ISSN:2348-2079

Volume-6 Issue-2

International Journal of Intellectual Advancements and Research in Engineering Computations

Summarize and Discover Hidden Patterns for Decision Making using Big Data Analytics

[1]Ms. N. Zahira Jahan, M.C.A., M.Phil., Associate Professor/MCA
[2]Mr. K. Kavin Kumar, III MCA
Department of MCA, Nandha Engineering College (Autonomous), Erode-52.
Email ID: zahirajahan1977@gmail.com, kavinkumarkaruppanan@gmail.com

***Abstract*—In this project used to Actionable intelligence It means much more than simply finding or summarizing data; it stresses the discovery of hidden (unknown and hard to find)patterns that can be used to predict concepts, events, trends, opinions and so on to support decision makers.The expected use of actionable intelligence varies from system to system, with each variation increasing the complexity involved when engineering such systems. For this reason, it's important to derive taxonomy for understanding actionable intelligence and for thinking about how it affects the complexity and development of big data software. The following three levels (L1, L2, L3) of actionable intelligence are proposed. (L1)Supervised actionable intelligence, (L2) Semi supervised actionable intelligence and (L3)Unsupervised actionable intelligence.**

The supervised system serves as decisionsupport for humans, so that humans supervise the intelligenceproduced by machines before takingaction. Semi supervised machines are relied upon to take actions based on self-produced intelligence; however, these actions can be reversed or corrected by humans. Unsupervised system machines are fully relied upon to take actions without requiring human intervention or correction.

***IndexTerms* — Actionable Intelligence, Hidden patterns, Map Reduce, Multi-hop, Semi supervised, Single-hop, Supervised, Un-supervised.**

I.INTRODUCTION

A survey of the literature reveals numerous attempts at defining big data, varying based on context, domain, and perspective. From the infrastructure's perspective, big data has been defined as data with high volume, velocity, and variety (3V), and unpredictability.

In this context, it has also been defined as data with some aspect that's so large that current, typical methods can't be used to process it.1,2 From the analytics' perspective, big data Has been defined as data so large that it contains significant low probability events that would be absent from traditional Statistical sampling methods. From the business user's perspective, big data represents opportunities for gaining a competitive advantage by gaining actionable intelligence.

Each of these definitions provides descriptive sand important aspects that must be supported by big data software. Borrowing from these definitions, we propose a definition for big data software as "software that supports the time-constrained processing of continuous information flows to provide actionable intelligence." The phrase software that supports acknowledges that big data software includes both infrastructure and analytics software.

Infrastructure software is needed to store, retrieve, transmit, and process big data. While it's essential to developing big data software, much of the emphasis and hype has been placed on the analytics portion of big data software. Nonetheless, our definition of big data software encompasses both types of software. The term time-constrained denotes the urgency in providing solutions. In a way, big data software shares a similar property with real-time software: late responses are wrong responses. The phrase continuous information flows generalizes the input of big data software, which has the unique properties of volume, velocity, and variety.

This generalization also extends to other important information properties of big data input, such as continuity (data in motion versus data at rest). Data in motion (or data streams) can include potentially infinite data streams, at high speeds of arrival, with data whose density changes over time. Finally, actionable intelligence generalizes the output of big data software that is, big data software exists to provide information that's directly useful for immediate (strategic, operational, or tactical) use without having to go through the full intelligence production process.

Verifying the results of massive data stream processing is impractical for humans, if not impossible. it's the reason why big data software exists in the first place. Throughout this process, many defects can go unnoticed. The research literature reports an "extreme imbalance between the numbers

of published algorithms versus those really workable in the business environment," and reasons given for this imbalance include academic researchers not taking into account business environments on the way.

II. LITERATURE SURVEY

A. Carlos E. Otero[2]

On context, domain, and perspective from the infrastructure's perspective, big data has been defined as data with high volume, velocity, and variety (3V), and unpredictability. In this context, it has also been defined as data with some aspect that's so large that cur- rent, typical methods can't be used to process it. From the analytics' perspective, big data has been defined as data so large that it contains significant low probability events that would be absent from traditional statistical sampling methods. From the business user's perspective, big data represents opportunities for gaining a competitive ad- vantage by gaining actionable intelligence. Each of these definitions provides descriptive and important aspects that must be sup- ported by big data software.

Borrowing from these definitions, we propose a definition for big data software as "software that supports the timeconstrained processing of continuous information flows to provide actionable intelligence." The phrase software that supports acknowledges that big data software includes both infrastructure and analytics software these have been referred as big throughput and big analytics software, respectively. Infrastructure software is needed to store, retrieve, transmit, and process big data. While it's essential to developing big data soft- ware, much of the emphasis and hype has been placed on the analytics portion of big data software. Nonetheless, our definition of big data software encompasses both types of software.

B. Adrian Peter[1]

The term has been in use since the 1990s, with some giving credit to John Mashey for coining or at least making it popular. Big data usually includes data sets with sizes beyond the ability of commonly used software tools to capture, curate, manage, and process data within a tolerable elapsed time.Big Data philosophy encompasses unstructured, semi-structured and structured data; however the main focus is on unstructured data.Big data "size" is a constantly moving target, as of 2012 ranging from a few dozen terabytes to many pet bytes of data. Big data requires a set of techniques and technologies with new forms of integration to reveal insights from datasets that are diverse, complex, and of a massive scale.

A consensual definition that states that "Big Data represents the Information assets characterized by such a High Volume, Velocity and Variety to require specific Technology and Analytical Methods for its transformation into Value". Additionally, a new V "Veracity" is added by some organizations to describe it,revisionism challenged by some industry authorities.The 3Vs have been expanded to other complementary characteristics of big data. In the case of Twitter sentiment analysis, attackers can simply create a public account to inject pollutant datasets.

III. ACTIONABLE INTELLIGENCE

Actionable intelligence means much more than simply finding or summarizing data; it stresses the discovery of hidden (unknown and hard to find) patterns that can be used to predict concepts, events, trends, and opinions, and so on to support decision makers. The expected use of actionable intelligence varies from system to system, with each variation increasing the complexity involved when engineering such systems. For this reason, it's important to derive taxonomy for understanding actionable intelligence and for thinking about how it affects the complexity and development of big data software.

- Level 1 (L1), supervised actionable intelligence;
- Level 2 (L2), semisupervised actionable intelligence;
- Level 3 (L3), unsupervised actionable intelligence.

Another important concept that needs to be explained before examining the engineering of big data software involves the intelligence production process. For big data software systems, it occurs primarily in two ways: single- and multi-hop processing. In single-hop processing, data is processed directly by a single component, and from that processing, all required actionable intelligence is produced and prepared for its intended consumer (human or machine), A much more difficult approach (from the engineering perspective) involves multi-hop processing, where interdependencies between components are necessary to extract detailed actionable intelligence.

In multi-hop processing, the output of one component is the input of another. Multi-hop processing is capable of providing various level of actionable intelligence. Such systems are expected to process continuous flows of YouTube news videos, transform input via speech-to-text, and extract sources and topics of conversation via text mining. Concurrently, the system monitors Twitter data, employs topic extraction seeking topics of interests, and uses sentiment analysis to predict public opinion. At each hop, the system produces actionable intelligence that's consumed by other components, and at the end, aggregate and correlate data to provide intelligence that can be used to determine who's saying what and the perceived public reaction to that event.

IV. BIG DATA ANALYTICS SOFTWARE

Having discussed big data software, its inputs, and outputs, let's turn our attention to the main problems encountered when attempting to engineer big data software. This requires the

ability to specify requirements, design and construct software to meet those requirements, and verify through testing that the software meets those requirements. The focus is placed mainly on the big analytics portion of big data software.

A. SINGLE-HOP PROCESSING OF INFORMATION FLOW

In single-hop processing, data is processed by a single component and the output made readily available for immediate use. If this is the case, any inconsistencies in the results can be detected easier than in the multi-hop case.

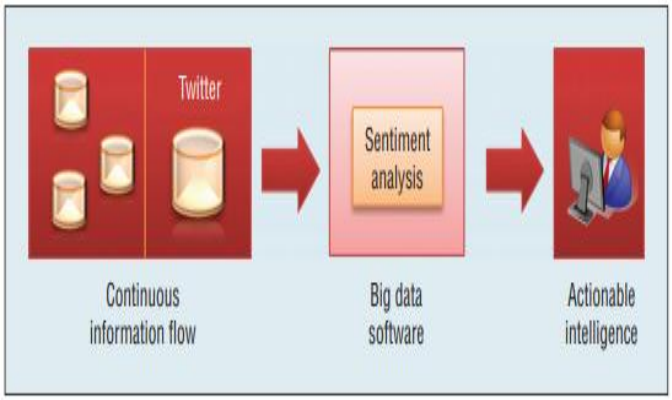

Fig1.Single-hop processing

B. MULTI-HOP PROCESSING OF CONTINUOUS INFORMATION FLOW

In multi-hop processing, data is processed by multiple components before making the results available for use. If this is the case, any inconsistencies in the results would be more difficult to detect since it will not be obvious which component produced unreliable results. Also, any unreliable output in the multi-hop sequence will trickle through all subsequent processing.

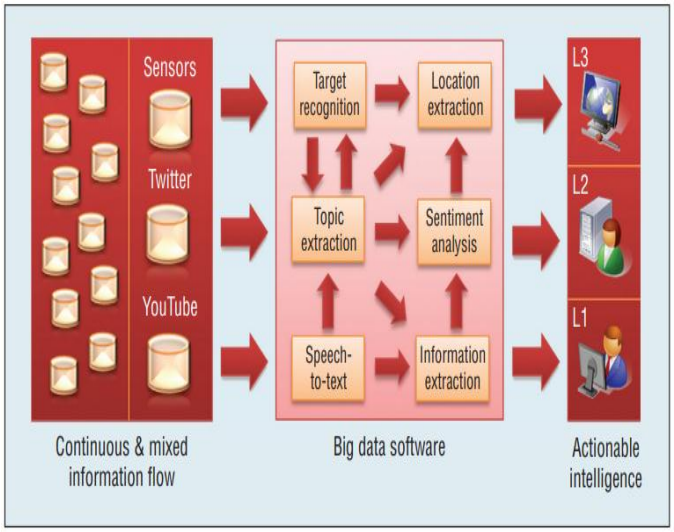

Fig 2. Multi-hop processing

C. SERVER CLIENTS

The Client-server computing or networking is a distributed application architecture that partitions tasks or workloads between service providers (servers) and service requesters, called clients. Often clients and servers operate over a computer network on separate hardware. A server machine is a high-performance host that is running one or more server programs which share its resources with clients. A client also shares any of its resources; Clients therefore initiate communication sessions with servers which await (listen to) incoming requests.

D. DISTRIBUTED DATA

We develop a distributed algorithm to solve the problem on multiple machines in a parallel manner. Our basic idea is to decompose the original large-scale problem into several distributive solvable sub problems that are coordinated by a high-level master problem. We jointly consider data partition and aggregation for a Map Reduce job with an objective that is to minimize the total network traffic.In particular, we propose a distributed algorithm for big data applications by decomposing the original large-scale problem into several sub problems that can be solved in parallel. Moreover, an online algorithm is designed to deal with the data partition and aggregation in a dynamic manner. Finally, extensive simulation results demonstrate that our proposals can significantly reduce network traffic cost in both offline and online cases.

E. SCHEDULING TASK

Map Reduce divides a computation into two main phases, namely map and reduce which in turn are carried out by several map tasks and reduce tasks, respectively. In the map phase, map tasks are launched in parallel to convert the original input splits into intermediate data in a form of key/value pairs. These key/value pairs are stored on local machine and organized into multiple data partitions, one per reduce task. In the reduce phase, each reduce task fetches its own share of data partitions from all map tasks to generate the final result.

F. NETWORK TRAFFIC TRACES

The Network traffic within a Map Reduce job, we consider to aggregate data with the same keys before sending them to remote reduce tasks. Although a similar function, called combiner has been already adopted by Hadoop, it operates immediately after a map task solely for its generated data, failing to exploit the data aggregation opportunities among multiple tasks on different machines. As an example shown in Fig. 2 in the traditional scheme, two map tasks individually send data of key K1 to the reduce task. If we aggregate the data of the same keys before sending them over the top switch, as shown in the network traffic will be reduced. We tested the real network traffic cost in Hadoop using the real data source from latest dumps files in Wikimedia (http://dumps.wikimedia.org/enwiki/latest/).

G. MAP REDUCE TASK

We focus on Map Reduce performance improvement by optimizing its data transmission optimizing network usage can lead to better system performance and found that high network utilization and low network congestion should be achieved simultaneously for a job with good performance.

Map Reduce resource allocation system, to enhance the performance of Map Reduce jobs in the cloud by locating intermediate data to the local machines or close-by physical machines locality-awareness reduces network traffic in the shuffle phase generated in the cloud data center. However, little work has studied to optimize network performance of the shuffle process that generates large amounts of data traffic in Map Reduce jobs.A critical factor to the network performance in the shuffle phase is the intermediate data partition. The default scheme adopted by Hadoop is hash-based partition that would yield unbalanced loads among reduce tasks due to its unawareness of the data size associated with each key.

V. REQUIREMENTS PROBLEM

Requirement specification is crucial to the successful construction of software. The requirements problem is better presented using a simple model for functional requirements' specification and verification. Consider the simplest form of requirement specification—one that specifies a single, verifiable function that the system must provide: ri ← fi(xi) Whererequirementri is a function of some functional feature xi that the system needs to provide. Systems specified with such requirements are easily verified, simply by enumerating all requirements and verifying that the product of each individual requirement function evaluates to 1.

That is, assuming a vector x of n functional features that the software needs to provide, a single requirement can be specified for each feature i and verification of the software can be modeled using the equation below, where V = 1 denotes successful verification: =∏ = V f ( ) x . Because big data software shares the same time-constrained properties as real-time software, time-constrained requirements also apply to big data software. At first glance, it would seem that Vreal-time is appropriate for verifying big data software. However, both specification and verification of big data analytics software are far more troublesome because a major function expected from such systems involves making predictions on data streams.

Every big data software product exists to process data that's dynamic, fast-changing, and most crucially, subject to concept drifts. In big analytics, the term concept drift refers to changes in statistical properties of the concept to be learned. This phenomenon applies to a host of stochastic learning modelsgraphical models, Monte Carlo sampling methodologies, naïve Bayesian classifiers, and so onwhose learned model parameters are directly correlated to the data stream used for their estimation.These changes occur over time in unforeseen ways, thus causing predictions to become less accurate as time passes. This means that big analytics software could be specified to meet a specific accuracy threshold (using a given test dataset) and verified over some period p.

TheConcept evolution, the emergence of new classes, is also another data dynamic that arises in rapidly evolving big data streams. Several algorithms address concept drift and concept evolution; however, our concern is with the requirements and verification modeling of the software that could support these functionalities, not with the peculiarities of any one specific methodology. If the analytics software is required to incorporate the detection of new classes, the proposed V Big Data verification model captures this desire inherently as this would be yet another time- or periodically-constrained functional requirement.

It has significant engineering implications. The most notable one is that requirements specified for big data analytics software may (at most) result in systems that can only be verified reliably during some period pi. Therefore, requirements for such systems need to be expressed in a way that they can be monitored automatically against software system events, such as concept drifts. A suggested approach involves the use of satisfaction arguments, where the verifiability of requirements can only be argued after assuming something about the domain.11 These new forms of requirements might result in additional software that needs to be integrated into the system, which could result in increased cost and extended schedules, since most likely they will have significant implications in the design, construction, testing, and maintenance phases.

The addition of p also requires a paradigm shift in the way software contracts are thought about. Companies contracted to develop big data analytics software (such as US Department of Defense contractors) will have to deal with these issues in some way, especially as big data software continues to find its way into mainstream applications. These could possibly include application of big data software to national security, government policymaking, or safety-critical applications, in which case, other legal issues addressed by professional licensure would have to be taken into consideration.

VI. DESIGN PROBLEM

A glance at the literature points to certain key quality properties that are often cited for big data software, including usability, performance, reliability, availability, security, interoperability, scalability, and testability, among others. Discussing all of these in the right amount of detail requires far more space than the one allotted for this article. For this reason, focus is placed on the reliability and security of big data software.

Aactionable intelligence in the form of predicted (positive, neutral, or negative) sentiment. At best, the system provides the best-effort prediction of sentiment. This prediction could be highly accurate, but minimal or no guarantees are made about the reliability of such prediction in some cases, this would even be difficult for humans. Under single-hop, L1 and L2 actionable intelligence conditions, verifying the reliability

is difficult, but not impossible, since humans can use judgment and domain knowledge to visually detect inaccurate or extreme results.

However, under multi-hop, L3 actionable intelligence conditions, the state of the art is insufficient for equipping software with such functionality. The question then arises, how software engineers design for reliability in big data software.

The design problem is related to the specification problem discussed earlier and is an open area of research in software engineering design. Suggested solutions include discovering new (or adapting old) architectural tactics to help increase the reliability of such systems.For example, architectural tactics that exploit deterministic metadata can be incorporated to help concurrently gauge the reliability of predictions.

To deal with concept drift, a concept-drift monitor or change detection component could be incorporated into the software design to detect and alert users when analytic models are no longer reliable. Two types of security attacks that exist for big data software involve execution phase attacks and training phase attacks. In the former, data streams are tapped to influence the actionable intelligence generated by the big data software. Consider the case of important organizations (such as government) using big data software to gauge public opinion from social media for a particular topic.

Verifying the results of massive data stream processing is impractical for humans; if not impossible it's the reason why big data software exists in the first place. Throughout this process, many defects can go unnoticed. The research literature reports an "extreme imbalance between the numbers of published algorithms versus those really workable in the business environment," and reasons given for this imbalance include academic researchers not taking into account business environments.

The construction problem is mainly driven by big data software requiring more than just programming skills. Today, most learning algorithms are created by machine learning scientists, who aren't necessarily the best programmers.18 Specific issues mentioned in the literature include sloppiness, maintainability, debugging, and reproducibility.18 Most of these problems have been addressed within the software engineering community where coding styles, peer reviews, configuration management software, and other tools are employed. However, a common trend exists where many machine learning advances are being used for improving software engineering, but not enough of software engineering advances are being used for improving machine learning software.

Other construction issues include lack of tools and frameworks that support assistive development. Popular development applications such as Waikato Environment for Knowledge Analysis (WEKA), Excel, Octave, or R are designed for machine learning software but not big data software. Applications of this sort require all the data to reside in memory, which isn't the case for big data. Some tools are beginning to exist, such as the Massive Online Analysis (MOA) framework for data streaming,8 but more are needed. Construction for big data software is also particularly complex because the inherent nature of big data requires multiprocessing technologies, such as CUDA, GPUs, Map Reduce, Hadoop, and so on, which must be mastered during the construction phase.

Recently, new software frameworks, specifically targeted at improving the usability of these technologies, are striving to reduce the learning curve for big data software engineering. Examples include Mahout, which leverages the Hadoop processing environment for large-scale machine learning; Spark, built on the Scale language; and Storm for real-time analysis. Undoubtedly, by the time of publication, a few more might have arrived.

Although not the focus of our present effort, it's worth mentioning that efficient big data analytics also requires you to keep in mind the distributed data store and retrieval technologies underlying the analytics computation. These technologies and frameworks include No Sql (supported in Cassandra and Hyper table) and SciDB. Under this massive and fast-changing technology landscape, developers not only have to keep up with the state of the art, but they must also expend significant development effort in experimenting with different algorithms, multiprocessing techniques, and software frameworks. This, in turn, will strongly impact cost and schedules.

VII. CONCLUSION

No one can deny the evidence found in today's mainstream big data software, such as Facebook, Netflix, and Amazon. Surely, a great deal of successful software engineering work has taken place. Perhaps a more relevant question is, has the rest of the software engineering community mastered the engineering of such systems.

Today, engineering these systems still entails seeking the most accurate prediction while accepting solutions that sacrifice accuracy for the sake of not having a solution at all. Many of the issues presented so far, and more, still remainopen to research. Software engineering research is needed to fully understand the role of requirements, design, and testing in big data software, and to characterize their crosscutting effects.

REFERENCES

[1] Adrian Peter is an assistant professor of engineering systems at the Florida Institute of Technology;He's a member of IEEE. Contact him at apeter@fit.edu.

[2] Carlos E Otero., "Research directions" , 2015/1 IEEE Intelligent System, Volume 30, Page 13-19.

[3] J. Cohen et al., "MAD Skills: New Analysis Practices for Big Data,"Proc.VLDBEndowment2009; http://db.cs.berkeley.edu/jmh/papers/madskills 032009. Pdf.

[4] M. Masud et al., "Detecting Recurring and Novel Classes in Concept-Drifting Data Streams," Proc. 2011 IEEE Int'l Conf. Data Mining, 2011, pp. 1176–1181.

[5] D.E. O'Leary, "Artificial Intelligence and Big Data," IEEE Intelligent Systems, vol. 28, no. 2, 2013, pp. 96–99, 2013.