Big Data Dimensional Analysis

Vijay Gadepally and Jeremy Kepner
Lincoln Laboratory
Massachusetts Institute of Technology
Lexington, MA 02470
Email: {vijayg, kepner}@ll.mit.edu

***Abstract*—The ability to collect and analyze large amounts of data is a growing problem within the scientific community. The growing gap between data and users calls for innovative tools that address the challenges faced by big data volume, velocity and variety. The Massachusetts Institute of Technology, Lincoln Laboratory is not immune to these challenges and has developed a set of tools that address many of these challenges. Previous work has outlined some of the infrastructural and software tools used to address data volume and velocity issues. One of the key challenges associated with big data is in the abilty to develop an understanding of the underlying structures and patterns of the data. Very often, such an understanding is required as a pre-requisite to the application of advanced analytics to the data. Further, big data sets often contain anomalies and errors which are difficult to know apriori. Researchers at MIT Lincoln Laboratory have developed a technique - Dimensional Data analysis - that allows big data analysts to quickly understand the overall structure of a big dataset, determine anomaies. This work provides details on an upcoming area - Dimensional Analysis and Big Data model fitting.**

Keywords*—*Big Data, Data Analytics, Dimensional Analysis

I. Introduction

Ther is a growing problem within the scientific community in collecting and analyzing large amounts of data. The growing gap between data and user's capabilities to process this data calls for innovative tools that address challenges faced by the 3 V's of big data - volume, velocity and variety. The 3 V's provide a guide to the largest outstanding challenges associated with working with big data systems. Big data volume stresses the storage, memory and compute capacity of a computing system and requires access to a computing cloud. The velocity of big data velocity stresses the rate at which data can be absorbed and meaningful answers produced. Big data variety makes it difficult to develop algorithms and tools that can address that large variety in input data.

The Massachusetts Institute of Technology, Lincoln Laboratory works with a large variety of data sources such as open source data, Intelligence documents, social media, web logs, health records. At MIT Lincoln Laboratory, these data sources have varying velocity and volume. In order to alleviate some of these challenges, we are active in researching tools that can help alleviate the 3 V's of big data.

This work is sponsored by the Assistant Secretary of Defense for Research and Engineering under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, recommendations and conclusions are those of the authors and are not necessarily endorsed by the United States Government

In order to address the challenge of big data volume, MIT Lincoln Laboratory created the MIT Supercloud infrastructure [1] To address big data velocity concerns, MIT Lincoln Laboratory worked with various U.S. government agencies to develop the Common Big Data Architecture. Finally to address big data variety problems, MIT Lincoln Laboratory developed the D4M techology and schema [2].

While these techniques and technologies continue to evolve with the exponential increase in each of the V's of big data, analysts who work with data to extract meaningful knowledge have realized that certain low level paramemters that describe big data can be an important first step in an analysis pipeline. Some of the common big data analysis we see is in the application of complex techniques such as machine learning on a given dataset. Very often, as a first step, it is necesary to "clean" the data using techniques such as dimensionality reduction techniques such as Random Projections [3] or sketching [4]. Prior to such techniques, it is often necessary to obtain a coherent understanding of the data set which may include dimensions such as users, functions, etc. Such an understanding can also provide insight into any weaknesses that may be present in the data set. Further, detailed analysis of each data set is required to determine any internal patterns that may exist. We believe that a general purpose process for analyzing big data should consist of the following steps:

1) Learn about data structure
2) Determine background model of big data
3) Use data structure and background model for feature extraction or dimensionality reduction or noise removal
4) Perform advanced techniques such as machine learning
5) Use data exploration techniques such as visual analytics

These steps can be adapted to a wide variety of data and borrows heavily from the processes and tools developed for the signal processing commumity. The first two steps in this process provide a high level view of a given data set - a very important step to ensure that data inconsistencies are known.

Dimensional Analysis is a technique to learn about data structure and should be used as a first step with a new or unknown big data set. This technique can be used to gain an understanding of corpus structure, important dimensions, and data corruptions if present. This technique is called Dimensional Analysis for Big Data.

The article is structured as follows. Section II describes the 3 V's of big data and MIT Lincoln Laboratory technologies.

Section 2 describes a big data pipeline, Section III-C provides a detailed look at the D4M tool and schema. Section IV provides the mathematical and practical aspects of dimensional analysis. We provide two application examples in Section V. Finally, we conclude and discuss future directions in VI.

II. Challenges Associated With Big Data

The challenges associated with big data are commonly referred to as the 3 V's of Big Data - Variety, Volume and Velocity [5]. These 3 V's continue to rise at unprecedented rates. For example, in [6], the authors describe the difference between data being generated for storage and the world's capability to process (compute) this data. This difference has influenced the movement towards cloud computing which offer centralized large scale computing, storage and communication networks.

Choosing the right cloud is problem specific. Currently, there are four multi-billion dollar ecosystems that dominate the cloud computing environment: enterprise clouds, big data clouds, SQL database clouds, and supercomputing clouds. Each cloud ecosystem has its own hardware, software, conferences, and business markets. The broad nature of business big data challenges make it unlikely that one cloud ecosystem can meet its needs and solutions are likely to require the tools and techniques from more than one cloud ecosystem. The MIT Supercloud was developed to address this challenge. To our knowledge, the MIT SuperCloud is the only deployed cloud system that allows all four ecosystems to co-exist without sacrificing performance or functionality.

Big data velocity stresses the rate at which data can be absorbed and meaningful answers produced. Led by the NSA, a Common Big Data Architecture (CBDA), Figure 1 was developed for the U.S. government based on the Google Big Table NoSQL approach and is now in wide use. MIT Lincoln Laboratory played a leading role in developing the CBDA and is a leader in adapting the CBDA to a variety of big data challenges. The CBDA consists of three architectural abstractions - data, system and users. The CBDA describes the flow of information within such systems as well as the variety of users, data sources, and system requirements. At the center of the CBDA is the NSA developed Accumulo database [7] which has demonstrated high performance capabilities (capable of millions of entries/second) in a variety of applications [8].

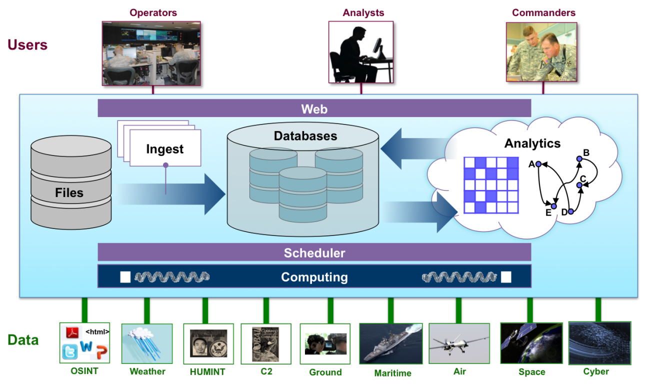


Fig. 1. The CBDA is designed for systems that require high throughput computing, storage and retrieval

Big data variety challenges the tools and algorithms developed to process big data sets. This may present the largest challenge and greatest opportunity for advancing the scientific community in the era of big data. The promise of big data is the ability to correlate diverse and heterogeneous data sources to reduce the time to insight. For example, data collected to support research in earth and social sciences can be correlated but come in a wide variety of formats. Correlating this data requires putting each format into a common frame of reference so that like entities can be compared. Historically, this process has involved significant human intervention via ontologies, data formats, and one-off database schemas. In the case of big data, this manual process is prohibitive and must be done via automated processes. The centerpiece of the CBDA is the NSA developed Apache Accumulo database and the MIT Lincoln Laboratory developed D4M Schema [2]. The D4M scheme allows vast quantities of highly diverse data (text reports, computer network logs, and social media data) to automatically be ingested into a simple common schema that allows every unique element to be quickly queried and correlated.

III. MIT Lincoln Laboratory Technologies

MIT Lincoln Laboratory has developed a set of tools to address the challenges associated with big data. These tools are used to further the development of the dimensional analysis discussed in this article.

A. Big Data Pipeline

At Lincoln Laboratory, we have taken a systems engineering approach in dveloping a general purpose five step pipeline which can be adapted for a variety of heterogeneus big data problems. The pipeline developed is shown in Figure 2 and has already been applied to health care, social media, defense applications, intelligence reports, building management systems, etc.

The generalized five step process was created after observing numerous big data systems in which the following steps were performed (application specific names may be different, but the goal is the same):

1) Raw Data Aquisition: Retrieve raw data from external sensors or sources.
2) Parse Data: Raw data is often in a format which needs to be parsed. Store results on distributed filesystem.
3) Ingest Data: If using a database, ingest parsed data into database.
4) Query or Scan for Data: Use either database or filesystem to find information.
5) Analyze Data: Perform complex analytics, visualizations, etc. for knowledge discovery.

B. Associative Arrays

Associative Arrays form the building blocks of data type used in big data databases. Associations between multidimensional entities (tuples) using number/string keys and number/string values can be stored in data structures called associative arrays. An associative array $\mathbf{A}$ with possible keys $\{k_1, k_2, ..., k_d\}$, denoted as $\mathbf{A}(\mathbf{k})$ is a partial function that maps two spaces S^d and S:

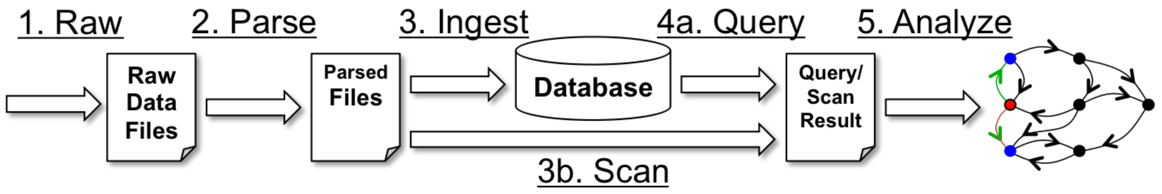


Fig. 2. General Steps involved in creating a big data system

$$A(k) : S^d \to S \; k = (k^1, ..., k^d) \\ k^i \in S^i$$

Where $A(k)$ is a partial function from d keys to 1 value where:

$$A(k_i) = \begin{cases} v_i \\ \phi & otherwise \end{cases}$$

Associative arrays also provide a natural mapping between the sparse matrix representation and graphical representation of the data.

C. D4M & Schema

The dynamic distributed dimensional data model (D4M) is used within the CBDA to support the database and computaitonal system. D4M was developed at MIT Lincoln Laboratory [2] to support prototyping of analytic solutions for big data applications. D4M applies the concepts of linear algebra and signal procesing to data bases through associative arrays; provides a data schema capable of representing most data; and provides a low barrier to entry through a computing API implemented in MATLAB and GNU Octave.

The D4M 2.0 schema [9], provides a four table solution that can be sued to represent most data values. The schema was invented at MIT Lincoln Laboratory and is at the center for the CBDA. The four table solution allows diverse data to be stored in a simple common format with just a handful of tables that can automatically be generated from the data with no human intervention. The schema allows every unique entity in the database to be looked up quickly. In addition, since all data uses the same schema, it is easy to correlate data across diverse data sources. Finally, since the data appears as a giant sparse matrix, linear algebraic signal processing techniques can be readily applied to the data. Consider the schema described in Figure 3.

From figure 3, each data entry is exploded into a sparse matrix where each unique column-value pair is a column. Once in sparse matrix form, the full machinery of linear algebraic graph processing [10] and detection theory can be applied. For example, multiplying two sparse associative arrays correlates the two.

IV. Dimensional Analysis

Dimensional Analysis provides a principled way to develop a coherent understanding of underlying data structures, data inconsistencies, data patterns, data formatting etc. Over time, a database may develop inconsistencies or errors, often due to a variety of reasons. Often, one wishes to perform advanced analytics on a database, looking for small artifacts in the data which may be of interest. In this section, we define the elements of a database in formal terminology.

A database can be represented by an ideal model that is described by a sum of sparse associative arrays, Figure 4. Consider a database $\underline{\mathbf{E}}$ represented as the sum of sparse associative arrays E_i:

$$\mathbf{E} = \sum_{1}^{n} E_i \tag{1}$$

Where i corresponds to different entities that comprise of the n entities of $\underline{\mathbf{E}}$. Each E_i has the following properties/definitions:

- N is the number of rows in the whole database (number of rows in associative array $\underline{\mathbf{E}}$)
- N_i is the number of rows in database entity i with atleaset one value (number of rows in associative array E_i).
- M_i is the number of unique values in database column i (number of columns in associative array E_i).
- V_i is the number of values in database column i (number of non zero values in associative array E_i).

Correspondingly, the gloabal sums are as follows:

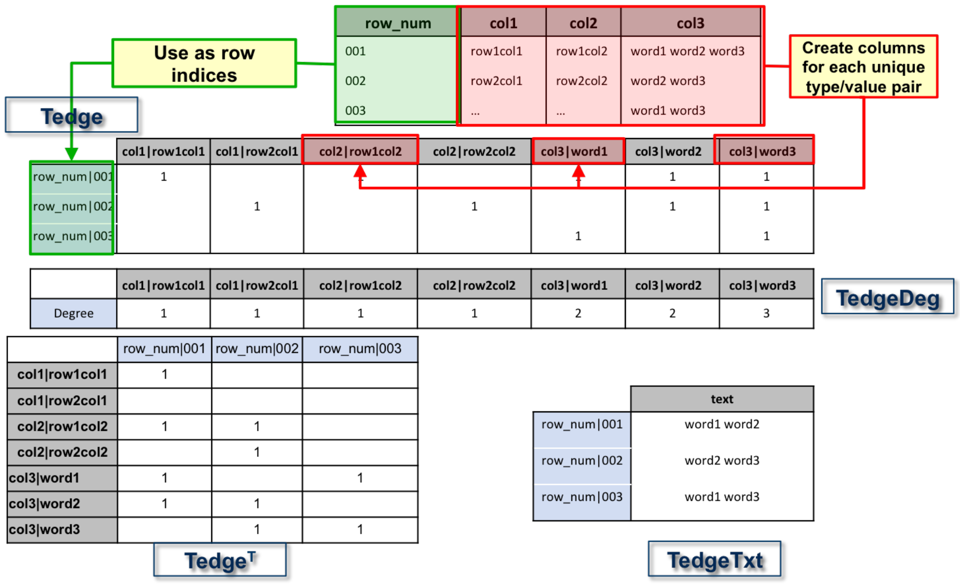
$$\begin{aligned} N &\leq \sum_i N_i, \; \forall i \\ M &= \sum_i M_i \; \forall i \\ V &= \sum_i V_i \; \forall i \end{aligned} \tag{2}$$

where N, M, and V correspond to the number of rows, columns and values in database $\underline{\mathbf{E}}$ respectively.

Each of the sub associative arrays (E_i)) can be typed as an *ideal* or *vestigial* array. The *ideal* and *vestigial* arrays are:

- Identity (**I**): Sub Associative Array E_i in which the number of rows and columns are of the same order:

$$N_i \sim M_i$$



row_num	col1	col2	col3
001	row1col1	row1col2	word1 word2 word3
002	row2col1	row2col2	word2 word3
003	...	...	word1 word3

	col1|row1col1	col1|row2col1	col2|row1col2	col2|row2col2	col3|word1	col3|word2	col3|word3
row_num|001	1					1	1
row_num|002		1		1		1	1
row_num|003					1		1

	col1|row1col1	col1|row2col1	col2|row1col2	col2|row2col2	col3|word1	col3|word2	col3|word3
Degree	1	1	1	1	2	2	3

	row_num|001	row_num|002	row_num|003
col1|row1col1	1		
col1|row2col1			
col2|row1col2	1	1	
col2|row2col2		1	
col3|word1	1		1
col3|word2	1	1	
col3|word3		1	1

	text
row_num|001	word1 word2
row_num|002	word2 word3
row_num|003	word1 word3

Fig. 3. D4M Schema typically consists of four tables in the NSA developed Apache Accumulo database. The majority of the data is stored in a primary table and its transpose that represent the data as a large sparse matrix (Tedge and Tedge^T. These tables allow every unique string in the database to be accessed quickly. In addition, there are sum tables (TedgeDeg) and raw data tables (TedgeTxt) that allow efficient query planning and allow the data to be viewed in its original context.

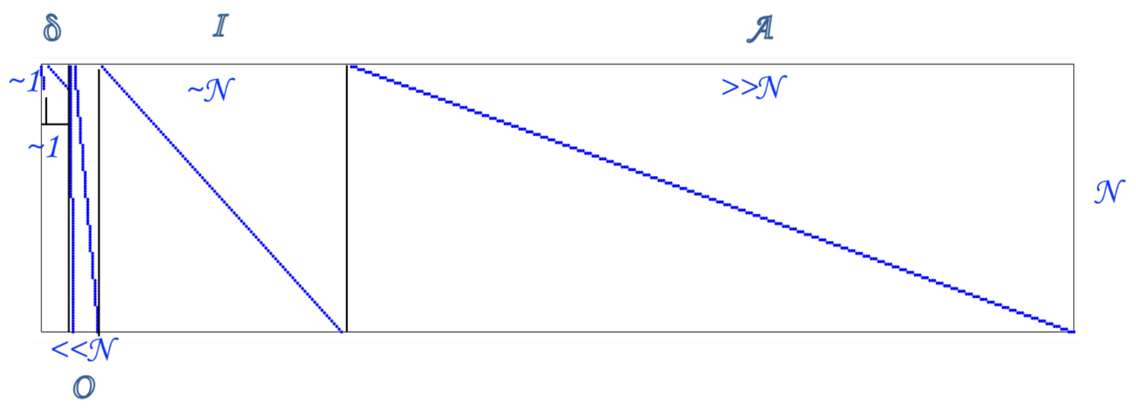


Fig. 4. A database can be represented as the sum (concatenation) of a series of sub associative arrays that correspon to different ideal or vestigial arrays

- Authoritive (**A**): Sub Associative Array E_i in which the number of rows is significantly smaller than the number of columns:

$$N_i \ll M_i$$

- Organizational (**O**): Sub Associative Array E_i in which the number of rows is significantly greater than the number of columns:

$$N_i \gg M_i$$

- Vestigial (δ): Sub Associative Array E_i in which the number of rows and columns are significantly small

$$N_i \sim 1$$
$$M_i \sim 1$$

By comparing a given sub associative array to the ideal types described above, it is possible to learn about the structure of the given data set.

A. *Performing dimensional analysis*

Consider a database $\underline{\mathbf{E}}$. In a real system, $\underline{\mathbf{E}}$ is a large sparse associative array representation of all the data in a database using the schema described in the previous sections. For example, $\underline{\mathbf{E}}$ could be a database consisting of health care data, intelligence reports, building management systems, social media, etc. Suppose that $\underline{\mathbf{E}}$ is made up of k entities, such that:

$$\mathbf{E} = \sum_{1}^{k} E_i$$

In a real database, these entities typically relate to various dimensions in the dataset. For example, entities may be time stamp, username, building id number, etc. Using an algorithm such as the following, one can perform dimensional analysis on a given dataset:

```
Data: Database represented by sparse associative array
      E
Result: Dimensions of sub associative arrays
        corresponding to entities
for entity i in k do
    read sub associative array $E_i \in E$;
    if number of rows in $E_i \geq 1$ then
        number of rows in $E_i$ = $N_i$;
        number of unique columns in $E_i$=$M_i$;
        number of values in $E_i$=$V_i$;
    else
        go to next entity;
    end
end
```

Algorithm 1: Dimensional Analysis Algorithm

Using the algorithm above, let the dimensions of each sub associative array (E_i) be contained in the 3-tuple (N_i, M_i, V_i) corresponding to the number of rows, columns and values in each sub associative arrays as described previously.

B. Using Dimensional Analysis results

Once the tuples corresponding to each entity is collected for a database E, one can compare the dimensions with the *ideal* and *vestigial* arrays described in section to determine the structural model for each entity.

Once the structural models are determined, it is possible to highlight interesting patterns, anomolies, formatting, and inconsistencies. For example:

- Authoritative (**A**): Important entity values (such as usernames, words, etc.) are highlighted by:

$$E_i * 1_{Nx1} > 1$$
$$1_{1xN} * E_i > 1$$

- Identity (**I**): Misconfigured or non non standard entity values are highlighted by

$$E_i * 1_{Nx1} > 1$$
$$1_{1xN} * E_i > 1$$
$$I_{NxN} - E_i \neq 0_{NxN}$$

- Organizational (**O**): The mapping structure of sub associative array is highlighed by counts and correlations in which

$$E_i * 1_{Nx1} \gg 1$$
$$E_i^T * E_j >> 1$$
$$1_{1xN} * E_i = 1$$

- Vestigial (δ): Erroroneous or misconfigured entries can typically be determined by inspecting E_i

V. Application Examples

In this section, we provide two example data sets and the results obtained through dimensional analysis. This section is meant to illustrate the concepts described before.

A. Twitter Dataset

Social media analysis is a growing area of interest in the big data community. Very often, large amounts of data is collected through a variety of data generation processes and it is necessary to learn about the low level structural information behind such data. Twitter is a microblog that allow up to 140 character "tweets" by a registered user. Each tweet is published by Twitter and is available via a publically acessible API. Many tweets contain geo-tagged information if enabled by the user. A prototype twitter dataset containing 2.02 million tweets was used for the dimensional analysis.

1) Dimensional Analysis Procedure: The process outlined in the previous section were used to perform dimensional analysis on a set of Twitter data with the intent of finding any anomolies, special accounts, etc. The database consists of 2.02 million rows and values distritubed across 10 different entities:

1) lat: The latitude when tweet was generationed
2) lon: The longitude when tweet was generated
3) latlon: A mertonized latitude and longitude ot tweet
4) place: (optional) The place ID of tweet
5) retweetID: (optional) The retweet ID given by Twitter
6) reuserID: (optional) The user ID of original tweet that was retweeted
7) time: Time stamp when tweet was generated
8) userID: User ID of user who generated tweet
9) User: Username of user who generated tweet
10) Word: Unigram of tweet payload (text)

The associative array representation of the Twitter corpus is shown in Figure 5. The 10 "dimensions" of data that make up the full data set are also shown.

2) Dimensional Analysis Results: Dimensional analysis of the dataset can be performed by performing Algorithm 1 on each of the entities (i) in k possible entities. For example, E_7 = E_{Time} which is the associative array in which all of the column keys correspond to time stamps. Thus, the triple ($N_{Time}, V_{Time}, M_{Time}$) is the number of entries with a time stamp, number of time stamp entries in the corpus, and number of unique time stamp values respectively. Peforming Algorithm 1 on each of the 10 entities yields the results described in Table 1.

Using the definitions defined in Section XX, we can quickly determine important characters.

For example, to find the most popular users, we can look at the difference where $E_{user} * 1_{Nx1} > 1$. Using D4M, this computation can easily be performed with Associative Arrays to yield the most popular users. Performing this analysis on the full 2.02 million tweet dataset represented by an associative array **E**:

```
% Extract Associative Array Euser
 >>Euser = E(:,StartsWith('user|,'));
%Add up count of all users
 >>Acommon = sum(Euser, 1);
%Display most common users
 >>display(Acommon>150);
   (1,user|SFBayRoadAlerts)      258
   (1,user|akhbarhurra)      177
   (1,user|attir_midzi)      159
```

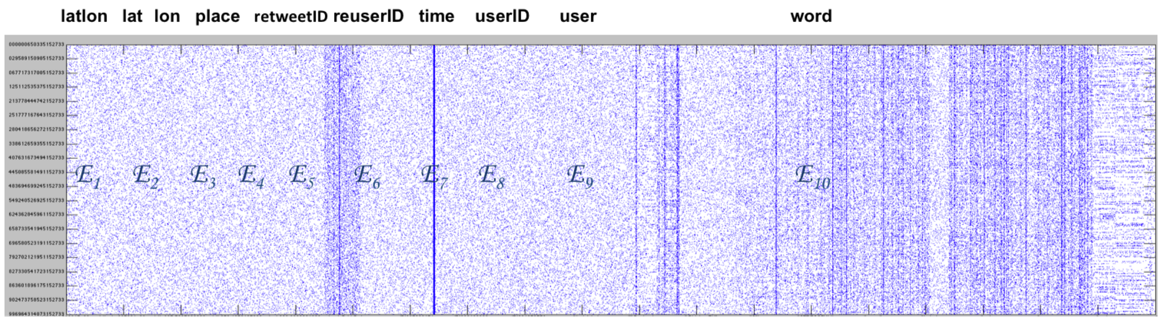


Fig. 5. Associative Array representation of Twitter data. $E_1, E_2, ..., E_{10}$ represent the concatenated associative arrays E_i that constitute all the elements in the full dataset.

Entity	N_i	V_i	M_i	Structure Type
latlon	1624984	1625197	1506465	Identity
lat	1624984	1625192	1504469	Identity
lon	1625061	1625725	1504619	Identity
place	1741337	1741516	1504619	Identity
retweetID	636455	636644	627163	Identity
reuserID	720624	722148	676616	Identity
time	2020000	2020000	35176	Organization
userID	2020000	2020000	1711141	Identity
user	2020000	2020000	1711143	Identity
word	1976746	17180314	7838862	Authority

TABLE I. DIMENSIONAL ANALYSIS PERFORMED ON 2.02 MILLION TWEETS

```
  (1,user|verkehr_bw)      300
```

The results above indicate that there are 4 users who have greater than 150 tweets in the dataset. Such analyses can easily be extended to other parameters of interest.

B. HPC Scheduler Log Files

LLSuperCloud uses the Sun Grid Engine scheduler for dispatching jobs. For each job that is finished, an accounting record is written to an accounting file. These records can be used in the future to generate statistics about accounts, system usage, etc. Each line in the accounting file represents an individual job that has completed.

1) Dimensional Analysis Procedure: The process outlined in the previous section were used to perform dimensional analysis on a set of SGE Accounting data with the intent of finding any anomolies, special accounts, etc. The database consists of approximately 11.5 million entries with 27 entities each. A detailed description of the entities in the SGE accounting file can be found at: [11].

The associative array representation of the SGE corpus is shown in Figure 6. The 27 "dimensions" of data that make up the full data set are shown in figure 6.

2) Dimensional Analysis Results: After performing dimensional analysis on the SGE corpus, the results are tallied for inspection. A subset of the results is shown in Table II. It is interesting to note that there are many accounting file entries that are not collected and have only defaul values. For example, the field "defaultdepartment" contains only one unique value in the entire dataset - "default". For an individual wishing to perform more advanced analytics on the dataset, this is an important result and can be used to reduce the dimensionality of each data point.

```
% Extract Associative Array Euser
 >>Ejobname = E(:,StartsWith('job_name|,'));
%Add up count of all users
 >>Acommon = sum(Ejobname, 1);
%Display most common users
 >>display(Acommon>1000000);
  (1,job_name|rolling_pipeline.sh)      2762791
  (1,job_name|run_blast.sh)      1256422
  (1,job_name|run_blast_parser.sh)      1162522
```

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a big data analytic called dimensional analysis. Using a technique such as dimensional analysis, a researcher can learn a great deal about the hidden patterns, structural characteristics and possible errors of a large unknown dataset. Dimensional analysis consists of representing a dataset using Associative Arrays and performing a comparison between the constituent Associative Arrays and ideal database arrays. Deviations from the ideal model can highlight important details or incorrect information.

We recommend that dimensional analysis form the first step of an analytic pipeline. The common next steps in an analytic pipeline such as background modeling, feature extraction, machine learning and visual analytics depend heavily on the quality of input data.

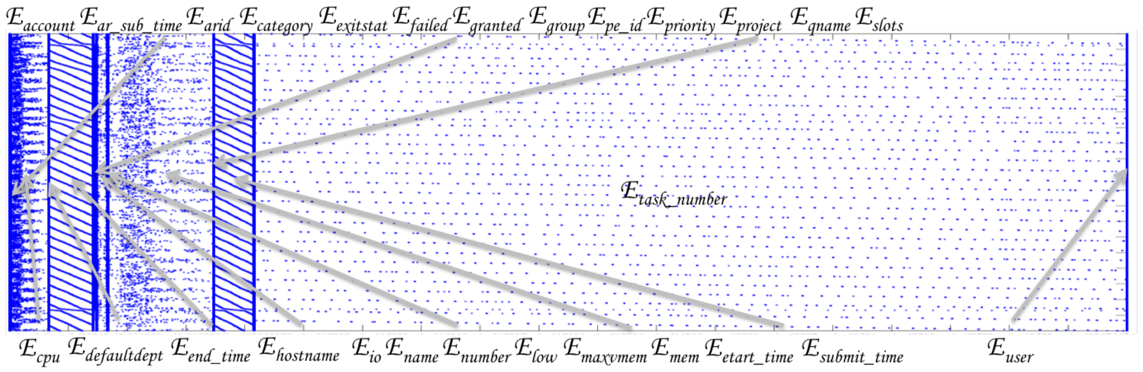


Fig. 6. Associative Array representation of SGE accounting data represent the concatenated associative arrays E_i that make up the full dataset.

Entity	N_i	V_i	M_i	Structure Type
Account	11446187	11446187	1	Vestigial
CPU Hours	11446187	11446187	2752964	Identity
Default Department	11446187	11446187	1	Vestigial
Job Name	11446187	11446187	90491	Organization
Job Number	11446187	11446187	485212	Identity
Memory Usage	11446187	11446187	5241559	Identity
Priority	11446187	11446187	1	Vestigial
Task Number	11446187	11446187	7491889	Identity
User Name	11446187	11446187	8388	Organization

TABLE II. DIMENSIONAL ANALYSIS PERFORMED ON APPROXIMATELY 11.5 MILLION SUN GRID ENGINE ACCOUNTING ENTRIES. ONLY SELECTED ENTRIES ARE SHOWN OF THE 27 TOTAL ENTRIES COLLECTED

Next steps to this work include developing an automated mechanism to perform background modeling of big datasets.

ACKNOWLEDGMENT

The authors would like to thank the LLGrid team at MIT Lincoln Laboratory for their support and expertise in setting up the computing environment LLSuperCloud.

REFERENCES

[1] A. I. Reuther, T. Currie, J. Kepner, H. G. Kim, A. McCabe, P. Michaleas, and N. Travinin, "Technology requirements for supporting on-demand interactive grid computing," in *Users Group Conference, 2005*, pp. 320–327, IEEE, 2005.

[2] J. Kepner, W. Arcand, W. Bergeron, N. Bliss, R. Bond, C. Byun, G. Condon, K. Gregson, M. Hubbell, J. Kurz, *et al.*, "Dynamic distributed dimensional data model (d4m) database and computation system," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 5349–5352, IEEE, 2012.

[3] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: applications to image and text data," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 245–250, ACM, 2001.

[4] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 604–613, ACM, 1998.

[5] D. Laney, "3d data management: Controlling data volume, velocity and variety," *META Group Research Note*, vol. 6, 2001.

[6] M. Hilbert and P. López, "The world's technological capacity to store, communicate, and compute information," *Science*, vol. 332, no. 6025, pp. 60–65, 2011.

[7] "Apache accumulo:," *https://accumulo.apache.org/*.

[8] C. Byun, W. Arcand, D. Bestor, B. Bergeron, M. Hubbell, J. Kepner, A. McCabe, P. Michaleas, J. Mullen, D. O'Gwynn, *et al.*, "Driving big data with big compute," in *High Performance Extreme Computing (HPEC), 2012 IEEE Conference on*, pp. 1–6, IEEE, 2012.

[9] J. Kepner, C. Anderson, W. Arcand, D. Bestor, B. Bergeron, C. Byun, M. Hubbell, P. Michaleas, J. Mullen, D. O'Gwynn, *et al.*, "D4m 2.0 schema: A general purpose high performance schema for the accumulo database," in *High Performance Extreme Computing Conference (HPEC), 2013 IEEE*, pp. 1–6, IEEE, 2013.

[10] J. Kepner, D. Ricke, and D. Hutchinson, "Taming biological big data with d4m," *Lincoln Laboratory Journal*, vol. 20, no. 1, 2013.

[11] "Ubuntu manpage: Sun grid engine accounting file format," *http://manpages.ubuntu.com/manpages/lucid/man5/sge_accounting.5.html*.