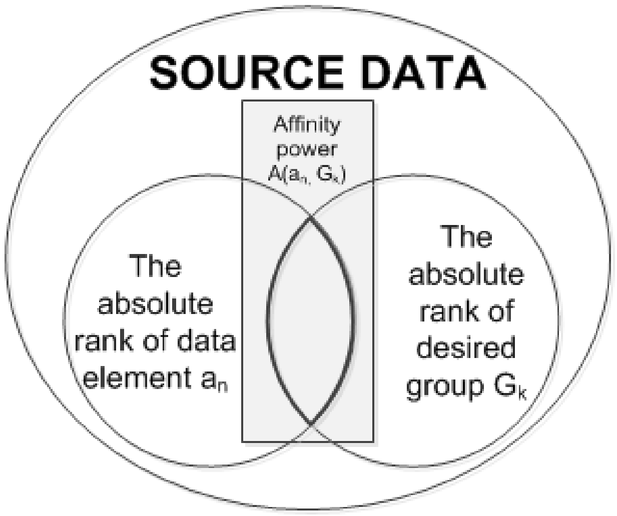
SOURCE DATA
Affinity power A(a_n, G_k)
The absolute rank of data element a_n
The absolute rank of desired group G_k

Figure 1. Venn diagram of the affinity power $A(a_n, G_k)$

The "absolute rank" on this figure is an integer value and it represents of how often the data element a_n is appearing in its own set. For example, in terms of publications we can find 30000 papers for Physics and only 200 for Kip Thorne, so 30000 and 200 are the absolute ranks here for the data elements "Physics" and "Kip Thorne" respectively. The same absolute rank can be counted for an element in the desired group G_k.

Then it's possible to calculate the Affinity power $A(a_n, G_k)$ based on the statistic gathered. So if to continue the presented example, let the "Physics" be one of our desired group, then the Affinity power A("Physics", "Kip Thorne") will be something like 198, because Kip Thorne is a famous physicist. If the algorithm can calculate how often the searching value is appearing together with the testing element, it will be already a ground to make some further conclusions, like assuming if the most of publications for Kip Thorne have fallen in the "Physics" desired group he is physicist.

To collect the particular statistics is crucial to count the described parameters. The collected statistics may be used for different purposes but mainly to adjust the search conditions for the next iteration. The statistics can serve also as the supervisor in any of the Machine Learning techniques [10], [11].

However, Machine Learning and Statistics are closely related fields in terms of methods, but distinct in their principal goal: "statistics draws population inferences from a sample, while Machine Learning finds generalizable predictive patterns" [11].

The proposed search algorithm can use statistics from both directions: gathered from the source data to determine of how often the data elements are used together, and also it can use stored statistics which is already collected before. Such statistics can be stored in the knowledge database and it represents how often the algorithm itself chosen the particular data elements. Especially it requires when we work with several or many desired groups and the search operation is continuous, so the stored results are used in further processing. The Figure 2 shows the algorithm' conceptual schema in details by example of searching for specific colors:

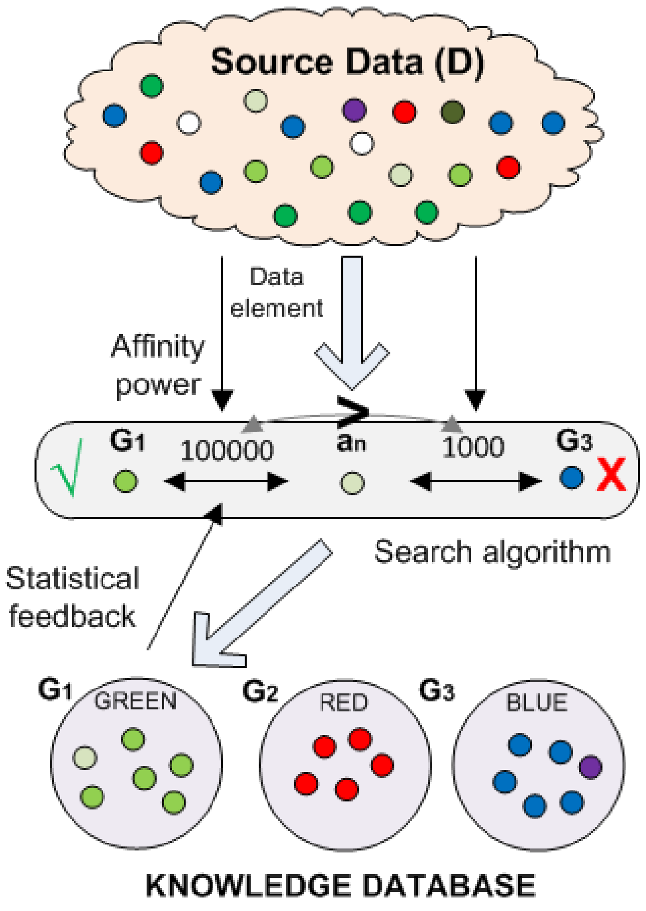

Figure 2. The concept diagram of the adaptive classification algorithm

Initially, the algorithm does not know what colors are "green", "red" and "blue". The goal for the researcher here is to teach the algorithm how to put the checked data elements into the proper basket or group.

Definition 3.*The desired group G_k, $k=1,..m$ itself and any data element a_n may have the absolute rank in the source data, which is an integer value showing the rate of how often it's appeared.*

So, each time the we take a data element from the data source its necessary to check how often this element appears together with each of the desired group.

The desired algorithm can check also statistic of its own work, i.e., "statistical feedback", which will help to narrow the decision and can be used in moments when statistics from the data source is temporarily unavailable.

So, when the affinity power between the checked element and the desired group is higher than between the checked element and other groups this time the algorithm should decide to put or set affiliation on the checked element to the desired group. To prevent choosing the outliner, the affinity power for the checked data element is supposed to be greater or equal than minimal affinity power for any the desired groups. The affinity power is not an absolute value; it's meaningful only when we compare two or more values together. The collected statistical feedback from the knowledge database is also important in the calculation as a percentage of the checked elements inside the desired group.

Definition 4.*The decision affinity $Ad(a_n, G_k)$ of the desired group G_k, where k=1,..m is percentage of testing data element a_n found inside G_k to the total number of the data elements in G_k.*

So, if D is the data source, and $a_n \in D$, the next statement is true:

$$a_n \in G_k \Leftrightarrow \begin{cases} A(a_n, G_k) = \max\left(A(a_n, G_1), .. A(a_n, G_m)\right) \\ A(a_n, G_k) \geq \min(A(a_1, G_k), .. A(a_n, G_k)) \\ Ad(a_n, G_k) > 0 \\ n > 1 \ \forall G_k \end{cases} \quad (1)$$

To increase effectiveness of the proposed algorithm at the beginning of its work, the second condition can be used separately: $A(a_n, G_k) \geq \min(A(a_1, G_k), .. A(a_n, G_k))$ (2), which can be also rewritten as $A(a_n, G_k) \geq \min\left(A(a_1, G_k) * \left(1 - \frac{P}{100}\right), .. A(a_n, G_k) * \left(1 - \frac{P}{100}\right)\right)$ with percentage factor P. The value P is recommended to be in the next range: $10 \leq P \leq 20$. The last condition means that some correct data elements are supposed to be in the G_k already to let the algorithm work properly i.e. there some initialization is required. The statistics about each choice the algorithm has made should be stored as well.

3.The "Self-Learning Query" Algorithm

This algorithm is named as self-learning due to implementation of Machine Learning techniques and the special ability to change its own code i.e., "mutate" during its execution. The word "query" is used because the algorithms are actually querying the source data, catching the data elements and choose to store it in the local knowledge database for further reports [12].

The "self-learning query" algorithm is based on the conceptual data search algorithm described above. It is a statistical driven algorithm. Theoretically, the algorithm can work with any search engine, but of course it requires to be properly configured to access the features of the particular engine. There are also no limitations for parallelization and work with multiple search engines at the same time. As for practical implementation it will require additional API support from the search engines for certain initial operations like source data parsing to distinguish the particular data elements. So, it's strongly suggested to implement this algorithm by using a special data language together with a knowledge database to store statistics and partial results [13].

The main task of this algorithm is to search for relations or affiliations, we talk here about the classification of gathering data elements.

The algorithm is designed for continuous cumulative classification of searching patterns in big data for uncertain conditions, so the user can define at least some common or template search conditions in order this algorithm can work properly. The clustering function is also achieved by setting the desired groups G_k. The groups can be set manually by the user or automatically by initial parameters just to minimize the amount of the source data for optimal performance of the main algorithm.

The self-learning query algorithm can be described in the next simplified manner:

Algorithm 1: "Self-learning query".

Input: Source data from search engines (D)

Output: Desired groups G_k with distributed a_n

```
1: set initial uncertain search conditions for search engine
2: do
3:   foreach search engine do
4:      download raw results web page code from the search engine according the search conditions
5:   parse the web page code and extract $a_n$
        until all web pages with search results are downloaded
6:      Put initial elements into the desired groups $G_k$ according to (2)
7:      for each $a_n$, n>0; $G_k$, k=1,..m do
```

```
8:  check A($a_n$, $G_k$) gathered from search engines
9:  **if** $a_n$ is qualified for $G_k$ according to (1) **then**
10:   **if** stored affinity power for $a_n$ is satisfying (2) **then**
11:        associate $a_n$ to $G_k$
12:        append stored statistic
13:        update search criteria
14:        update $G_k$
         **until** all qualified $a_n$ are distributed to desired groups $G_k$
15:  update the conditions for the search engines
**until** the user interrupts the algorithm
```

The algorithm actually starts by downloading the results web pages for certain URL with encoded initial (or preset) conditions, then the desired groups G_k are set (step 6) i.e., steps from 1 to 6 needed to gather the source data and set the desired groups. Properly setting the initial conditions for the search engines will limit size of the downloadable source data set and thus speed up the overall process; it increases performance of the algorithm and helps to cover the connection interruptions. The algorithm downloads the search results page as its source code. The HTML/XML/JavaScript source code of the downloaded page definitely requires proper initial parsing in order to extract data elements into the local knowledge database, which called here as a set of source data. Then the actual processing occurs in the steps 7 to 14. During the main phase of the algorithm, each extracted a_n is examined and in case it's qualified its associated with some of G_k. Then in the steps 12-13 the algorithm' choice is recorded, and the search conditions are updated, so the next iteration will be more precise, i.e. the algorithm actually gather its own experience and learns how to deal with specific data. Since the algorithm works with scientific publications, which means processing a symbolic data, the main operation there will be string comparison by using the "Soundex" function [14]. Since new publications are constantly appear its suggested constantly run the algorithm in background mode as well, so the user may always have the most recent results.

4.Conclusions

In this paper it was presented the conceptual schema and description of the "self-learning query" algorithm to search for scientific publications among the special search engines for uncertain criteria. The algorithm uses the Classification and Clustering techniques of Machine Learning and gathered statistic to improve results with each iteration, so it provides adaptive search with persisting search results. The presented algorithm provides the researches a solution for the critical part of the searching process.

The next suggestions are valuable for users and programmers: 1) always do the initial set up of the desired groups G_k to increase the effectiveness of the algorithm; 2) the values for the desired groups G_k can be uncertain since the algorithm is just using statistics to find similarities between the checked data element a_n and an element of the desired group. This provides an advantage and disadvantage at the same time, since the algorithm can deliver some unexpected results. So, it's up to the user if what he desires to find; 3) it's better if the user can set for groups G_k the values at least similar to what he desires to find; 4) the algorithm requires an automated access to some external statistics from a global search engine (perhaps Google) to deliver precise results. However, the implementation of this algorithm by using some particular programming language and gather the results of its real work is the topic for another research.

This paper will help the researchers who willing to work in the area of specialized search engines to improve the search systems they use and provide more comprehensive results in a comfortable way for its users.

References

Zaki J. Mohammed and Wagner Meira Jr. Data Mining and Analysis: Fundamental Concepts and Algorithms. 2013.

Charu C. Aggarwal. Data Mining. Springer. 2015.

Erl Thomas, Khattak Wajid and Buhler Paul. Big Data Fundamentals. Prentice Hall. 2015

Alpaydin, Ethem. Introduction to Machine Learning. MIT Press. p. 9. 2010.

Everitt, Brian. Cluster analysis. Chichester, West Sussex, U.K: Wiley. 2011.

Zimek, Arthur; Filzmoser. There and back again: Outlier detection between statistical reasoning and data mining algorithms. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 8 (6). 2018.

Aggarwal, Charu C. (Ed.). Managing and Mining Uncertain Data. Springer. 2009.

Shan Suthaharan. Machine Learning Models and Algorithms for Big Data Classification Thinking with Examples for Effective Learning. Springer. 2016.

Mikhail Gilula. Structured Search for Big Data from Keywords to Key-objects. Morgan Kaufmann. 2015.

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An introduction to Statistical Learning. Springer. p. vii. 2013.

Bzdok, Danilo; Altman, Naomi; Krzywinski, Martin. Statistics versus Machine Learning. Nature Methods. 15 (4): p. 233–234. 2018.

Andrii Gakhov. Probabilistic Data Structures and Algorithms for Big Data Applications. Books on Demand. 2019.

T.J. Green, S.S.Huang, B.T. Loo, and W. Zhou. Datalog and recursive query processing. Foundations and Trends in Databases, 5(2):105–195. 2013.

Knuth, Donald E. The Art of Computer Programming: Volume 3, Sorting and Searching. Addison-Wesley. pp. 391–92. 1973