A Data Model for Integrating Data Management and Data Mining in Social Big Data

Hiroshi Ishikawa
Faculty of System Design
Tokyo Metropolitan University
Tokyo, Japan
e-mail: ishikawa-hiroshi@tmu.ac.jp

Richard Chbeir
LIUPPA Lab.
University of Pau and Adour Countries
Anglet, France
e-mail: richard.chbeir@univ-pau.fr

***Abstract*—We propose an abstract data model for integrating data management and data mining necessary for describing social big data applications by using mathematical concepts of families, collections of sets. Our model facilitates reproducibility and accountability required for social big data researches and developments. We have partially validated our proposal by adapting our model to real case studies.**

Keywords- social data; big data; data model; data management; data mining.

I. INTRODUCTION

A. Social Big Data

In the present age, large amounts of data are produced continuously in science, on the internet, and in physical systems. Such phenomena are collectively called data deluge. According to some researches carried by International Data Corporation, or IDC [4][5] for short, the size of data which are generated and reproduced all over the world every year is estimated to be 161 Exa bytes. The total amount of data produced in 2011 exceeded 10 or more times the storage capacity of the storage media available in that year. Experts in scientific and engineering fields produce a large amount of data by observing and analyzing the target phenomena. Even ordinary people voluntarily post a vast amount of data via various social media on the internet. Furthermore, people unconsciously produce data via various actions detected by physical systems, such as sensors and Global Positioning System, or GPS for short, in the real world. It is expected that such data can generate various values. In the above-mentioned research report of IDC, data produced in science, the internet, and in physical systems are collectively called big data. The features of big data can be summarized as follows:

- The quantity (Volume) of data is extraordinary, as the name denotes.
- The kinds (Variety) of data have expanded into unstructured texts, semi-structured data, such as XML, and graphs (i.e., networks).
- As is often the case with Twitter and sensor data streams, the speed (Velocity) at which data are generated is very high.

Therefore, big data is often characterized as V^3 by taking the initial letters of these three terms Volume, Variety, and Velocity. Big data are expected to create not only knowledge in science but also derive values in various commercial ventures. “Volume” and “velocity” require more computing power than ever before. “Variety” implies that big data appear in a wide variety of applications and then data have a wide variety of structures. Further, big data inherently contain “vagueness,” such as inconsistency and deficiency. Such vagueness must be resolved in order to obtain quality analysis results. Moreover, a recent survey done in Japan has made it clear that a lot of users have “vague” concerns as to the securities and mechanisms of big data applications [7]. In other words, service providers deploying big data have accountability for explaining to generic users among stake holders how relevant big data are used. The resolution of such concerns is one of the keys to successful diffusion of big data applications. In this sense, V^4 should be used to characterize big data, instead of V^3. Big data typically include IoT data collected by a variety of networked sensors and mobile gadgets, social data posted at social media sites, such as Twitter and Flickr, and open data published for everyone to access.

In big data applications, especially, cases where two or more data sources including at least one social data source are involved, are more interesting from a viewpoint of usefulness to businesses [7]. If more than one data source can be analyzed by relating them to each other, and by paying attention to the interactions between them, it may be possible to understand what cannot be understood, by analysis of only either of them. For example, even if only sales data are deeply analyzed, reasons for a sudden increase in sales, that is, what has made customers purchase more products suddenly, cannot be known. By analysis of only social data, it is impossible to know how much they contributed to sales, if any. However, if both sales data and social data can be analyzed by relating them to each other, it is possible to discover why items have begun to sell suddenly, and to predict how much they will sell in the future, based on the results. In a word, such integrated analysis is expected to produce bigger values than otherwise. We would like to call such an analytic methodology Social Big Data, or SBD for short.

Even if only one social data source, such as Twitter articles and Flickr images is available and if such articles and images have geo-tags (i.e., location information), as well, social big data mining is useful. That is, by collecting those articles and images based on conditions specified with respect to locations and time intervals and counting them for each grid (i.e., unit location), probabilities that users post such data at the locations can be basically computed. By using such probabilities, human activities can be analyzed, such as probabilities of foreigners staying at specific spots or those moving from one spot to another. The results will be applied to tourism and marketing.

Furthermore, a certain level of location can be

 ISBN: 978-1-61208-548-7

represented as a collection of lower levels of locations. Similarly, a time interval can be divided into a collection of shorter time intervals. As such, locations and time have hierarchical structures inherently.

B. Reproducibility

In general, the validity of published results of scientific researches has recently been judged based on not only traditional peer reviews but also reproducibility [15]. Reproducibility means that the same results with reported ones can be obtained by independent researchers. Success of reproduction hinges on detailed descriptions of methods and procedures, as well as data which have led to the published results.

At an extreme end of reproducibility spectrum [1] [10] is repeatability. Repeatability in computer science means that independent researchers can obtain exactly the same results by using the same data and the same codes that the reporters used. However, it is not always possible to use the same data and codes due to several reasons, such as limited space for publishing research results and lack or delayed spread of related standardization. Rex [11] is among ambitious attempts to facilitate repeatability. Rather, reproduction is done in order to make certain the essence of the experiments.

All this is true of SBD researches and developments. Reproducibility in researches and developments of SBD applications requires at least the following requirements.

- Description of SBD applications must be as independent from individual programming languages and frameworks as possible. Generally speaking, it is not always possible for all researchers to access the same data and tools that the authors have used. In other words, by enabling the mapping from description of applications by an abstract *SBD model* proposed in this paper to individual tools available for the other researchers, reproducibility can be realized even if the tools are not the same with the original one. Therefore, application description (i.e., at conceptual level) must be more abstract than codes (i.e., at logical level). It is expected that the amount of description is reduced by this. The descriptions must be even as independent from programing models, such as parallel computing as possible. This *SBD model* approach can lead to increase of accountability of SDB applications to stake holders including generic users.
- Both data management and data mining must be described in an integrated manner. In SBD applications, a lot of time is spent on development and execution of data management including preprocessing and postprocessing in addition to data mining. Further, data management and data mining cannot be always separated in a crisp manner. Rather, most SBD applications require hybrid processes mixed with data management and data mining. Later, such examples will be described in case studies.

This paper, which is rather positional, introduces an integrated data model for describing SBD applications and describes applications using the model as case studies. The reference architecture for SBD is illustrated in Figure 1.

C. Relation with Other Work

To our knowledge, there are no abstract data models that can handle data management and data mining. Indeed, there exist a lot of programming languages and frameworks that can host data management and data mining, such as Spark [17] and MLI [14]. However, such language interfaces have different levels of abstraction from those of our data model proposed in this paper. Rather, those are among candidate targets to which our abstract model can be translated. Section II introduces a data model for SBD and Section III explains case studies for SBD model.

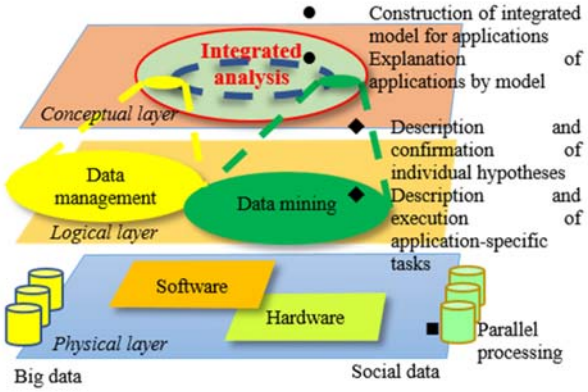

Figure 1. The reference architecture for social big data.

II. DATA MODEL FOR SBD

A. Overview

We propose an abstract data model as an approach to reinforcing both reproducibility and accountability of SBD. Our SBD model aims to satisfy the following requirements:

- Enable to describe data management and data mining in an integrated fashion or seamlessly.
- Be independent enough from existing programming languages and frameworks and easy enough to translate into executable programming languages, as well.

We extend relational model [12] prevalent in data management fields as our approach to SBD model. The Relational model is based on a mathematical concept of a *set*. On the other hand, data mining includes clustering, classification, and association rules [7]. Clustering partitions a given set of data into a collection of sets, each of which has elements similar to each other. Classification divides a given set of data into pre-scribed categories, that is, a collection of sets by using supervised learning. Association rule mining discovers a collection of frequent itemsets collocating in transactional data. Unlike relational models, all these data mining techniques handle a *collection of sets* instead of a set. In other words, we must bridge gaps between data management and data mining with respect to levels of granularity. At the same time, we would like to adopt abstractness comparable to those that relational models [12] and object models [6] have because such abstractness is widely prevalent.

 ISBN: 978-1-61208-548-7

B. Data Structure

Our SBD model uses a mathematical concept of a *family* [13], a collection of sets, as a basis for data structures. Family is an apparatus for bridging the gaps between data management operations and data analysis operations.

Basically, our database is a *Family*. A Family is divided into *Indexed family* and *Non-Indexed family.* A Non-Indexed family is a collection of sets or a collection of Non-Indexed families. In other words, a Non-indexed family can constitute a hierarchy of sets.

- {*Set*} is a Non-Indexed family with *Set* as its element.
- {Set_i} is an Indexed family with Set_i as its *i-th* element. Here *i: Index* is called *indexing set* and *i* is an element of Index. To be more exact, Index is either a set or an Indexed-family. In other words, Index itself can also be nested like Non-Indexed family.
- *Set* is {<*time space object*>}.
- Set_i is {<*time space object*>}$_i$. Here, *object* is an identifier to arbitrary identifiable user-provided data, e.g., record, object, and multimedia data appearing in social big data. *Time* and *space* are universal keys across multiple sources of social big data.

Please note that the following concepts are interchangeably used in this paper.

- Singleton family ⇔ set
- Singleton set ⇔ element

As described later in Section III, we can often observe that SBD applications contain families, as well as sets because such applications involve both data mining and data management. Please note that family is also suitable for representing hierarchical structures inherent in time and locations associated with social big data.

If operations constructing a family out of a collection of sets and those deconstructing a family into a collection of sets are provided in addition to both family-dedicated and set-dedicated operations, SBD applications will be described in an integrated fashion by our proposed model.

C. SBD Operations

SBD model constitutes an algebra with respect to Family as follows.

SBD is consisted of Family data management operations and Family data mining operations. Further, Family data management operations are divided into Intra Family operations and Inter Family operations.

1) Intra Family Data Management Operations

a) Intra Indexed Intersect (*i:Index Db p(i)*) returns a singleton family (i.e., set) intersecting sets which satisfy *p(i)*. Database *Db* is a Family, which will not be mentioned hereafter.

b) Intra Indexed Union *(i:Index Db p(i)*) returns a singleton family union-ing sets which satisfy *p(i)*.

c) Intra Indexed Difference (*i:Index Db p(i)*) returns a singleton family, that is, the first set satisfying *p(i)* minus all the rest of sets satisfying *p(i)*

d) Indexed Select (*i:Index Db p1(i) p2(i)*) returns an Indexed family with respect to *i* (preserved) where the element sets satisfy *p1(i)* and the elements of the sets satisfy *p2(i)*. As a special case of true as *p1(i)*, this operation returns the whole indexed family. In a special case of a singleton family, Indexed Select is reduced to Select (a Relational operation).

e) Indexed Project (*i:Index Db p(i) a(i)*) returns an Indexed family where the element sets satisfy *p(i)* and the elements of the sets are projected according to *a(i)*, attribute specification.

f) Intra Indexed cross product (*i:Index Db p(i)*) returns a singleton family obtained by product-ing sets which satisfy *p(i)*. This is extension of Cartesian product, one of relational operators.

g) Intra Indexed Join (*i:Index Db p1(i) p2(i)*) returns a singleton family obtained by joining sets which satisfy *p1(i)* based on the join predicate *p2(i)*. This is extension of join, one of relational operators.

h) Sort (*i:Index Db p(i) o()*) returns indexed family where the element sets satisfy *p(i)* and the elements of the sets are ordered according to *compare function o() with respect to two elements*.

i) Indexed Sort (*i:Index Db p(i) o()*) returns an indexed family where the element sets satisfy *p(i)* and the sets are ordered according to *o()*, compare function with respect to two sets.

j) Select-Index (*i:Index Db p(i)*) returns *i:Index* of set_i which satisfy *p(i)*. As a special case of true as *p(i)*, it returns all index.

k) Make-indexed family (*Index Non-Indexed Family*) returns an indexed Family. This operator requires *order-compatibility*, that is , that *i* corresponds to *i-th set of Non-Indexed Family*.

l) Partition (*i:Index Db p(i)*) returns an Indexed family. Partition makes an Indexed family out of a given set (i.e. singleton family either w/ or w/o index) by grouping elements with respect to *p (i:Index).* This is extension of "groupby" as a relational operator.

m) ApplyFunction (*i:Index Db f(i)*) applies *f(i)* to *i-th* set of DB, where *f(i)* takes a set as a whole and gives another set including a singleton set (i.e., Aggregate function). This returns an indexed family. *f(i)* can be defined by users.

2) Inter Family Data Operations Index-Compatible

a) Indexed Intersect (*i:Index Db1 Db2 p(i)*) union-compatible

b) Indexed Union (*i:Index Db1 Db2 p(i)*) union-compatible

c) Indexed Difference (*i:Index Db1 Db2 p(i)*) union-compatible

d) Indexed Join (*i:Index Db1 Db2 p1(i) p2(i)*)

e) Indexed cross product (*i:Index Db1 Db2 p(i)*)

Indexed (*) operation is extension of its corresponding Relational operation. It preserves an Indexed Family. For example, Indexed Intersect returns Indexed family whose element is intersection of corresponding sets of two indexed families *Db1* and *Db*2, which satisfy *p(i)*. At this time, we impose *union-compatibility*. Further, in case both *Db1* and *Db*2 are singleton families and *p(i)* is constantly true, Indexed

 ISBN: 978-1-61208-548-7

Intersect is reduced to Intersect, which returns intersection of two sets (a Relational operation). Indexed Union and Indexed Difference are also similar.

3) Family Data Mining Operations

a) Cluster (*Family method similarity* {*par*}) returns a Family as default, where Index is automatically produced. This is an unsupervised learner. In *hierarchical agglomerative clustering* or HAC, as well as similar methods, such as some spatial, temporal, and spatio-temporal clustering, index is merged into new index as clustering progresses. *method* includes k-means, HAC, spatial, temporal, etc. *similarity/distance* includes Euclidean, Cosine measure, etc. *par* (ammeters) depend on *method.*

b) Make-classifier (*i:Index set:Family learnMethod* {*par*}) returns a classifier (Classify) with its accuracy. This is a supervised learner. In this case *index* denotes classes (i.e., predefined categories). Sample *set* includes both training set and test set. *learnMethod* specifies methods, such as decision tree, SVM, deep learning. *par* (ammeters) depend on *learnMethod.* This operation itself is out of range of our algebra. In other words, it is a *meta*-operation.

c) Classify (*Index/class set*) returns an indexed family with class as its index.

d) Make-frequent itemset (*Db supportMin)* returns an Indexed Family as frequent itemsets, which satisfy *supportMin*.

e) Make-association-rule (*Db confidenceMin*) creates association rules based on frequent itemsets *Db*, which satisfy *confidenceMin*. This is out of range of our algebra, too.

III. CASE STUDY

A. Case One

First, we describe a case study, analysis of behaviors of foreigners (visitors or residents) in Japan [3]

We use the following colors for each category of SBD operations for illustration:

- Relational (set) operation
- Family operation
- Data mining operation

To classify foreign users as *residents* or *visitors*, we will classify the length of the stay in the country of interest as *long* and *short*, respectively. We assume the target country (i.e., the country of interest) uses one language dominantly. We first obtain the tweets that a user posted in Japan. We detect the principal language of the user in order to extract only foreign Twitter users. We define the *principal language* of a user as the language that meets the following two conditions.

- The language must be used in more than half of all the user's tweets. Since the Language-Detection toolkit [8] is over 99% precision according to their claim in detecting the tweet language, we used this toolkit in the experiment.
- The language must be selected by the user in his/her account settings. This means that the user claims that they use that language.

If the resultant principal language for a Twitter user is a language other than the one dominantly used in the target country, we regard the user as a foreign Twitter user and then classify the user as residents or visitors.

First, we sort a user's tweets posted in the target country in chronological order, where t_i denotes *i*-th tweet. Next, we set parameters *start date* and *stop date*, which specify the start and end date of interest, respectively. We define the oldest tweet between *start date* and *stop date* as T_{old} and define the parameter *travel period* as the maximum length of the stay. We define the newest tweet between T_{old} and T_{old} + *travel period* as T_{new}. Also, we set parameter *j*, a margin that ensures the foreign user is out of the target country. We identify a foreign user's tweets during a visit, if and only if all his/her tweets satisfy the following conditions:

- The foreign user posts more than T_{min} tweets between T_{old} and T_{new} and the user posts no tweets during the period from *j* days before to T_{old} to and the period from T_{new} to *j* days after T_{new}.

Here, T_{min} is the minimum number of tweets to prevent misclassification owing to a small number of tweets. The tweets posted between T_{old} and T_{new} are identified as the tweets during the visit. Since some users repeatedly visit the target country, we repeat the identification of tweets during a visit after T_{new}.

A foreign user is identified as a *visitor* to the target country, if and only if all his/her tweets between *start date* and *stop date* are tweets during visits. Foreign users who are not visitors are identified as *residents*. Here, we excluded foreign users who tweeted equal to or less than T_{min} times between *start date* and *stop date* as *unrecognizable*.

Classify$_{foreign/domestic}$ ({*Foreign Domestic*} DB_{tweet}) binarily splits into foreign and domestic sets as *AccountOrigin* (i.e., index class).

This Classifier is based on a heuristic (i.e., manually-coded) rule for deciding foreigner as follows:

if Count *(t:tweet t.AccountId=AccountId & t.DetectedLanguage()=t.AccountLanguage & t.AccountLanguage<>*"Japanese") >=0.5*Count (t:tweet *t.AccountId=id*) then return foreign else domestic

The following fragment of descriptions collects only tweets posted by foreigners ("←" is the assignment operator):

DB_t ← Sort (Select (DB_{tweet} *Time of Interest* & Within "*Japan"*) *compare-time*()) ; singleton family (i.e., set).

$DB_{foreign}$ ← Indexed-Select (Classify$_{foreign/domestic}$ ({*Foreign Domestic*} DB_t) *AccountOrigin="foreign"*); singleton family.

Next Classify$_{visitor/resident}$ ({*Visitor Resident*} DB_{tweet}) binarily splits into visitor and resident sets as *AccountStatus* (i.e., index class).

This is based on a heuristic rule for deciding inbound visitor as follows:

if Count (*t.tweet t.AccountId=AccountId* & T_{old}=<*t.time*=<T_{new})>=C_{min} & *Count (t.tweet*

t.AccountId=id & T_{old}-*j*<=*t.time*<T_{old})=0 & Count (*t.tweet t.AccountId=id* & T_{new}<*t.time*<=T_{new}+*j*)=0 then return visitor else resident

The following fragment classifies tweets by foreigners into ones by inbound visitors and ones by foreign residents:

$DB_{foreignVisitorOrResident}$ ← Classify$_{visitor/resident}$ ({*Visitor Resident*} $DB_{foreign}$); This returns an indexed family.

$DB_{visitor}$ ← Indexed-Select ($DB_{foreignerVisitorOrResident}$ *AccountStatus*="*visitor*") ; This returns a singleton family.

$DB_{resident}$ ← Indexed-Select ($DB_{foreignerVisitorOrResident}$ *AccountStatus*="*resident*"); This returns a singleton family.

B. Case Two

Next, we describe another case study, finding candidate access spots for accessible free WIFI in Japan [9].

This section describes our proposed method of detecting attractive tourist areas where users cannot connect to accessible Free Wi-Fi by using posts by foreign travelers on social media.

Our method uses differences in the characteristics of two types of SNSs and we focus on two of these:

Real-time: Immediate posts, e.g., Twitter

Batch-time: Data stored to devices for later posts, e.g., Flickr

Twitter users can only post tweets when they can connect devices to Wi-Fi or wired networks. Therefore, travelers can post tweets in areas with Free Wi-Fi for inbound tourism or when they have mobile communications. In other words, we can obtain only tweets with geo-tags posted by foreign travelers from such places. Therefore, areas where we can obtain huge numbers of tweets posted by foreign travelers are identified as places where they can connect to accessible Free Wi-Fi and /or that are attractive for them to sightsee.

Flickr users, on the other hand, take many photographs by using digital devices regardless of networks, but whether they can upload photographs on-site depends on the conditions of the network. As a result, almost all users can upload photographs after returning to their hotels or home countries. However, geo-tags annotated to photographs can indicate when they were taken. Therefore, although it is difficult to obtain detailed information (activities, destinations, or routes) on foreign travelers from Twitter, Flickr can be used to observe such information. We are based on our hypothesis in this study of "A place that has a lot of Flickr posts but few Twitter posts must have a critical lack of accessible Free Wi-Fi". We extracted areas that were tourist attractions for foreign travelers, but from which they could not connect to accessible Free Wi-Fi by using these characteristics of SNSs. What our method aims to find is places without accessible Free Wi-Fi.

There are two main reasons for areas from where foreign travelers cannot connect to Free Wi-Fi. The first is areas where there are no Wi-Fi spots. The second is areas where users can use Wi-Fi but it is not accessible. We treat them both the same as inaccessible Free Wi-Fi because both areas are unavailable to foreign travelers. Since we conducted experiments focused on foreign travelers, we could detect actual areas without accessible Free Wi-Fi. In addition, our method extracted areas with accessible Free Wi-Fi, and then other locations were regarded as regions without accessible Free Wi-Fi.

This subsection describes a method of extracting foreign travelers using Twitter and Flickr. We obtained and analyzed tweets posted in Japan from Twitter using Twitter's Streaming application programming interface (API) [16]. We used the method introduced in Case study to extract foreign travelers.

We obtained photographs with geo-tags taken in Japan from Flickr using. Flickr's API [2]. We extracted foreign travelers who had taken photographs in Japan. We regard Flickr users who had set their profiles of habitation on Flickr as Japan or associated geographical regions as the users *living* in Japan; otherwise, they are regarded as foreign *visitors*. We used the tweets and photographs that foreign visitors had created in Japan in the analysis that followed. Our method envisaged places that met the following two conditions as candidate access spots for accessible free WIFI:

- Spots where there was no accessible Free Wi-Fi
- Spots that many foreign visitors visited

We use the number of photographs taken at locations to extract tourist spots. Many people might take photographs of subjects, such as landscapes based on their own interests. They might then upload those photographs to Flickr. As these were locations at which many photographs had been taken, these places might also be interesting places for many other people to sightsee or visit. We have defined such places as tourist spots in this paper. We specifically examined the number of photographic locations to identify tourist spots to find locations where photographs had been taken by a lot of people. We mapped photographs that had a photographic location onto a two-dimensional grid based on the location at which a photograph had been taken to achieve this. Here, we created individual cells in a grid that was 30 square meters. Consequently, all cells in the grid that was obtained included photographs taken in a range. We then counted the number of users in each cell. We regarded cells with greater numbers of users than the threshold as tourist spots.

Figure 2. High density areas of tweets (left) and of Flickr photos (right).

The fragment collects attractive tourist spots for foreign visitors but without accessible free WIFI currently (See Figure 2):

$DB_{t/visitor}$ ← Tweet DB of foreign visitors obtained by similar procedures like case one;

$DB_{f/visitor}$ ← Flickr photo DB of foreign visitors obtained by similar procedures like case one;

T ← Partition (*i:Index grid* $DB_{t/visitor}$ *p(i)*); This

partitions foreign visitors tweets into grids based on geo-tags; This operation returns a indexed family.

F ← Partition (*j:Index grid* $DB_{f/visitor}$ *p(j)*); This partitions foreign visitors photos into grids based on geo-tags; This operation returns a indexed family.

Index1 ← Select-Index (*i:Index T Density*(*i*) >= *th1*); *th1* is a threshold. This operation returns a singleton family.

Index2 ← Select-Index (*j:Index F Density*(*i*) >= *th2*); *th2* is a threshold.This operation returns a singleton family.

Index3 ← Difference (*Index2 Index1*); This operation returns a singleton family.

For example, grids indexed by *Index3* contain "Osanbashi Pier". Please note that the above description doesn't take unique users into consideration.

IV. CONCLUSION

We have proposed an abstract data model for integrating data management and data mining by using mathematical concepts of families, collections of sets. Our model facilitates reproducibility and accountability required for SBD researches and developments. We have partially validated our proposal by adapting our model to real case studies. However, there still remains to describe mapping from our model to existing programming tools, such as Spark. Further, we must devise some kinds of optimization comparable to query optimization of SQL. We would like to validate our proposed model more thoroughly by adapting it to different kinds of applications and theoretically, stick as well.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number 16K00157, 16K16158, and Tokyo Metropolitan University Grant-in-Aid for Research on Priority Areas Research on social big data.

REFERENCES

[1] D. G. Feitelson, "From Repeatability to Reproducibility and Corroboration," ACM SIGOPS Operating Systems Review - Special Issue on Repeatability and Sharing of Experimental Artifacts, Volume 49, Issue 1, pp. 3-11, January 2015.

[2] Flickr, *The App Garden*. https://www.flickr.com/services/api/ Accessed 2017.03

[3] M. Hirota, K. Saeki, Y. Ehara, and H. Ishikawa, "Live or Stay?: Classifying Twitter Users into Residents and Visitors," Proceedings of International Conference on Knowledge Engineering and Semantic Web (KESW 2016), 2016.

[4] IDC, *The Diverse and Exploding Digital Universe* (white paper, 2008). http:// www.emc.com/collateral/analyst-reports/diverse-exploding-digital-universe.pdf Accessed 2017.03

[5] IDC, *The Digital Universe In 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East* (2012). http://www.emc.com/leadership/digital-universe/iview/index.htm accessed 2017.03

[6] H. Ishikawa, Y. Yamane, Y. Izumida, and N. Kawato, "An Object-Oriented Database System Jasmine: Implementation, Application, and Extension," IEEE Trans. on Knowl. and Data Eng. 8, 2, pp. 285-304, April 1996.

[7] H. Ishikawa, Social Big Data Mining, CRC Press, 2015.

[8] GitHub. Language Detection. https://github.com/shuyo/language-detection Accessed 2017.03

[9] K. Mitomi, M. Endo, M. Hirota, S. Yokoyama, Y. Shoji, and H. Ishikawa, "How to Find Accessible Free Wi-Fi at Tourist Spots in Japan," Volume 10046 of Lecture Notes in Computer Science, pp. 389-403, 2016.

[10] R. D. Peng, "Reproducible Research in Computational Science," SCIENCE, VOL 334, 2, December 2011.

[11] S. Perianayagam, G. R. Andrews, and J. H. Hartman, "Rex: A toolset for reproducing software experiments," Proceedings of IEEE International Conference on Bioinformatics and Biomedicine (BIBM) 2010, pp. 613-617, 2010.

[12] R. Ramakrishnan, and J. Gehrke, Database Management Systems, 3rd Edition, McGraw-Hill Professional, 2002.

[13] D. Smith, R. St. Andre, and M. Eggen , A Transition to Advanced Mathematics, Brooks/Cole Pub Co., 2014.

[14] E. R. Sparks, A. Talwalkar, V. Smith, J. Kottalam, X. Pan, J. E. Gonzalez, M. J. Franklin, M. I. Jordan, and T. Kraska, "MLI: An API for distributed machine learning," Proceedings of the IEEE ICDM International Conference on Data Mining (Dallas, TX, Dec. 7–10). IEEE Press, 2013.

[15] T. C. Südhof, "Truth in Science Publishing: A Personal Perspective," PLOS August 26, 2016.

[16] Twitter. Twitter Developer Documentation. https://dev.twitter.com/streaming/overview Accessed 2017.03

[17] M. Zaharia, R. S. Xin, Patrick Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, "Apache Spark: a unified engine for big data processing," Com. ACM, 59, 11, pp. 56-65, October 2016.

 ISBN: 978-1-61208-548-7