Journal of Basic and Applied Engineering Research
Print ISSN: 2350-0077; Online ISSN: 2350-0255; Volume 1, Number 7; October, 2014 pp. 25-29

http://www.krishisanskriti.org/jbaer.html

Mining of Big Data – A Survey of Current State of the Art

Nimrita Koul[1], Vikas Veshishth[2]

[1]*Jamia Hamdard, New Delhi*
[2]*IIT Roorke, U.P*

Abstract: We all are a witness to an enormous increase in use of the Internet through desktops, laptops, mobile phones and other interfaces worldwide over the last decade. People from all walks of life, of all ages, of all occupations use the web for entertainment, communication, work or education. We put huge data online and we access this data. Social networking sites like facebook.com, twitter.com etc which are a chosen platform for billions of people to communicate with friends and family by sharing photos, videos and text produce huge amount of unstructured data. Huge amount of data available online contains a wealth of useful information if only we can discover it efficiently. Data mining or knowledge discovery aims at the discovery of useful knowledge from huge amounts of data.

The enormous amount of unstructured data available online data can't be handled by conventional methods of data storage and manipulation. It needs specialized algorithms and technologies for storage and manipulation. This paper is a survey report of existing techniques for mining big data for useful knowledge. We have studied mining algorithms for big data and presented their relative performance.

1. Data Mining

The importance of data to business decisions, strategy and behavior has proven unparalleled in recent years. Predictive analytics, data mining and machine learning are tools giving us new methods for analyzing massive data sets.

Companies place true value on individuals who understand and manipulate large data sets to provide informative outcomes.

Pivotal issues pertaining to mining massive data sets will range from how to deal with huge document databases and infinite streams of data to mining[1] large social networks and web graphs.

2. Big Data

Big data[2] refers to data sets that are too large and complex to manipulate or interrogate with standard methods or tools i.e. database and software technologies. It includes both structured as well as unstructured data. The data is too big and moves too fast or it exceeds current processing capacity. This data is very valuable as it has the potential to help companies improve operations and make faster, more intelligent decisions.

An example of big data might be petabytes (1, 024terabytes) orexabytes (1, 024 petabytes) of data consisting of billions to trillions of records of millions of people—all from different sources (e.g. Web, sales, customer contact center, social media, mobile data and so on). The data is typically loosely structured data that is often incomplete and inaccessible.

Sources for Big Data

Transaction data is the main source of big data. In an RDBMS used for transactions, every time you update/delete/insert and select (queries), this information is logged. The business user can analyze this data for business purposes. This Log data is also an important source of big data.

Sensor data[3] obtained from RFID detectors. Sensors are one of the biggest contributors of Big Data, enabling new applications across industries.

Examples include

- Telemetrics for auto insurance and vehicle monitoring & service
- Smart metering for energy & utilities organizations
- Inventory management and asset tracking in the retail and manufacturing sectors
- Fleet management in logistics and transportation organizations

Applications that rely on sensor-generated data have unique Big Data requirements to efficiently collect, store, and analyze the data to take advantage of its value.

Social Media like Facebook and Twitter is also a major contributor to bigdata.

The clinical data, surveillance data, mobile cameras, etc. are actually around 70-80% unstructured data.

Web log and database log data are major producers of machine generated data.

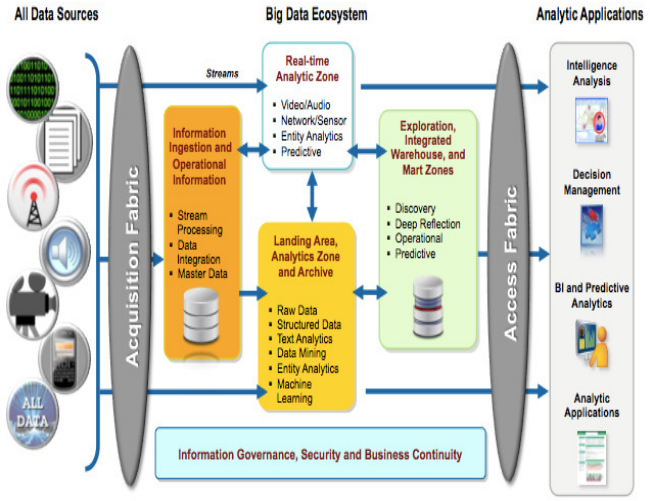

Image Courtesy: Google Images

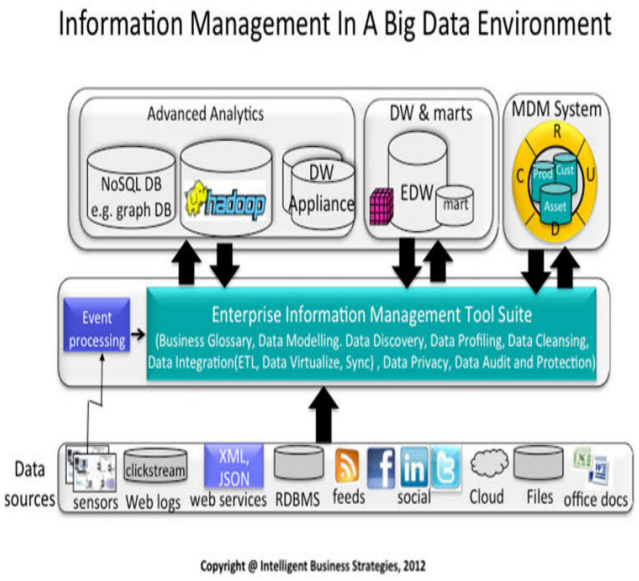

Image Courtesy: Google Images

Components of Big data

1. Data Warehouses

Data warehouses [4]are part of a big data platform. They deliver deep insight with advanced in-database analytics & operational analytics. Data warehouses provide online analytic processing (or OLAP).

2. Accelerators

Accelerators are software libraries[4] that can be used to jumpstart the development of applications by providing toolkits that will have the operators, functions, and connectors needed for specific types of data or application areas. Examples are:

- Analytic Accelerators — text analytics, geospatial, time-series, data mining
- Application Accelerators — financial services, machine data, social data, Telco event data
- Industry Models — comprehensive data models based on deep expertise and industry best

3. Hadoop

Apache Hadoop[9] is an open-source software framework for storage and large-scale processing of datasets on clusters of commodity hardware. Hadoop is an Apache top-level project being built and used by a global community of contributors and users. The Apache Hadoop framework [10]is composed of the following modules:

- *Hadoop Common* – contains libraries and utilities needed by other Hadoop modules.
- *Hadoop Distributed File System (HDFS)[9]* – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster.
- *Hadoop YARN* – a resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications.
- *Hadoop MapReduce* – a programming model for large scale data processing.

All the modules in Hadoop[10] are designed with a fundamental assumption that hardware failures (of individual machines or racks of machines) are common and thus should be automatically handled in software by the framework. Apache Hadoop's MapReduce and HDFS components originally derived respectively from Google's MapReduce and Google File System (GFS) papers.

Beyond HDFS, YARN and MapReduce, the entire Apache Hadoop "platform" is now commonly considered to consist of a number of related projects as well – Apache Pig, Apache Hive, Apache HBase, Apache Spark, and others.

Hadoop can handle all types of data from disparate systems: structured, unstructured, log files, pictures, audio files,

communications records, email – just about anything you can think of, regardless of its native format. Even when different types of data have been stored in unrelated systems, you can dump it all into your Hadoop cluster with no prior need for a schema. In other words, you don't need to know how you intend to query your data before you store it.

Hadoop makes all data useable hence lets us see relationships that were hidden before and information from complete data not just samples of data.

Big Data Analytics

Big data analytics[5] refers to the process of collecting, organizing and analyzing large sets of data to discover patterns and other useful information. Not only will big data analytics help you to understand the information contained within the data, but it will also help identify the data that is most important to the business and future business decisions. Big data analysts basically want the *knowledge* that comes from analyzing the data. Enterprises are increasingly looking to find actionable insights into their data. Many big data projects originate from the need to answer specific business questions. With the right big data analytics platforms in place, an enterprise can boost sales, increase efficiency, and improve operations, customer service and risk management.

5. ALGORITHMS FOR MINING BIG DATA SETS

The K-Means algorithm

The k-means[11] algorithm is a simple iterative method to partition a given dataset into a user specified number of clusters, *k*. This algorithm has been discovered by several researchers across different disciplines, most notably Lloyd (1957, 1982) Forgey (1965), Friedman and Rubin (1967), and McQueen (1967).

The algorithm operates on a set of *d*-dimensional vectors, $D = \{\mathbf{xi} \mid i = 1, \ldots, N\}$, where $\mathbf{xi} \in R^d$ denotes the *i*th data point. The algorithm is initialized by picking *k* points in R^d as the initial *k* cluster representatives or "centroids". Techniques for selecting these initial seeds include sampling at random from the dataset, setting them as the solution of clustering a small subset of the data or perturbing the global mean of the data *k* times. Then the algorithm iterates between two steps till convergence:
Step 1: Data Assignment. Each data point is assigned to its *closest* centroid, with ties broken arbitrarily. This results in a partitioning of the data.

Step 2: Relocation of "means". Each cluster representative is relocated to the center (mean) of all data points assigned to it. If the data points come with a probability measure (weights), then the relocation is to the expectations (weighted mean) of the data partitions.

The algorithm converges when the assignments (and hence the **cj** values) no longer change.

Each iteration needs $N \times k$ comparisons, which determines the time complexity of one iteration. The number of iterations required for convergence varies and may depend on *N*, but as a first cut, this algorithm can be considered linear in the dataset size. One issue to resolve is how to quantify "closest" in the assignment step. The default measure of closeness is the Euclidean distance, in which case one can readily show that the non-negative cost function,

$$\sum_{i=1}^{N} (\operatorname{argmin}_j \|\mathbf{xi} - \mathbf{cj}\|^2_2)$$

will decrease whenever there is a change in the assignment or the relocation steps, and hence convergence is guaranteed in a finite number of iterations. The greedy-descent nature of k-means on a non-convex cost also implies that the convergence is only to a local optimum, and indeed the algorithm is typically quite sensitive to the initial centroid locations.

Support Vector Machines

It offers one of the most robust and accurate methods among all well known algorithms. It needs only a dozen examples for training and is insensitive to the number of dimensions. In a two class learning task, the aim of SVM [11]is to find the best classification function to distinguish between members of two classes in the training data. The metric for the concept of the "best" classification function can be realized geometrically. For a linearly separable dataset, a linear classification function corresponds to a separating hyperplane *f (x)* that passes through the middle of the two classes, separating the two. Once this function is determined, new data instance *xn* can be classified by simply testing the sign of the function *f (xn)*; *xn* belongs to the positive class if $f(xn) > 0$.

Because there are many such linear hyperplanes, what SVM additionally guarantee is that the best such function is found by maximizing the margin between the two classes. Intuitively, the margin is defined as the amount of space, or separation between the two classes as defined by the hyperplane. Geometrically, the margin corresponds to the shortest distance between the closest data points to a point on the hyperplane. Having this geometric definition allows us to explore how to maximize the margin, so that even though there are an infinite number of hyperplanes, only a few qualify as the solution to SVM. The reason why SVM insists on finding the maximum margin hyperplanes is that it offers the best generalization ability. It allows not only the best classification performance (e.g., accuracy) on the training data, but also leaves much room for the correct classification of the future data. To ensure that the maximum margin hyperplanes are actually found, an SVM classifier attempts to maximize the following function with respect to **_w** and *b*:

$$L_P = \frac{1}{2} \| \vec{w} \| - \sum_{i=1}^{t} \alpha_i y_i (\vec{w} \cdot \vec{x}_i + b) + \sum_{i=1}^{t} \alpha_i$$

where t is the number of training examples, and αi, $i = 1, \ldots, t$, are non-negative numbers such that the derivatives of $L\ P$ with respect to αi are zero. αi are the Lagrange multipliers and $L\ P$ is called the Lagrangian. In this equation, the vectors _**w** and constant b define the hyperplane.

The Apriori Algorithm

It is characterized as a level-wise complete search algorithm[11] using anti-monotonicity of itemsets, "if an itemset is not frequent, any of its superset is never frequent". By convention, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order. Let the set of frequent itemsets of size k be Fk and their candidates be Ck . Apriori first scans the database and searches for frequent itemsets of size 1 by accumulating the count for each item and collecting those that satisfy the minimum support requirement. It then iterates on the following three steps and extracts all the frequent itemsets

1. Generate Ck+1, candidates of frequent itemsets of size k +1, from the frequent itemsets of size k.
2. Scan the database and calculate the support of each candidate of frequent itemsets.
3. Add those itemsets that satisfies the minimum support requirement to Fk+1.

Algorithm 1 *Apriori*

F_1 =(Frequent itemsets of cardinality 1);
for($k = 1; F_k \neq \phi; k++$) do begin
  C_{k+1} = apriori-gen(F_k); //New candidates
  for all transactions $t \in$ Database do begin
    C_t = subset(C_{k+1}, t); //Candidates contained in t
    for all candidate $c \in C_t$ do
      c.count++;
    end
    $F_{k+1} = \{C \in C_{k+1} \mid c.count \geq$ minimum support $\}$
  end
end
Answer $= \cup_k F_k$;

Function apriori-gen in line 3 generates Ck+1 from Fk in the following two step process:

1. Join step: Generate RK+1, the initial candidates of frequent itemsets of size k + 1 by taking the union of the two frequent itemsets of size k, Pk and Qk that have the first k−1 elements in common.

   RK+1 = $Pk \cup Qk$ = {*i teml, . ., i temk*−1*, i temk, i temk_* }
   Pk = {*i teml, i tem*2*, . . ., i temk*−1*, i temk* }
   Qk = {*i teml, i tem*2*, . . ., i temk*−1*, i temk_* }
   where, *i teml* < *i tem*2 < · · · < *i temk* < *i temk_* .
2. Prune step: Check if all the itemsets of size k in Rk+1 are frequent and generate Ck+1 by removing those that do not pass this requirement from Rk+1. This is because any subset of size k of Ck+1 that is not frequent cannot be a subset of a frequent itemset of size k + 1. Function subset in line 5 finds all the candidates of the frequent itemsets included in transaction t. Apriori, then, calculates frequency only for those candidates generated this way by scanning the database.

It is evident that Apriori scans the database at most $k$max+1 times when the maximum size of frequent itemsets is set at $k$max.

The Apriori achieves good performance by reducing the size of candidate sets. However, in situations with very many frequent itemsets, large itemsets, or very low minimum support, it still suffers from the cost of generating a huge number of candidate sets and scanning the database repeatedly to check a large set of candidate itemsets. In fact, it is necessary to generate 2100 candidate itemsets to obtain frequent itemsets of size 100.

Page Rank Algorithm

PageRank [12] was presented and published by Sergey Brin and Larry Page at the Seventh International World Wide Web Conference (WWW7) in April 1998. It is a search ranking algorithm using hyperlinks on theWeb. Based on the algorithm, they built the search engine Google, which has been a huge success. Now, every search engine has its own hyperlink based ranking method. PageRank produces a static ranking of Web pages in the sense that a PageRank value is computed for each page off-line and it does not depend on search queries. The algorithm relies on the democratic nature of the Web by using its vast link structure as an indicator of an individual page's quality. In essence, PageRank interprets a hyperlink from page x to page y as a vote, by page x, for page y. However, PageRank looks at more than just the sheer number of votes, or links that a page receives. It also analyzes the page that casts the vote. Votes casted by pages that are themselves "important" weigh more heavily and help to make other pages more "important". This is exactly the idea of rank prestige in social networks

The algorithm

Let us first state some main concepts in the Web context.

In-links of page *i* : These are the hyperlinks that point to page i from other pages. Usually, hyperlinks from the same site are not considered.

Out-links of page *i* : These are the hyperlinks that point out to other pages from page *i* . Usually, links to pages of the same site are not considered. The following ideas based on rank prestige are used to derive the PageRank algorithm:

1. A hyperlink from a page pointing to another page is an implicit conveyance of authority to the target page. Thus, the more in-links that a page i receives, the more prestige the page *i* has.
2. Pages that point to page *i* also have their own prestige scores. A page with a higher prestige score pointing to *i* is more important than a page with a lower prestige score pointing to *i* . In other words, a page is important if it is pointed to by other important pages.

According to rank prestige in social networks, the importance of page *i* (*i* 's PageRank score) is determined by summing up the PageRank scores of all pages that point to *i*. Since a page may point to many other pages, its prestige score should be shared among all the pages that it points to.

To formulate the above ideas, we treat the Web as a directed graph *G = (V, E)*, where *V* is the set of vertices or nodes, i.e., the set of all pages, and *E* is the set of directed edges in the graph, i.e., hyperlinks. Let the total number of pages on the Web be *n* (i.e., *n* = |*V*|).

The PageRank score [12] of the page *i* (denoted by *P(i)*) is defined by

$$P(i) = \sum_{(j,i)\in E} \frac{P(j)}{O_j},$$

where *Oj* is the number of out-links of page *j* . Mathematically, we have a system of *n* linear equations (12) with *n* unknowns. We can use a matrix to represent all the equations. Let*P* be a *n*-dimensional column vector of PageRank values, i.e.,

P = (P(1), P(2), . . ., P(n))T.

Let *A* be the adjacency matrix of our graph with

$$A_{ij} = \begin{cases} \frac{1}{O_i} \text{ if } (i,j) \in E \\ 0 \text{ otherwise} \end{cases}$$

We can write the system of n equations with $\mathbf{P} = \mathbf{A}^T\mathbf{P}$

This is the characteristic equation of the *eigensystem*, where the solution to *P* is an *eigenvector* with the corresponding *eigenvalue* of 1. Since this is a circular definition, an iterative algorithm is used to solve it. It turns out that if some conditions are satisfied, 1 is the largest eigenvalue and the PageRank vector *P* is the principal eigenvector. A well known mathematical technique called power iteration can be used to find *P*.

The power iteration method for PageRank PageRank-Iterate(G)

P0 ← **e**/n
k ← 1
repeat
(1) ; k -1
T
k **P** ← – d **e** + d**A P**
k ← k + 1;
until ||**P**k – **P**k-1||1 < ε
return **P**k
The PageRank formula for each page i:

$$P(i) = (1 \quad d) + d \sum_{j=1}^{n} A_{ji} P(j).$$

D= damping factor, its value is between 0 and 1.

6. CONCLUSION

The importance of knowledge that can be derived from vast oceans of data available to us cannot be downplayed. Effective tools and techniques for extracting this knowledge from data are always precious to human kind, particularly, scientists for researching, to businessmen for promoting favorable practices or targeted marketing of products or services for better revenues. Big data as the name indicates is huge reservoir of structured, unstructured and un- structured data that holds enormous amount of valuable knowledge. Our researchers are earnestly working out new and more effective algorithms and techniques to extract knowledge from huge data sets and put it to good use in research, medicine etc.

REFERENCES

[1] **http://web.engr.illinois.edu/~hanj/bk2/**
[2] en.wikipedia.org/wiki/**Big_data**
[3] **http://www.mongodb.com/big-data-explained**
[4] Brin S, Page L (1998) The anatomy of a large-scale hypertextualWeb Search Sngine. Comput Networks 30(1–7):107–117 management of data, pp. 13–23
[5] Chi Y, Wang H, Yu PS, Muntz RR (2006) Catch the moment: maintaining closed frequent itemsets over a data stream sliding window. Knowl Inf Syst 10(3):265–294
[6] Cost S, Salzberg S (1993) A weighted nearest neighbor algorithm for learning with symbolic features.Mach Learn 10:57.78 (PEBLS: Parallel Examplar-Based Learning System)
[7] Cover T, Hart P (1967) Nearest neighbor pattern classification. IEEE Trans Inform Theory 13(1):21–27
[8] Dasarathy BV (ed) (1991) Nearest neighbor (NN) norms: NN pattern classification.
[9] **w**ww.cloudera.com/content/cloudera/en/.../**hadoop**-and-big-data.html
[10] **hadoop**.apache.org/
[11] web.engr.illinois.edu/~hanj/bk2/slidesindex.htm
http://www.math.cornell.edu/~mec/Winter2009/Raluca Remus/Lecture3/lecture3.html