Big Data acquisition, preparation and analysis using Apache Software Foundation Projects

Gouri Ginde[1], *Rahul Aedula*[1], *Snehanshu Saha*[1], *Archana Mathur,*[1] *Sudeepa Roy Dey*[1],

Gambhire Swati Sampatrao[1], *BS Daya Sagar*[2]

[1]PESIT Bangalore South Campus, Bangalore, India

[2]Indian Statistical Institute, Bangalore, India

Contents

Big Data acquisition, preparation and analysis using Apache Software Foundation Projects

1. Apache Software Foundation (ASF) projects for Big Data

Challenges in Big Data analysis include data inconsistency, incompleteness, scalability, timeliness and data security. Most fundamental challenge is the existing computer architecture. For several decades the latency gap between multi-core CPUs and mechanical hard disks is growing every year, making the challenges of data-intensive computing harder to overcome (Hey, Tansley, & Tolle, 2009). A systematic and general approach to these problems with a scalable architecture is required. Most of the big data is unstructured or of complex structure, which is hard to represent in rows and columns. A good candidate for a large design space can efficiently solve the big data problem in different disciplines. The book chapter highlights two specific objectives: (1) To introduce an efficient model,SVD for complex computer experiments arising in Big data which can be used across different scientific disciplines and (2) To introduce optimization techniques and tools for handling big data problems. Different aspects of these two objectives will be discussed in later sections of this chapter.

Modern data-mining applications (Leskovec et al, 2011), often called "big-data" analysis, require us to manage and analyze immense amounts of data quickly. Some examples work with “irregular” structures and efficient solutions have been proposed using crowd-sourcing (Agarwal, Ravikumar, & Saha 2016a; Agarwal, Ravikumar & Saha, 2016b). However, many of these applications are endowed with extremely regular data and there is ample opportunity to exploit parallelism. A few examples are as follows:

- The ranking of Web pages by importance, which involves an iterated matrix-vector multiplication where the dimension is many billions.
- Searches in "friends" networks at social-networking sites, which involve graphs with hundreds of millions of nodes and many billions of edges
- The search for life on the planets outside the solar System, which involves data analytics on massive data volume generated from next-generation telescopes (Bora, 2016).

To deal with such applications, a new software stack has evolved. These programming systems are designed to inherit their parallelism not from a "supercomputer," but, from "computing clusters" defined as large collections of commodity hardware, including conventional processors ("compute nodes") connected by Ethernet cables or inexpensive switches. The software stack begins with a new form of file system, called a "distributed file system," which features much larger units than the disk blocks in a conventional operating system. Distributed file systems also provide replication of data or redundancy to protect against the frequent media failures that occur when data is distributed over thousands of low-cost compute nodes. On top of these file systems, many different higher-level programming systems have been developed. Central to the new software stack is a programming system called Map Reduce. Implementations of Map Reduce and many other projects from Apache Software Foundation (ASF) provide a few path breaking software for data intensive problem solving. ASF projects enable large-scale data operations using computing clusters efficiently in a manner that is tolerant of hardware failures during the computation.

Apache has been a powerful contributor to the open source ecosystem. ASF has been home to numerous important open source software projects from its inception in 1999 (Web API,2017), the all-volunteer Foundation oversees more than 350 leading open source projects, including Apache HTTP Server, the world's most popular web server software. Through the ASF's meritocratic process known as "The Apache Way," more than 620 individual members and 5,500 committee members successfully collaborate to develop freely available enterprise-grade software, benefiting millions of users worldwide: thousands of software solutions are distributed under the Apache License; and the community actively participates in ASF mailing lists, mentoring initiatives, and ApacheCon, the foundation's official user conference, training, and exposition. The ASF is a US 501(c)(3) charitable organization, funded by individual donations and corporate sponsors including Alibaba Cloud Computing, ARM, Bloomberg, Budget Direct, Cerner, Cloudera, Comcast, Confluent, Facebook, Google, Hortonworks, HP, Huawei, IBM, InMotion Hosting, iSigma, LeaseWeb, Microsoft, OPDi, PhoenixNAP, Pivotal, Private Internet Access, Produban, Red Hat, Serenata Flowers, WANdisco, and Yahoo. There are currently 300+ open source initiatives at the ASF. Of these 225 projects use Java. 36 are big data related projects and 12 are cloud related projects. The success story of ASF ranges from Geronimo, Tomcat to Hadoop, the distributed computing system that now serves as a lynch-pin of the big data realm.

Hadoop project is all the rage these days and is synonymous with big data, in which enterprises and web properties sift through reams of data to surface insights about customers and users. Hadoop provides an operating system for distributed computing. "If you want to run computations on hundreds of thousands of computers instead of just on one, Hadoop lets you do that," says

Doug Cutting, a primary contributor to Hadoop for several years. Hadoop originated from the Nutch Web software project in 2006. Companies like Cloudera and HortonWorks are building businesses around Hadoop.

Chapter overview

The chapter provides basic methods of big data gathering, curation and analysis with an example. Figure 1 shows the overall flow of the chapter. In the 2nd section, we will discuss the data acquisition methods and various methodologies of data gathering. The next section will explore the data curing and pre-processing methods in the "Big Data" paradigm. Finally, the last section will focus on data analysis. We will briefly introduce data visualization and Apache Foundation projects which may help in processing and representing big data.

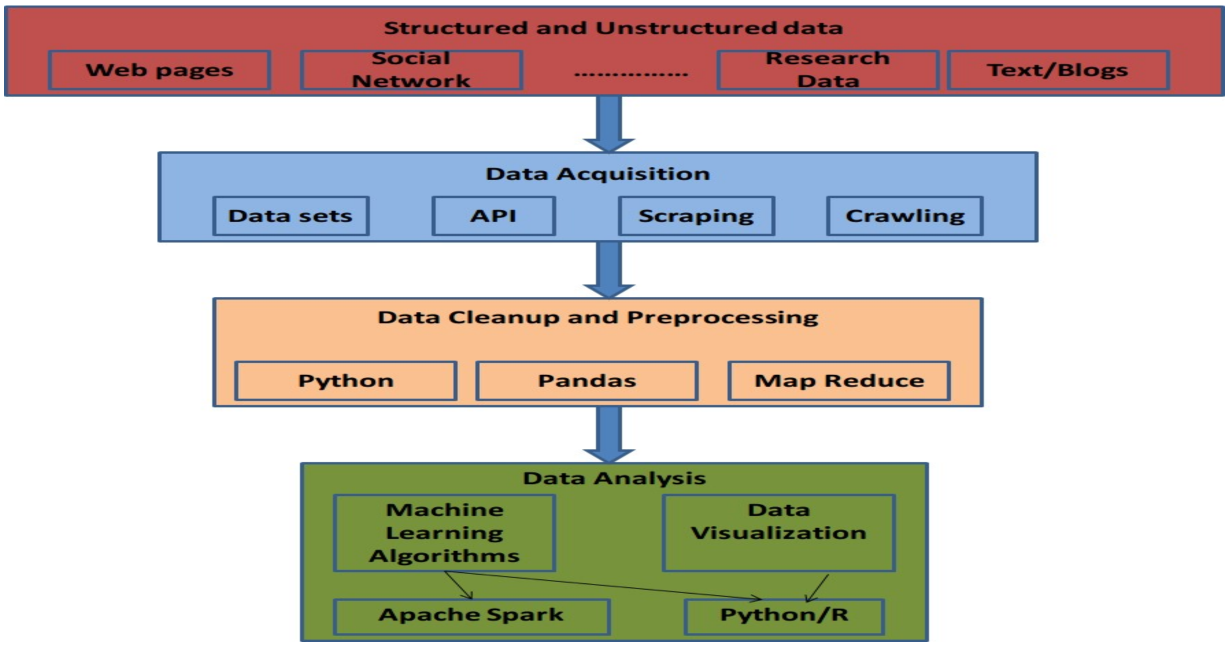


Figure 1: Overall flow of the chapter

2 Data acquisition

For any data related project, the critical step is to chart the blueprint of expected end result and the stages of reaching those goals through data tunnel. Data accumulation is the first milestone in that direction. Data collection (Shukla, 2014) is arguably as important as analyzing it to extrapolate results and form valid claims which may be generalized. It is a scientific pursuit; therefore, great care must be taken to ensure unbiased and representative sampling. There isn't much to analyze without data, , demanding careful observation of the techniques laid out to build a formidable corpus. There are various ways to harness data once the problem to be solved has been decided. Information can be described as structured, unstructured, or sometimes a mix of the two i.e. semi-structured. In a very general sense, structured data is anything that can be parsed by an algorithm. Common examples include JSON, CSV, and XML. Provided any structured data, we may design a piece of code to dissect the underlying format and easily produce useful results. As mining structured data is a deterministic process, it allows us to automate the parsing. This, in effect, lets us gather more input to feed to data analysis algorithms. Unstructured data is everything else and defined in a specified manner. Written languages such as English are often regarded as unstructured because of the difficulty in parsing a data model out of a natural sentence. In our search for good data, we will often find a mix of structured and unstructured text. This is called semi-structured text. This recipe will primarily focus on obtaining structured and semi-structured data from the following sources. However, this list is not at all exhaustive.

2.1 Freely available sources of data sets

There are a number of NGO's, Government websites and other places which host the data sets, available for download for free. Some examples are the following,

- http://pesitsouthscibase.org/datacenter provides scholarly article data sets that are mined from authentic websites such as ACM, IEEE, Springer etc; organized and fed to scientific computing exercises.
- https://aws.amazon.com/datasets/ Amazon public data sets on AWS provide a centralized repository of public data sets that can be seamlessly integrated into AWS cloud-based applications. AWS is hosting the public data sets at no charge for the community, and like all AWS services, users pay only for the compute and storage they use for their own applications.
- https://www.kaggle.com/datasets Kaggle releases data sets and provides a platform for predictive modeling and analytics competitions. On this website companies and researchers post their data in the form of data sets for free to all statisticians and data miners from all over the world to compete and produce the best models.
- http://data.worldbank.org/ A world bank data repository which is the source of

poverty and world development data. It regards itself as a free source that enables open access to data about development in countries around the globe.

2.2 Data collection through APIs (Application Programming Interface):

Many governments collect a lot of data, and some are now offering access to this data. The interfaces through which this data is typically made accessible are web APIs. Web APIs allow access to data, such as "budget, public works, crime, legal, and other agency data" by any developer in a convenient manner. The United States are one of the pioneers in allowing government data for public use through the portal, Data.gov, the central site for U.S. Government data (Wang, Wang & Alexander, 2015). Following are a few more examples of websites which provide API based data access

- https://developers.facebook.com/docs/graph-api
- https://dev.twitter.com/
- http://api.altmetric.com/

2.3 Web scraping

This is a website-specific data acquisition method, where a program is written to read the website's HTML page and decode the HTML tags of a web page to extract the data of our interest. Web scraping represents a very superficial node of web crawling and data source can be chosen as anything for the scraping task.

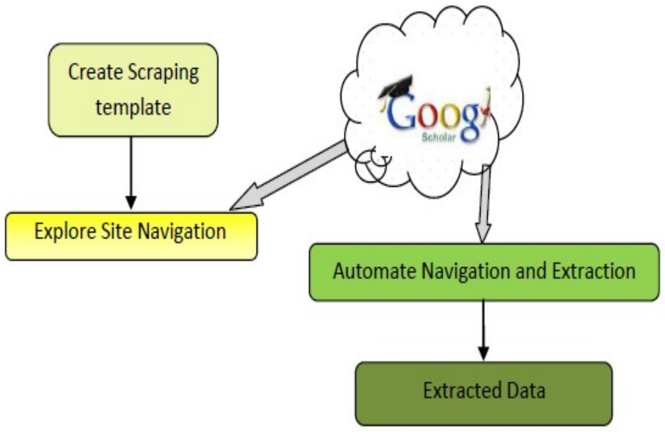


Figure 2 Web Scraping for the data acquisition from Google Scholar website

Figure 2 demonstrates the essential components of web scraping methodology (Ginde et al.,2016; Ginde et al.,2015). DOM Parsing is the philosophy which assists the system with retrieving element content created by client-side scripts utilizing undeniable web program controls, for example, the Internet Explorer browser or the Mozilla browser control. These programs control similar parsed web pages into a DOM tree, in light of which program can recover parts of the pages.

Create scraping template: Inspect Element is a developer's tool that allows viewing the HTML, CSS, and JavaScript that is currently on a web page. One may right click and select inspect element on practically every web page. This will pull up the developers console to view the HTML and CSS of the web page.

Explore site navigation: In order to explore and understand the website's navigation for dynamic URL formulation and data extraction, Beautiful Soul parser library package developed in Python can be used. Beautiful Soup parser is also called as elixir and tonic the screen-scrapers friend. It uses a pluggable XML or HTML parser to parse a possibly invalid document into a tree representation. It also provides methods and pythonic idioms that make it easy to navigate, search and modify the parse tree.

Automate navigation and extraction: Python is a scripting language which is easy to learn and equipped with powerful programming interfaces. Python interpreter library which is freely available in source and binary form for all major platforms.

2.4 Web crawling

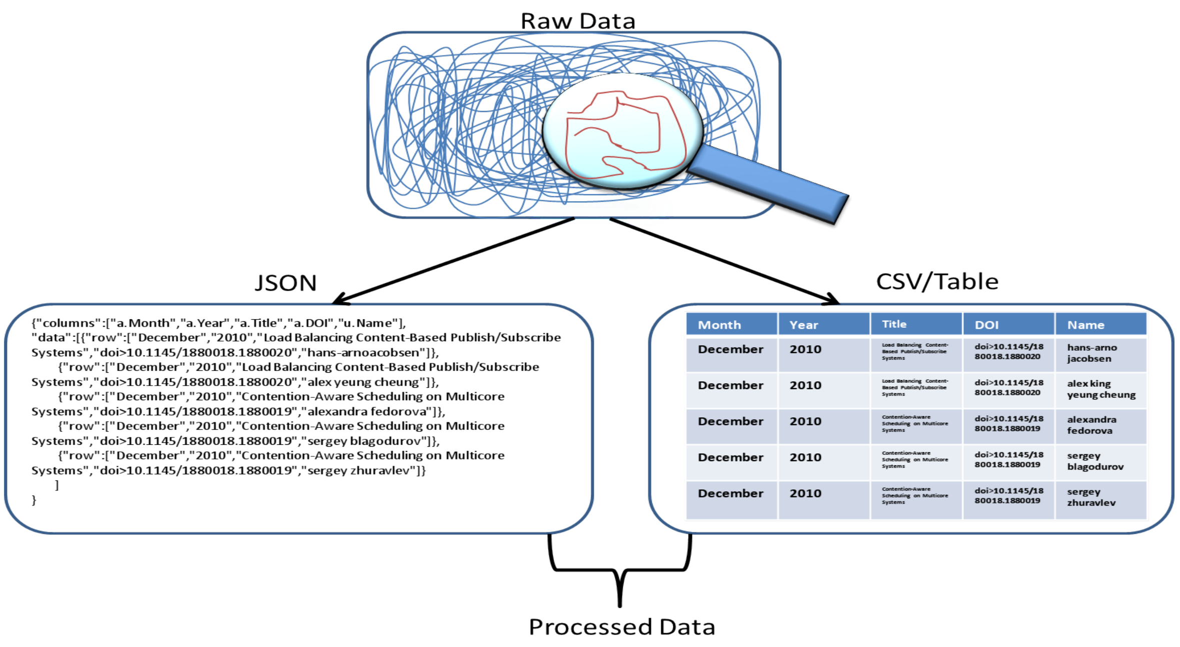
The internet is the only data source for web crawling and this task is similar to what Google performs. Web crawling is essentially what search engines do. It's all about viewing a page as a whole and indexing it. When a bot crawls a website, it goes through every page and every link, until the last line of the website, looking for any information. Web crawling is mostly harmless. Crawling usually refers to dealing with large data sets where one develops crawlers (or bots) which crawl to the deepest of the web pages. Nutch crawler is one such open source crawler which is widely used. One can use one or a combination of various data accumulation methods to gather useful data. This accumulated data needs to be tidied up for any further processing which necessitates the data preprocessing and cleanup on it.

3. Data pre-processing and cleanup

Data can be structured, unstructured or semi structured. Structured database management is accomplished in two stages. The first stage stores data in a schema based data storage, known as relational database. The stored data is queried via query based data retrieval in the second stage. In order to manage large scale structured data sets, data warehouses and data marts are known to be widely used approaches. Data mart handles data that belongs to a unit or operation of an organization. Conversely, data warehouse is not limited to a particular department and it represents the database of a complete organization. Unstructured data is generally non-relational. Special type of databases called "non-relational NoSQL: Not only SQL" are widely used in storing massive scale unstructured data. NoSQL databases focus on the scalability of data storage with high-performance. They are also schema less, which enables applications to quickly modify the structure of data without any need of write operation queries on multiple tables as opposed to relational schema based databases. Irrespective of the type of data, pre-processing of the data is necessary before it is stored. Data in its raw form, called raw data, might not always be lucid for interpretation. It is not always possible to directly start analyzing data in its native construct. Data curation/ pre-processing are aimed at data discovery and retrieval, data quality assurance, value addition, reuse and preservation over time (Chen & Zhang, 2014). Raw data may contain a lot of inconsistencies, duplicate entries, and redundancies. Data pre-processing and clean-up is performed to eliminate all of these and make the data more palpable for the analysis and transformation. Data preparation is not just a first step, but must be repeated many times over the course of analysis as new problems come to light or new data is collected. It is an iterative process. In case of ready data sets there might be inconsistencies such as incomplete fields/parameter values or special characters etc. In the case of scraped data sets, the data might be in pure text format, comma separated values format or JSON format. Such data might have just long strings which further need text processing, computation of parameters based on the various numerical values, a mere sorting of the data or all of these to extract the meaningful information for further analysis. Pre-processing steps might vary from one data set to another data set. The complexity of any such data pre-processing operation manifolds with the increase in volume of raw data. Generally, massive scale data in its crude form is voluminous and can be very messy to work with. It is essential to use the right kind of data tools to perform such pre-processing operations on a massive scale. The most challenging of these problems is the magnitude of the data. Most conventional method is to use a higher specification system to contain all of the data in a single computing system. The drawback with this approach is the exponential increase in cost and complexity. Alternatively, we can resort to cluster computation, which provides a foolproof solution for massive scale computation. Cluster-based frameworks like Apache Hadoop Map Reduce are the next biggest step to achieve such high efficiency. The use of clusters significantly improves the ease to process voluminous data, reduces the overhead of pre-processing on a single system, instead utilizes a collective set of machines to perform clean-up in a comparatively short time. Voluminous task are divided into several smaller tasks which are then assigned to machines of the cluster to produce the collective final output at the end. Python and R are the other preferred tools used to perform clean-up. These tools don't have a significant use of clusters for load reduction but their use of in-memory computation and an eclectic collection of libraries make it very favorable to handle the data at a reasonable velocity. Tools like Apache Spark try to integrate scalability and in-memory computation features into a single framework to maximize efficiency. Appendix 1 provides additional information related to Apache spark's architecture. The next concern is the removal of noise in the data. Noise removal refers to the deletion of any unwanted part of the data to prepare it for further analysis and decision making. Almost all the languages discussed in this chapter are capable of performing data pre-processing task.

3.1 Need for Hadoop Map Reduce and other languages for big data pre-processing

Data in its raw form does not provide any value. As shown in figure 3, raw data needs to be processed and transformed into a readable structure which can be further processed. Accomplishing this by using some algorithms to draw inferences is a daunting task. Preparing a volume of data to run through algorithms is the most time-consuming task of the complete data processing tunnel. For pre-processing tasks, we can use batch processing frameworks and languages to ease the work.



```
{"columns":["a.Month","a.Year","a.Title","a.DOI","u.Name"],
"data":[{"row":["December","2010","Load Balancing Content-Based Publish/Subscribe
Systems","doi>10.1145/1880018.1880020","hans-arnoacobsen"]},
    {"row":["December","2010","Load Balancing Content-Based Publish/Subscribe
Systems","doi>10.1145/1880018.1880020","alex yeung cheung"]},
    {"row":["December","2010","Contention-Aware Scheduling on Multicore
Systems","doi>10.1145/1880018.1880019","alexandra fedorova"]},
    {"row":["December","2010","Contention-Aware Scheduling on Multicore
Systems","doi>10.1145/1880018.1880019","sergey blagodurov"]},
    {"row":["December","2010","Contention-Aware Scheduling on Multicore
Systems","doi>10.1145/1880018.1880019","sergey zhuravlev"]}
    ]
}
```

Month	Year	Title	DOI	Name
December	2010	Load Balancing Content-Based Publish/Subscribe Systems	doi>10.1145/1880018.1880020	hans-arno jacobsen
December	2010	Load Balancing Content-Based Publish/Subscribe Systems	doi>10.1145/1880018.1880020	alex king yeung cheung
December	2010	Contention-Aware Scheduling on Multicore Systems	doi>10.1145/1880018.1880019	alexandra fedorova
December	2010	Contention-Aware Scheduling on Multicore Systems	doi>10.1145/1880018.1880019	sergey blagodurov
December	2010	Contention-Aware Scheduling on Multicore Systems	doi>10.1145/1880018.1880019	sergey zhuravlev

Figure 3 Transformation from raw data to processed data

When to choose Hadoop over Python, R?

Data processing method needs close scrutiny of various aspects, such as size of the data, size of RAM, language of choice, etc. If data is ranging in few GB's (medium scale) with RAM size 4 or 8 GB, preference should be given to scripting languages such as R and Python. Moreover, if this data is structured i.e. in CSV format, tabular format or any format that can be converted to CSV, then R, Python or even Java can be agreeably used. For advanced data analytics of medium scale data, one can use python's machine learning library Scikit-learn or R. For data visualization, online tools such as Tableu, Plotly can give quick visual results for medium scale data.

Conversely, when data is unstructured and ranges in a few hundreds of GB's or Tera Bytes, Hexa Bytes and a few Peta Bytes, distributed computing methodology such as Hadoop Map Reduce or PIG can be used, as this volume of data cannot be contained in standard RAM size for processing. Hadoop uses distributed computing methodology with clustered and fault tolerant storage mechanism. This provides a robust solution for batch mode of data processing. It is important to understand the advantages and disadvantages of various languages for efficient processing of the medium and massive scale data.

3.2 Comparison of Hadoop Map Reduce, Python and R

Hadoop Map Reduce is a programming mechanism which is used for batch processing of large data sets, where the data is split into blocks and is distributed to all the available nodes for parallel computation. This proves to be an effective model in solving the big data problem because of its ability to utilize the Hadoop Distributed File System (HDFS). The Hadoop file system is a distributed file system which stores data in data nodes that are scattered across a cluster. The meta-data is maintained by the name node which is also a master node. To increase fault tolerance a single data block is replicated 3 times by default. This feature provides robust fault tolerance as the data loss is immediately handled by one of the replicas. The Map Reduce splits the tasks into mapper and reducer. The mapper first maps all the similar elements of the data set and sends it to a temporary data file and the reducer reads from this file and performs the computations on these separated data sets. This use of parallelization makes Hadoop Map Reduce more effective than most of the other tools in terms of how the data is handled and also shows the efficiency on large volumes of data.

Python is a general purpose programming language created for easy readability. Python's biggest strength lies in its eclectic collection of libraries to perform various tasks such as data extraction, text processing and machine learning algorithms. It is widely supported programming language today. In the context of data analysis, Python provides a lot of built in packages and frameworks such as Pandas, Scipy, Scikit-learn and Numpy. These make data pre-processing and analysis an easy task but scalability poses to be a serious issue. Python's performance over a large batch processing doesn't make it a favorable tool. Since most of the analysis tasks and computations take place in-memory, its performance drastically degrades compared to distributed system as data is massively scaled.

R language was designed in a statistical point of view. Its primary focus is to bolster mathematical modeling and to help create analytical prototypes. R provides a less convoluted route to access many machine learning and mathematical functions. However from the data analysis and data processing prospective, R is comparatively slow in performance to its rival Python and also the aforementioned Apache Hadoop. The use of a cluster to achieve parallelization in Hadoop and the faster in-memory computation of Python make them much more dominant for such data related tasks.

3.3 Example of cleansing methods and routines

Data cleansing is a highly iterative process which completely depend on the type of the data at hand. In this section we will focus on text data which is in the JSON structure. We will first provide the generic cleansing operations on a test data and then provide Hadoop Map Reduce based solution example for the same data set. This exercise will help understand how the data cleansing prototype, developed for a small data set, scales up for batch processing of voluminous data.

Table 1 shows the sample data scraped using web scraping methodology. This sample contains the affiliations of the authors of the various scholastic articles, which belong to a journal. Some authors have multiple affiliations. The data cleansing on this data set involves the following operations.

- Elimination of extra spaces such as tab, newline etc.
- Strip/truncate the unwanted characters suffixed and post fixed such as '# ', ';', etc.
- Encode the text using UTF-8 encoder just to be careful about the Unicode characters, if any.
- Process Affiliation string (the address string) and disintegrate into University and Country names. All these operations can be performed iteratively or in a single iteration. table 2 is the output after processing.

Author : Beth A. Reid
Affiliation : Institute of Space Sciences (CSIC-IEEC), UAB, Barcelona 08193, Spain

Author: Daniel J. Eisenstein
Affiliation: Steward Observatory, University of Arizona, 933 N. Cherry Ave., Tucson, AZ 85121, USA

Author: David N. Spergel
Affiliation: Department of Astrophysical Sciences, Princeton University, Princeton, NJ 08544, USA

Author: Ramin A. Skibba
Affiliation: Max-Planck-Institute for Astronomy, Knigstuhl 17, D-69117 Heidelberg, Germany

Table 1: Sample data

Author: Beth A. Reid
University: Institute of Space Sciences
Country: Spain

Author: Daniel J. Eisenstein
University: University of Arizona
Country: USA

Author: David N. Spergel

Table 2: Output of content in table 1 after processing

Cleansing operations like these can be easily translated into Map Reduce jobs or PIG scripts. Above explained operations are mutually exclusive operations, hence the massively scaled data can be easily run through these algorithms in batches.

Table 3 is the Map Reduce pseudo code to get the above output for massive scale data

Note: In depth source code, instructions to load the data to HDFS and aggregation commands to demonstrate the final output are out of scope of this chapter. These can be looked up easily in the book, Hadoop: The Definitive Guide by Tom White.

```
map(String key, String value)
// key: Journal name
// value: Journal contents; All the article's authors and affiliation information
 for each sentence S in value
   EmitIntermediate(key, S)

reduce(String key, String values):
// key: some random number
// values: Author or Affiliation information
  if values contain 'Author' :
  result += TextOperation(values);
 else :
         result += TextOperation and UniversityInfo(v);
 Emit(AsString(result));
```

Table 3: Sample Map Reduce pseudo code for massive scale processing

When data is in the form of flat files or compressed files or in different formats, one has to give more effort to bring the data into a format/structure that can be understood by a programming/scripting language for further processing. Next subsections elaborate more on these challenges and the solutions offered by contenders of big data processing.

Loading data from flat files

Loading data with these respective tools differ significantly as Hadoop and Python interpret and analyze data in different ways. As mentioned earlier, Hadoop uses a cluster based approach and Python uses an in-memory approach to process data. The corresponding input formats are accordingly tailored to suit its processing methodology.

Hadoop in Java provides various categories of input format for structured data. It can process the data or files only when they are mounted on the Hadoop Distributed File System (HDFS). HDFS is a specialized file system, which is designed to exploit the cluster computing. Hadoop can process only flat files as of now. Apache POI and HSSF java libraries can be used to read and write to Excel files. JDHF5 can be used to read and write to HDF5 format of files. Official Hadoop support for Excel and HDF5 is yet to be provided.

Reading and writing to database using Hadoop Map Reduce could be tedious and cumbersome. Data warehousing also becomes difficult due to similar problems. Solution for these is to use Apache community developed Sqoop and Hive solutions. Sqoop is a tool built for Hadoop, which is used to transfer contents from a traditional Relational Database (RDBMS) such as MySQL, Oracle to a NoSQL type database such as Hbase, Hive etc. Sqoop works by spawning tasks on multiple data nodes to download various portions of the data in parallel. Each piece of data is replicated to ensure reliability and fault tolerance for parallel processing on a cluster. This can be easily accomplished by specifying a JDBC connection between RDBMS.

The Sqoop import command can be used to read data from an RDBMS and an export command to transfer data from Hadoop to an RDBMS. The parallel execution feature of Sqoop boosts the processing performance when database is voluminous. For example, table 4 shows the commands to import data from an RDBMS:

```
sqoop import-all-tables \
   -m 1 \
   --connect jdbc:mysql://quickstart:3306/mydata_db \
   --username=ABC \
   --password=ABC123 \
   --compression-codec=snappy \
   --as-parquetfile \
   --warehouse-dir=/user/hive/warehouse \
   --hive-import
```

Table 4: Sqoop command to import data from RDBMS

To export data, we may use the corresponding export command. This helps to integrate the data into a SQL free format.

Python on the other hand is built from an eclectic collection of libraries, so it has a lot of native support for almost all the data formats. Depending on the type of operation, one can use a wide variety of ways to load the different formats of data. For example, if Numpy is used for matrix manipulations, then the inbuilt commands like loadtxt() can be utilized to load the flat files. If library like Pandas is used, the file can be read using pd.read_csv (). Similarly, to write a file corresponding save operations, savetxt () and to_csv () can be used respectively. For data formats like excel files and HDF5, Python provides number of libraries, such as, xlrd, openpyxl for excel and h5py for HDF5 etc.

Loading data from databases can be done using the MYSQLdb API. It supports a wide variety of databases such as MySQL, Oracle and Sybase etc. This API can be used to run queries and extract the required data if necessary. Python also provides other libraries such as pymysql and oursql to perform the similar tasks.

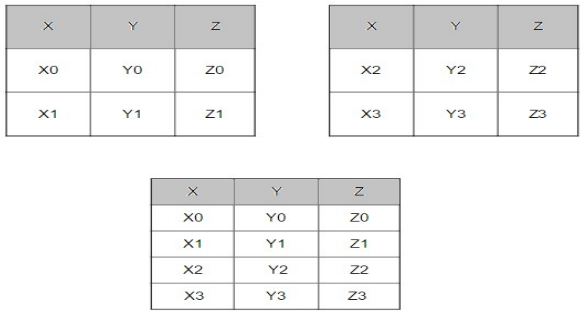
Merging and joining data sets

Merging data refers to concatenation of the two different data sets composed of the either same or different structure. It is generally done to eliminate redundancy and to effectively store similar data in one data set. The methods of merging data sets vary based on the type and structure of the data. Hadoop stores all data in the HDFS, so the data can be accessed and manipulated by various Apache Foundation software such as Hive and PIG. Data sets can be directly modified by using the Hadoop built-in framework "Map Reduce". However, performing such tasks can be tedious and convoluted when Map Reduce alone is used. Hive which is integrated with SQL helps in simplifying this task because SQL not only makes it easy to use, but also utilizes Map Reduce in the background thus making it much more efficient. In order to rake up the benefits of Hive and Map Reduce, data should be voluminous and the number of nodes in the cluster should be proportional to handle the massive data efficiently. So, usage of Hive to perform merging and joining the data sets is the ideal choice.

```
SELECT *
FROM (
select_statement // Select condition 1 from the first table
UNION ALL
select_statement // Select condition 2 from the second table
) unionResult
```

Table 5: UNION ALL command to merge data sets

This can be done by using the UNION ALL command, a simplified syntax to join 2 tables from the Hive warehouse into one single resultant table as shown in table 5. The command spawns a set of Map Reduce tasks in the background which collectively queries the data, gathers the necessary result and joins the end result into one single table. The command can also utilize various other clauses like order by, sort by, cluster by etc. These clauses will refine the results and aid quality merge by reducing any type of redundancy. figure 4 shows one such sample merging operation.



X	Y	Z
X0	Y0	Z0
X1	Y1	Z1

X	Y	Z
X2	Y2	Z2
X3	Y3	Z3

X	Y	Z
X0	Y0	Z0
X1	Y1	Z1
X2	Y2	Z2
X3	Y3	Z3

Figure 4. Sample merging operation

Python can also achieve the same results by using its abundant libraries. If data merging is to be done at the database level then, MYSQLdB can be used. A query similar to Hive query, where UNION query can be made using the previously mentioned API and then the tables can be merged and updated. To perform even more intricate operations joins like inner join, left join and right outer join can also be utilized which will yield more refined results.

Pandas, a Python package, uses a different merging technique. The data is stored in files and based on requirement, data is retrieved into memory and required actions are performed. Interestingly, data is not stored in a conventional database but rather stored within the files themselves. It is retrieved subsequently into memory and required analysis tasks are performed. For example, in Pandas, let us assume that there are 2 data frames which are similar to the tables in figure 4. The instruction from Table 6 has be executed to merge these two tables.

```
df1 = pd.DataFrame({'X': ['x0', 'x1'],
'Y': ['Y0', 'Y1'],
'Z': ['Z0', 'Z1']}
index=[0, 1, 2])
df2 = pd.DataFrame({'X': ['x2', 'x3'],
'Y': ['Y2', 'Y3'],
'Z': ['Z2', 'Z3']}
index= [0, 1, 2])
merge1 = [df1, df2]
result = pd.concat(merge1)
```

Table 6: Pandas code snippet to merge data frames

Pandas has been the leading package used in Python for data analysis because of its ease of use and its flexibility in handling the data. Although not suitable for large voluminous data, Python utilizes its in-memory capacity for analyzing data with high efficiency therefore making Python suitable to handle low or middle-sized data.

Query the data

Querying the data usually refers to the extraction of results which are computed based on several operations on the data sets. Generally, querying the data ranges from displaying information to making charts for visualization. Visualization will be covered in the later phase. Python and Hadoop operate very similarly in terms of implementation when displaying information is concerned. Hadoop's output is stored in a directory of HDFS and it can be shifted to the local file system when required. Hadoop can utilize Hive's service for a better quality query result. Hive as mentioned earlier typically works in the same fashion as that of a standard database, so the query results can be obtained just by running the SELECT command in Hive. One can specify the parameters and the required query conditions (such as WHERE, ORDER BY CLUSTER BY) and can obtain a result appropriately. Other queries include insertion and deletion of records from a database and they too can be executed by using the appropriate commands like INSERT etc. Hive is usually preferred when handling data in Hadoop because it maintains data in a readable format.

Other tools such as PIG can also be used to achieve these results. In juxtapose, Python can query from databases similarly by utilizing the MYSQLdb API and furnish the same results. It is important to note that query capabilities of Python are highly dependent on the memory at its disposal. Functions such as, fetchall () yield good results to retrieve the data present in the rows. Similarly, one can interact with the database with other queries as well. For example insertion and updating the database also can be done by first using the regular SQL query and executing the API query function. However, when using packages like Pandas in Python, we can improvise a number of ways to query the data. Similarly, for insert operation, Pandas has a pd.DataFrame.insert () function, which helps in inserting a certain value to a specified position. Similarly, for all other operation it has its respective functions. Depending on the choice of package there are different functions for different query operations. Nevertheless, Python provides versatility compared to Hadoop because it is tightly coupled with the data, rendering some flexibility in easier manipulation compared to that of Hadoop. Selecting one of the various methods or a combination of methods provides a near perfect solution for efficient pre processing, which paves the crucial foundation for a solid analysis.

4. Data analysis

Once data is accumulated and cured to a desired level, we may proceed to perform analysis, which is the most interesting stage of the complete process. Data analysis refers to the meticulous examination of data to determine any useful results or changes. The analysis process on the data could be operations such as estimation, reductions etc. Each process trying to extract a specific result can help in making informed decisions, and giving other important inferences of significant value concurrently. The tools which perform data analysis have to keep up with the variety and volume of the data. The quality of the results of analysis depends on the type of methods used and the effective use of the tools. These various analysis operations could be as simple as counting of occurrences to as intensive as machine learning algorithms such as regression and forest ensemble (Mitchell, 1997). The characteristic to efficiently obtain results within the given finite resources makes it the superior form of data analysis. It is said that 2.5 quintillion bytes of data is created every day and most of the data generated is subjected to different forms of analysis. It is absolutely imperative to optimize the resource consumption while analyzing such heavy loads of data of so much variety and velocity. But the ultimate goal is to obtain results which give us a decision theoretic advantage over a scenario. Apache Foundation projects have consistently evolved in this front and have shown promising results so far.

4.1 Big data analysis using Apache Foundation Projects

Big data analysis using machine learning is often misunderstood in a convoluted sense. It simply refers the use of various algorithms which make prudent choices to ensure a more accurate result. Nature of the data at hand and the expected results from data analytics are the primary driving factors for selection of a machine learning algorithm. The structure of data can be explained using factors such as, format, variance in the data, accuracy and volume. Also, it is important to find if the data is linearly separable. For example, if the expected result is to effectively store a large matrix, then, one would prefer using dimensionality reduction such as Principle Component Analysis and knowledge discovery paradigm such as Singular Value Decomposition.

Machine learning is slowly taking over the data analysis platform as data, when examined for results, is no longer dependent on the old trivial ways for a solution. The present occurrence of data is in a much more complex state, which not only refers to the volume but also the finer details that it provides. It is imperative to use more refined algorithms to not only analyze the current state of data but also have a prudent approach to the upcoming data. This gives an edge over traditional analysis methods and boosts the effort in the search for the most ideal estimations to give the best results.

4.2 Various language support by Apache Hadoop

Hadoop Map Reduce is conventionally written in Java. This forced data analysts, who are trying to use Hadoop, to learn Java. However, in order to eliminate this inconvenience, Apache developers introduced Hadoop streaming, a feature of Apache Hadoop, which supports Map Reduce code in any language, provided that the language has a model to handle standard input (stdin) and standard output (stdout). This support allows use of an array of scripting languages such as Python, Scala etc to perform map educe tasks. The biggest advantage of Hadoop streaming is not just the eclectic languages, which can be used, but also the libraries and packages that are associated with them. For example Python is filled with diverse libraries such as Numpy and Scipy etc. which can be used along with Map Reduce to give better choices in terms of the tasks execution. Reading from different formats, which are not supported by Hadoop, is also temporarily solved as one can use the corresponding libraries of those languages to extract from those formats. Although this shows a proportional increase in time with the increase in the data size, it presents a temporary fix for the problem.

4.3 Apache spark for big data analytics

Apache Spark is the latest addition to the cluster computation family (Ryza, Laserson, Owen, & Wills, 2015; Spark, 2015). It not only claims lightning fast computation but also shows prudent promise in efficient RAM utilization compared to any other previously discussed frameworks. Apache Spark originated at the UC Berkeley AMPLab. It is an open source framework. Spark combines an engine for distributing programs across clusters of machines with an elegant model for writing programs atop it. Spark has been contributed to the Apache Software Foundation which is arguably the first open source software that makes distributed programming truly accessible to data scientists. Spark (Ryza, Laserson, Owen, & Wills, 2015; Spark, 2015) maintains Map Reduce linear scalability and fault tolerance, but extends it in three important ways.

1. Unlike Map Reduce which writes intermediate results to the distributed file system, Spark can pass them directly to the next step in the pipeline, i.e. Spark’s engine can execute a more general directed acyclic graph (DAG) of operators rather than relying on a rigid map-then-reduce format

2. It supplements this ability with rich transformations that empower developers to express

computations all the more actually. It has a solid engineer center and streamlined API that can speak to complex pipelines in a couple of lines of code.

3. Spark extends Map Reduce with in-memory processing. This ability opens up use cases that distributed processing engines could not previously be approached. Spark is well suited for algorithms that require multiple passes over a data set such as profoundly iterative/linear algorithms, as well as interactive applications that need scanning large in-memory data sets in order to quickly respond to user queries.

Hence, Machine learning algorithms that make multiple passes over their training set can cache it in memory. Data scientists can keep data set in memory while exploring and getting a feel for it they can run queries and easily cache transformed versions of it without suffering a trip to disk. Spark supports variety of tools already existing in the Hadoop ecosystem. It can read and write data in all of the data formats supported by Map Reduce. This allows spark to interact with the formats commonly used to store data on Hadoop such as Avro, Parquet (and good old CSV). It can also read from and write to NoSQL databases like HBase and Cassandra.

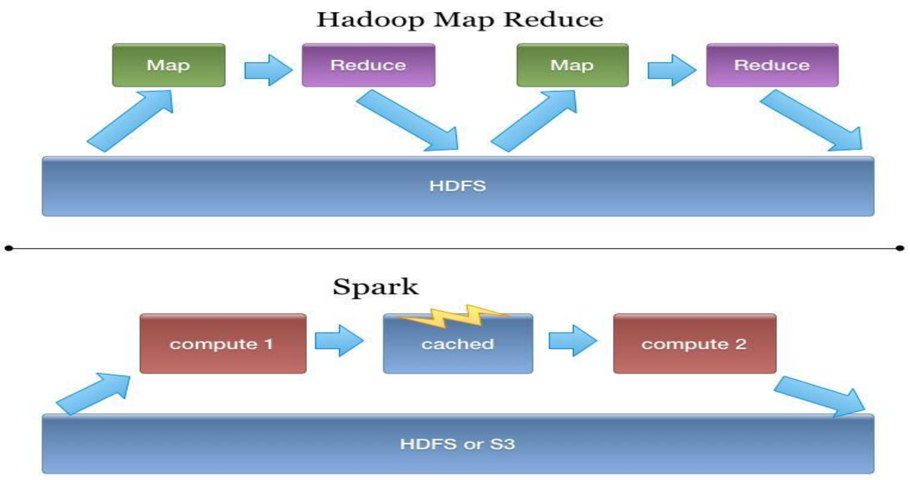


Figure 5. Comparison of Hadoop and Spark data processing pipeline (White, 2012)

Spark programming model: As shown in the figure 5, Spark programming (Maniyam, 2015) starts with a data set which is residing in some form of distributed and persistent storage like the Hadoop Distributed File System (HDFS). A typical Spark program consists of following steps:

• Specifying the transformations on the provided data

• Call the functions to generate outputs and return the results to the local memory or persistent storage

• Execute the local operations on the results generated by distributed computing and based on these results, next set of actions are determined

Understanding Spark means understanding the intersection between the two sets of abstractions of the framework: storage and execution. Spark pairs these abstractions in an elegant way that essentially allows any intermediate step in a data processing pipeline to be cached in memory for later use.

Resilient Distributed Data set (RDD): Spark uses Resilient Distributed Data set (RDD) to achieve full performance efficiency. An RDD is nothing but partitioned data, a chunk of data which is fault tolerant, easily parallelizable and immutable. Apache Spark has been the leading software to perform much more efficiently in juxtapose to the other software that are mentioned above. Its use of the RDD makes a very high utilization of the RAM and the cache such that it is able to tackle big data with a much more efficient approach. The fast access of the data stored in the cache and the use of RDD in coalition with the involvement of a cluster makes this the most relevant big data framework for data analysis. In addition, Apache Spark has a collection of a wide variety of machine learning algorithms which take advantage of RDD mechanism and the cluster to provide better results compared to any other tool in this genre.

4.4 Case Study: Dimensionality reduction using Apache Spark, an illustrative example from Scientometric big data

Scientometrics is the study of measuring and analyzing science, technology and innovation. It is a study to differentiate quality from quantity. Scientometrics deals with the measurement of impact of journals and institutes, understanding of scientific citations, mapping scientific fields and the production of indicators for use in policy and management contexts. The implosion of journals and conference proceedings in the science and technology domain coupled with the insistence of different rating agencies and academic institutions to use journal metrics for evaluation of scholarly contribution present a big data accumulation and analysis problem. The motivation of this project proposal originates from the paradox the research community is facing today which is “Need Vs Greed”. The otherwise noble insight to build and maintain citation-based indices and to endorse all scientists and researchers, who have made major contributions in the advancement of research and development, attributed as Impact Factor is being abused of late. H-index, another highly known metric, often misused and manipulated by the research community through practices like extensive self-citation, copious citation and at journal level, coercive citation, is no longer beyond doubt. There have been instances of misuse, where Editor-in-Chiefs of two high impact factor journals are seen publishing extensively not only in their own journals but also into each other. The journals continue to demonstrate such practice till date undermining the dignity and integrity of the structure of scholarly publications. Another notable trend, followed opportunistically by editors of low ranked journals is to persuade authors cite their journal articles with the ulterior motive of pushing impact factors. If one journal coerces to improve its rank, others gravitate and this becomes a trend that contaminates the whole publication process. One possible solution could be penalizing the usage of self-citations which might reduce, if not deter the coercive motivation. Although journal’s prestige is reflected in its impact factor, the overall framework of measuring “prestige” and for that matter its “internationality” demands clear understanding and usage of deep rooted parameters. Choice of a set of smart indicators and alternatively, defining new and unbiased ones is inevitable for measuring “internationality”. There exist several vague and unreliable indicators for the term “Internationality” which are based on including attributes like ISSN number, Editorial board members, reviewers, authors, Editors, Publishers and other associated entities of publishing community. Authors refute such definitions and claim that “internationality” should be defined as a quantifiable metric, a measure of international spread and influence, which is devoid of any kind of misuse or manipulation. It must remain unbiased towards the origin of the journal, author or article and must calibrate journals on the basis of quality of research output it publishes. The Case Study intends to build and validate such models.

4.4.1 *Why SVD? The solution may be the problem*

A section of academia is not very pleased with scientometric indicators being used as the primary yardstick for faculty evaluations. There are "theories" that discredit the entire methodology and blame "Publish or Perish" doctrine. The authors have been to discourses where Eugene Garfield, the father of Impact Factor has been criticized, unfairly and relentlessly. There is some element of truth in the claims that the evaluation scheme is not fair; however, the premise that the field of Scientometric analysis should not be taken seriously, is a bit farfetched. Scientometric indicators have become tools for survival and a weapon to attain glory. It is not the metric or the study of metrics, rather survival instinct and human greed that are responsible for importing uncertainty. So, incremental improvements and sophisticated modeling and evaluation methods are required in Scientometric analysis, not the extremist recommendation of ignoring it altogether. The solutions presented by several scholastic investigations include proposing new metrics such as NLIQ (Non Local Influence Quotient), OCQ (Other Citation Quotient), Internationality, Cognizant Citations, Copious Citations, Diversity Score, NGCR (Non Genealogy Citation Score), NGCN (Non Genealogy Citation Network) *etc. (Ginde et al.,2016; Ginde et al.,2015; Ginde, 2016).* However, given a huge corpus of articles published in several issues of a journal, published over a period of years present a daunting task in terms of rating the articles. The task of scholarly value characterization by using a single metric is therefore a pragmatic way to address the issue of identifying article impact rather than exploring and computing several different metrics. SVD could thus turn out to be an effective way of achieving such goal. After mining the massively scaled scholarly data using web scraping methodology, we constructed a matrix where the rows represent the number of articles in a journal, published over a period of years, and columns represent the different metrics proposed by experts and agencies in Scientometrics. Clearly, the matrix is rectangular and computing the Eigen values, a reliable indicator of the characteristics of the information matrix, is not possible. This justifies the motivation for exploiting SVD. This is a classic big data problem with Scientometric implications.

4.4.2 *Need for SVD:*

The huge rectangular matrix, which represents the complete scholarly data set gathered using web scraping is the source of this case study. We describe SVD method for visualization and representation of article data set using a smaller number of variables. SVD also yields detection of patterns in this article information matrix. Since the matrix is humongous, we performed exploratory research and evaluated SVD computation using Map Reduce, Apache Mahout and Apache Spark. Dimensionality reduction has played a significant role in helping us ascertain results of analysis for this voluminous data set. The propensity to employ such methods is evident because of the phenomenal growth of data and the velocity at which it is generated. Dimensionality reduction such as Singular Value decomposition and Principal Component Analysis(PCA) solve such big data problems by means of extracting more prominent features and obtaining a better representational version of the data. The representative data set tends to be much smaller to store and significantly easier to handle for further analysis. These dimensionality reduction methods are often found in most of the tools which handle large data sets and are used to perform rigorous data analysis. These tools include Apache Mahout, Hadoop, Spark, R, Python to name a few. The ease of employing these tools is directly dependent on the performance of each one of these tools to compute, assess and store the results efficiently, with optimal use of resources. Singular Value Decomposition is a convoluted dimensionality reduction technique. It requires multiple steps that need to be performed in order to extract singular values and the following right and left singular vectors. Therefore, the capacity of a big data tool to perform such computation efficiently is a necessity.

The singular value decomposition (SVD) is a very useful tool, and it is used for almost every scientific discipline. For example, it can be used for efficiently simulating high-dimensional partial differential equations by taking all the data generated from the simulations, reducing the data dimensionality by throwing away some of the singular values, and then simulating the lower-dimensional system. SVD gives us an optimal low-rank representation. This fact guarantees that previously explained sort of simulation preserves most of the detail in a system, as getting rid of the extra modes (singular values) in the system is guaranteed to get rid of the least important modes. With a little variation, SVD is used everywhere from physics to machine learning for dimensionality reduction. In fact, the algorithm commonly known as Principal Component Analysis (PCA), for instance, is just a simple application of the singular value decomposition.

Relationship between Singular Values and the Input matrix

The singular value decomposition (Kalman, 1996; Golub & Van Loan, 2012) takes an m × n matrix and returns three matrices that approximately equal it when multiplied together.

$$A \approx U\,DV^{T}$$

1. U is an m × k matrix whose columns form an orthonormal basis for the articles space.
2. D is a k × k diagonal matrix, each of whose entries correspond to the strength of one of the concepts. The values on the diagonal of D are called the singular values.
3. V is a k × n matrix whose columns form an orthonormal basis for the features parameters space.

The m-dimensional vectors making up the columns of U are called left singular vectors, whereas the n-dimensional vectors making up the columns of V are called right singular vectors. The set of left and set of right singular vectors are orthonormal (that is, both orthogonal and normal). Normality means that each singular vector is of unit length (length 1). A set of vectors is said to be orthogonal if any pair of vectors in the set is orthogonal; recall that vectors are orthogonal if and only if their product (equivalently cosine) is zero (0).

For large matrices, usually we don't need the complete factorization but only the top singular values and its associated singular vectors. This can save storage, denoise and recover the low-rank structure of the matrix.

Suppose $D = \text{diag}\{\sigma_1, \sigma_2, \ldots\ldots\ldots\ldots\ldots, \sigma_n\}$ By convention it is assumed that $\sigma_1 \geq \sigma_2 \geq \sigma_3 . \geq \sigma_n \geq 0$ The values $\sigma_1, \sigma_2, \sigma_3, . \sigma_n$ are called the singular values of A

In other words Singular values are unique to a particular column in A and they can be used to associate a relation even if they are not arranged in descending order they can be used to obtain the A matrix back. This transformation itself doesn't solve dimensionality reduction, to completely be able to reduce dimensions we must be able to decide the least significant Singular Values of the Σ Matrix. Here least significant values most often refer to the lowest values of the Σ matrix and then equate them to zero. In doing so we retained the most prominent values and discarded the least significant values that form the original matrix. When we put back the decomposed matrix to derive a new matrix, it will have reduced dimensions. To demonstrate this let's assume that

$$\Sigma = \begin{matrix} \sigma 1 & 0 & \dots & 0 \\ 0 & \sigma 2 & \dots & \vdots \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \dots & \sigma n \end{matrix}$$

So on removing p least significant values or the lowest values of the Singular values and equating them to zero, let's say we get Σ'

$$\Sigma' = \begin{matrix} \sigma 1 & 0 & \dots & 0 \\ 0 & \sigma 2 & \dots & \vdots \\ 0 & 0 & \sigma(n-p) & 0 \\ 0 & 0 & \dots & 0 \end{matrix}$$

Now on re-composing the matrix using the newly formed Σ' we obtain a dimensionality reduced matrix

So, $U\ \Sigma'\ V^T = A'$

A' is the dimensional reduced matrix which has an order lesser than m x n ideally. In this manner, we can observe how SVD achieves dimensionality reduction and how we can manually set the least significant values or the lowest values of the Singular Values. For example, less values of p are used to ensure minimal information loss for a more precise and reduced matrix. Higher values of p are needed for a resultant matrix with relatively higher information loss and a greater degree of dimensionality reduced matrix depending upon the use case.

4.4.3 *SVD using Hadoop Streaming with NUMPY*

The aforementioned use of various languages in Hadoop is termed as Hadoop Streaming. This feature showcases the flexibility of using different languages to achieve certain tasks more effectively. It uses API such that it interacts with the Unix standard streams as input and output hence making it very suitable for large text batch processing (White, 2012). Firstly, in a Map Reduce job the input is passed to the mapper as a standard input file. Hence, using the NUMPY function loadtxt function, we get the initialized array, which is fed to HDFS when the program is initialized. Next, we use a self-improvised method to pass the values from the mapper and reducer. Conventionally, Map Reduce works in a shuffle and sort fashion in order to segregate the data and process it line by line. Since we are using an array, sorting this could prove fatal as the values of the matrix get interchanged. Thus after the array is read from standard input, we initialize the Key to 1.

```
For every row 1st column element = key
                                   Key=key+1
```

We fill the entire first column with the key values. Sorting the array doesn't change the orders of the original values. This provides an extra row of an arbitrary key value solving the sort problem on Map Reduce with arrays. Newly keyed values are sent to the reducer through standard output. Next, the matrix sorting is stopped to enable reducer to directly work on the dense matrix. Just like the mapper, this reducer too reads input from standard input file written by the mapper. The reducer has essentially only one task before it starts to compute the singular values, which is to first de-key the array passed by the mapper as it initially has a column of key elements in the first column. This is done in the first nested for loop, where x[i][j]=a[i][j+1]. Next, we shift the actual array within the key values to a new array which is going to be used for computing those values of the SVD. API provided by the NUMPY library is used to compute the individual matrices of the SVD. The reason for using NUMPY is its speed and accuracy in matrix manipulations. The corresponding data sets can be easily imported for other ML algorithms as NUMPY extensively supports ML on data sets. After finding the singular values, we observe that the following singular values appear in a single row and it does not retain its diagonal matrix form. However, since we need the singular values in its original form, we will transfer all the values to the diagonals of another arbitrary matrix so as to associate the following relationship later with the input matrix. We now print the following individual matrices of the singular valued decomposition: $A = U \Sigma V^T$

Complexity of using Map Reduce

Following equations represent complexity of the mapper and reducer functions used for this computations. These complexity polynomials are derived from the code snippets sourced from GitHub. The gist of the mapper and reducer codes outline a few operations such as reading the values line by line, keying the array and then SVD is performed by the reducer.

For mapper:

For reducer:

A(n) is the complexity polynomial of NUMPY SVD function.

Since the following Map Reduce operations take place sequentially because reduce can not start until we have the complete dense matrix. This implies that, we have to wait for the mapper to pass the complete matrix to the reducer. The complexity will be the summation of mapper and reducer . Calculations yield n>=23.

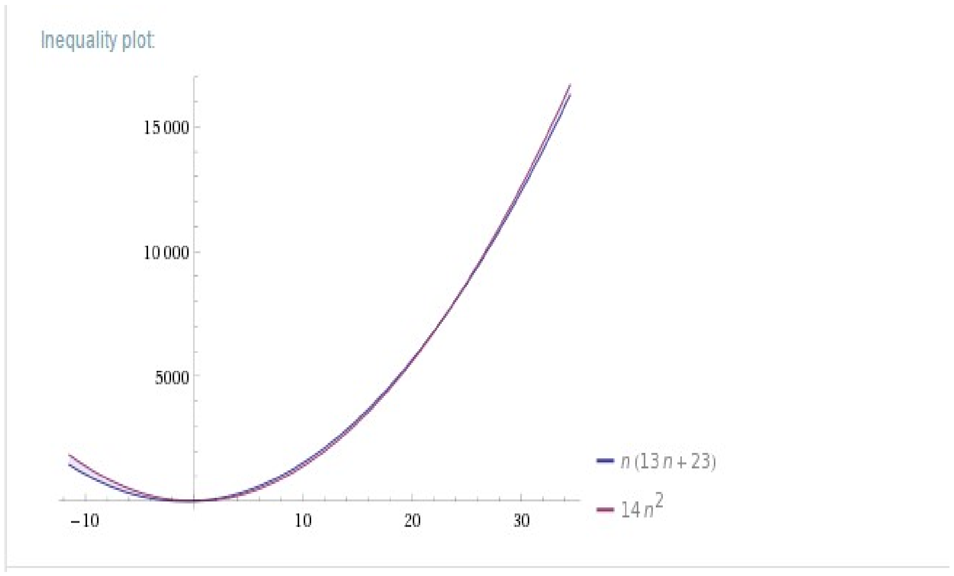


Figure 6: Complexity plot

So by definition

So

As mentioned above, A(n) is the complexity polynomial of NUMPY SVD function. Figure 6 shows the complexity plot of these two functions. The results show that Using Map Reduce alongside Python might be easy to use but the complexity of performing this task is not ideal. It is noted that after a certain cardinality of dimensions, this method is known to fail for much larger matrices (Ding, Zheng et al.,2011).

4.4.4 *SVD using Apache Mahout*

Apache Mahout is an open source project that is primarily used in producing scalable machine learning algorithms and also to perform complex matrix operations given a large data set (Owen & Owen, 2012). Apache Mahout heavily depends on the use of HDFS and the Hadoop

Ecosystem in general as its main focus is on performing intensive distributed machine learning on commodity hardware. This is considered as an advantage as most other tool kits require a highly configured hardware to perform similar operations. Mahout allows these computations in a distributed environment. It enables to perform such tasks in spite of the greater complexity of the computation involving big data. Next section will showcase the performance and accuracy of Apache Mahout's singular value decomposition function over a voluminous data set.

Generating sequence files for Mahout

Apache mahout requires a very specific type of input for it to process any operation. Apache Mahout utilizes the use of sequence files. A sequence file is essentially a binary file which contains data in a key – value pair formation. Usually Hadoop uses Sequence files as a part of its intermediate operations in Map Reduce which occurs during the shuffle and sorting operation of the Map Reduce phase. However for our requirement we need to associate the sequence files into vectors so as to run it through Mahout. The primary reason for this approach is that a vectorized sequence file can be easily parallelized within the Hadoop ecosystem . This is done using code snippet in table 7 in Java.

```
Writer matrixWriter = new SequenceFile.Writer(fs,
configuration, new Path(matrixSeqFileName),

IntWritable.class, VectorWritable.class);
```

Table 7: Source code to generate Sequence files

Further processing completely relies on this sequence file. Writer code in table 7 accepts any text file and converts it into a sequence file accordingly. Code in table 8 is used to convert sequence file format into a vectored state for further computation.

```
IntWritable key = new IntWritable();

VectorWritable value = new VectorWritable();
```

Table 8: Code to convert Sequence file to Vector format

Lanczo's Algorithm

Lanczo's algorithm is a neat design to extract Eigen value decomposition of a matrix. This also tries to achieve parallelization in order to utilize Hadoop to its best potential . To understand this algorithm better let us look at the necessary computations and transformations that are required to perform SVD using Lanczo's algorithm. In this case to extract singular vectors they are essentially taken from the Eigen vectors of $X^T * X$, where X is the matrix which as to be decomposed (Wilkinson, Wilkinson, 1965). This operation utilizes a seed vector (v) which is obtained from the cardinality equal to that of the number of columns that are found in the matrix. The seed vector is repeatedly multiplied with X to obtain v'=X .times(v). Syntactically we use times(v) instead of just multiplying it v times because this ensures mahout runs the whole operation a single iteration or pass hence maintaining the efficiency and then after that the previous v'' part is removed, here X is assumed to be symmetric. If the matrix is not symmetric then the following seed vector is multiplied with the product of $X *X^T$. After a certain number of iterations, let us assume it to be k. thus, there is now a formation of a new auxiliary matrix of the order k x k. This matrix provides the best estimation of the singular values that are needed to be extracted. Most times, the estimation of these values are very near to the expected value, but there is a small chance that the following singular values might not be same. So to ensure better accuracy the largest singular values around 3k and the smallest singular values are maintained and the rest are usually discarded for practical purposes. A given spectrum is maintained to ensure accuracy.

Reading the Mahout generated vectors. On running the mahout job using CLI we obtain our results in a vector format. The small snippet of code in table 9, which is written in Java can be used to help us convert these vector files into readable format. Similar to writing a file we have to also read a file this can be done using

```
Reader matrixReader = new SequenceFile.Reader(fs, new
Path(fileName), configuration);
```

Table 9: Code to convert vector file into readable format

Code in table 10 refers to the Hadoop sequence file reader to ensure that the configuration is met. We then use the vector package to perform read operations on the so called given sequence file. This can be done as below

```
Vector vector = value.get();
```

Table 10: Sequence File reader

And we then place the values into a temporary matrix and print it back into a text file for readable format. This is pretty much the inverse of the writer function. Upon running the Java program to convert these sequence files back to text files, then, we get the Orthonormal matrix V from the Singular Value Decomposition which can be used for further use.

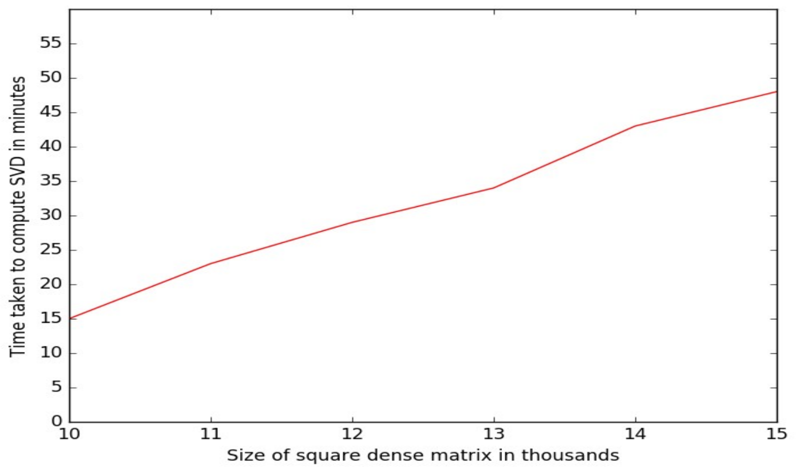


Figure 7. Time vs Size of matrix for SVD using Apache Spark

4.4.5 *SVD using Apache Spark*

MLlib is Spark's scalable machine learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as underlying optimization primitives (Meng et al.,2016). The Spark Machine learning library is the primary source of packages for all the machine learning algorithms. Here we utilize the spark.mllib.linalg package for the purpose of computing SVD and any other linear algebra operations. We also utilize the spark.mllib.linalg.distributed.RowMatrix. This is one of the most efficient ways to perform operations on large, dense or sparse matrices that use Distributed Row matrices. This utilizes the RDD element in spark to perform faster operations on matrices. MLlib has other packages as well, such as, regression, recommendation etc., but we will be using only the linalg package for demonstration of the working principle. The MLlib library as mentioned before is built on spark so that it can use the spark's RDD frame work more efficiently but more importantly to take advantage of the parallelization potential offered by Spark.

We utilize the distributed matrix component of Spark to meet this objective. A distributed matrix has both row and column indices. It is stored in a distributed manner in one or multiple RDDs. This is an effective tool to handle large scale matrix operations in an in-memory approach of computation. There are 4 categories of distributed matrices provided by Spark but we will emphasize on only Row Matrix. To demonstrate SVD or any eigenvalue decomposition, the best implementation can be done only by using Row Matrix. A Row Matrix is a row oriented matrix. The indices are labeled for each row of the matrix . Each row is maintained inside an RDD as a local vector hence making the Row Matrix a collective set of local vectors. However, the column size is kept as small as possible to ensure better efficiency while storing the matrix in an RDD. As shown in the code snippet in table 12, we find out the singular values and the right singular vectors from the eigenvalues and the eigenvectors of the matrix X^TX, where X is the input matrix. The left singular vectors on the other hand are calculated if specifically requested by the user by setting the parameter as true. Depending on the size of n and k we calculate the eigenvalues and vectors respectively. If $n < 100$ or if $k > n/2$, then we calculate the following values of greatest and smallest eigenvalues locally, else we use ARPACK to compute it in a more distributive manner and compute the following results on the driver.

Execution of the Scala code in the Spark Scala shell can provide different computation times for different sizes of the Matrices. Figure 7 shows how well Spark maintains its scalability over different sizes of data. The Spark Scala code utilizes the complete potential of the distributed row matrix synchronously with the use of RDD to provide a much more efficient use of resources involved in computation. Following is a simple code for the SVD computation in Apache spark along with the starting and completion time for the operation.

```
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.linalg.Matrix
import org.apache.spark.mllib.linalg.Vector
import org.apache.spark.mllib.linalg.distributed.RowMatrix
import org.apache.spark.mllib.linalg.SingularValueDecomposition
import java.util.Calendar

  // Load and parse the data file.
  val rows = sc.textFile("file:///home/cloudera/test.txt").map { line =>
    val values = line.split(' ').map(_.toDouble)
    Vectors.dense(values)
}
//Show Starting Time
  val mat = new RowMatrix(rows)
val currentHour = Calendar.getInstance().get(Calendar.HOUR)
val currentMinute = Calendar.getInstance().get(Calendar.MINUTE)
{
  // Compute SVD
  val svd = mat.computeSVD(mat.numCols().toInt, computeU = true)
 }
//Show Ending Time
val currentHour = Calendar.getInstance().get(Calendar.HOUR)
val currentMinute = Calendar.getInstance().get(Calendar.MINUTE)
```

Table 12: Simple code for SVD computation in Apache Spark.

Of the three tests, Apache Sparks wins with flying colors and proves to be the best solution for Big Data analytics based problem solving among all the Apache Foundation Projects for machine learning based analysis.

4.5 Data analysis through visualization.

Data visualization can be termed as a form of data analysis. It provides the visual representation of patterns in data (Ginde, 2016). Effective visualization can help to understand and relate to the data, communicate and represent data intuitively to others. As author, data journalist and information designer David McCandless said in his TED talk: "By visualizing information, we turn it into a landscape that you can explore with your eyes, a sort of information map. And when you're lost in information, an information map is kind of useful."

Data visualization can be as trivial as a simple table, elaborate as a map of geographic data depicting an additional layer in Google Earth or complex as a representation of Facebook's social relationships data. Visualization can be applied to qualitative as well as quantitative data. Visualization has turned into an inexorably well-known methodology as the volume and complexity of information available to research scholars has increased. Also, the visual forms of representation have become more credible in scholarly communication. As a result, increasingly more tools are available to support data visualization. High impact visualization is like a picture speaking a thousand words. Selecting good visual technique to display the data holds key to a good impact. Fancy bubble charts, Time domain based motion graphs are possible now because of the languages such as python and R.

Why Big data visualization is challenging and different from traditional data visualization?

Visualization approaches are used to create tables, diagrams, images, and other intuitive display ways to represent data. Big Data visualization is not as easy as traditional small data sets visualization. The extension of traditional visualization approaches have already been emerged but far from good enough. As Wang (Wang, Wang & Alexander, 2015) says, Visualization can be thought of as the "front end" of big data. Traditional data visualization tools are often inadequate to handle big data, scalability and dynamics are two major challenges in visual analytics of it. In large-scale data visualization, many researchers use feature extraction and geometric modeling to greatly reduce data size before actual data rendering. Choosing proper data representation is also very important when visualizing big data (Wang et al., 2015). There are plenty of visual analytics solutions available online. SAS Visual analytics, Plotly and Tableau are the most popular. Tableau is a business intelligence (BI) software tool that supports interactive and visual analysis of data. It has an in-memory data engine to accelerate visualization. Tableau has three main products to process large-scale data sets, including Tableau Desktop, Tableau Server, and Tableau Public. Tableau Desktop is free for students and academicians. It embeds Hadoop infrastructure and uses Hive to structure queries and cache information for in-memory analytics. Caching helps reduce the latency of a Hadoop cluster. Therefore, it can provide an interactive mechanism between users and Big Data applications (Wang et al. 2015). Plotly is built with 100% web technology. This makes our Plotly is Free for hosting public data (like Tableau Public). It is an online analytics and data visualization tool which provides online graphing, analytics and statistics tools for individuals and collaboration, as well as scientific graphing libraries for Python, R, MATLAB, Perl, Julia, Arduino, and REST. However, all these visualization tools lack advanced visual analytics capabilities. Hence, scripting languages such as R and Python need to be used to provide adept visualizations utilizing the complex data sets. Following are a few visualizations for the scholarly articles data set, acquired through web scraping and cured using various data cleansing operations in Plotly.

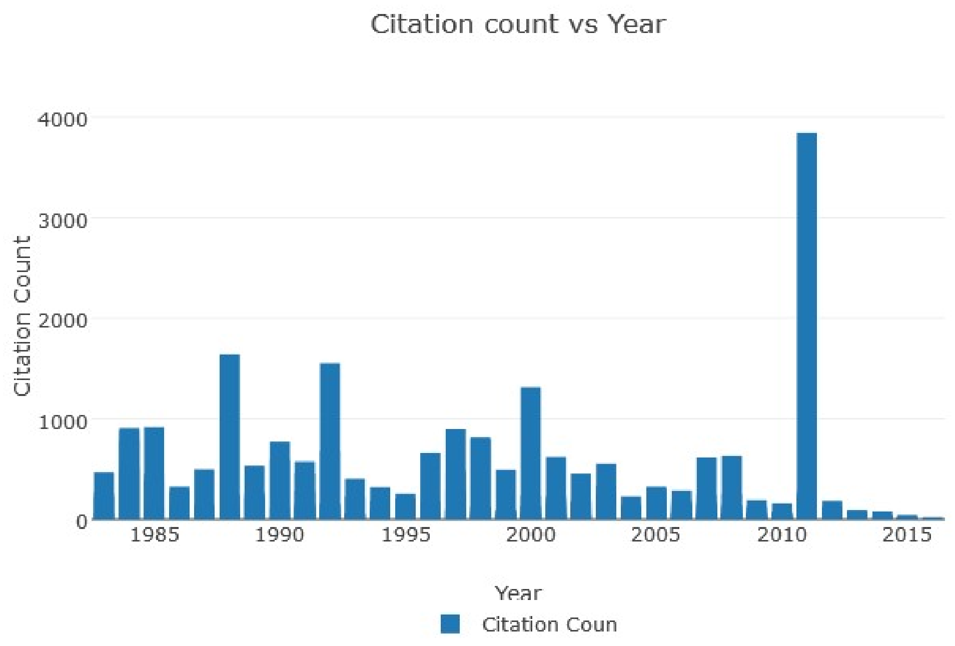


Figure 8 Simple bar chart of citation count vs year

Figure 8 shows one such graphs plot for over 40 Journals and 4000 thousand articles spanning over last 30 years.

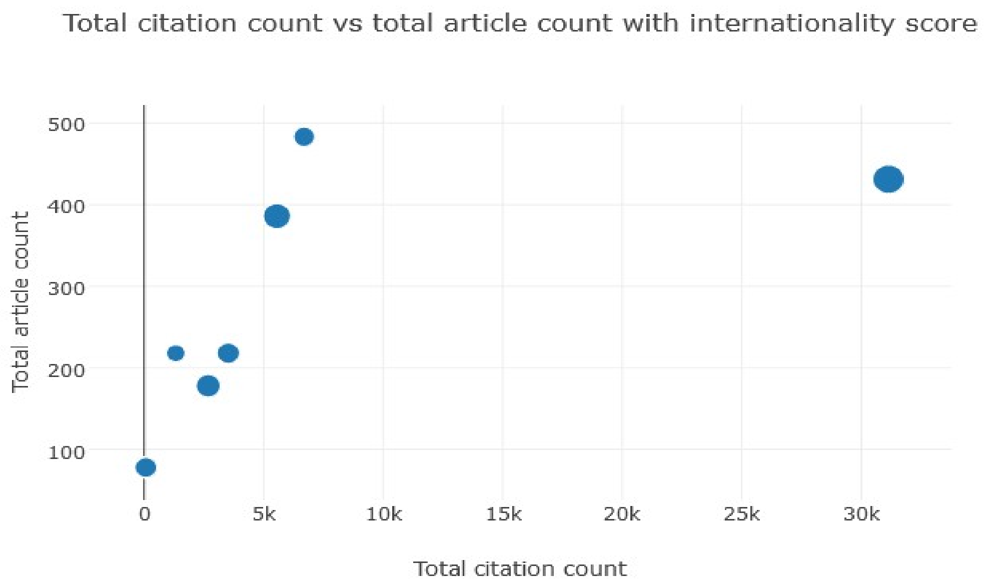


Figure 9 Bubble chart with multiple dimensions

Figure 9 shows the visualization of the total citations against articles where each bubble size corresponds to the Journal's internationality score.

Conclusion

This era of Big Data which is the next frontier for innovation, competition and productivity, has begun a new wave of scientific revolution. This revolution is here to stay for a very long time. Fortunately, we will witness the upcoming technological leapfrogging. Today's Big Data problem will become small data set problem in the near future due to steady growth in technology and improved computer architecture. The gap between the computation and performance is steadily reducing. Any Big Data problem has to be solved in various stages. First one is data accumulation. There are four basic methods of data accumulation and toughest ones are web scraping and web crawling. Second one is Data curation. This stage is the most time consuming and most crucial stages of Big Data problem solving. Data curing can be done using various languages and methodologies based on the requirements. Python, R and Apache Foundation Project's Hive and PIG are the most preferred language and platforms for this. We have explained how a simple python cleansing operations can be scaled to Map Reduce program with legible and apt adaptations in source code. Third stage is Data analysis. This stage provides solution and answers to all the complex questions. The answers are acquired using desired Machine Learning algorithms on the cured data. Data visualization which is also a form of effective data analysis can be used based on the requirement. Tableau is a most widely used solution of data visualization so far. As a case study we have elaborated on architectural changes of Apache spark, such as RDD, which boost the overall performance of the Spark in big data computation. Apache Foundation Projects is the place to find solution for all the Big Data relation computational solutions.

It should not be lost in translation that the role of computing in big data is beyond storage and high performance. Rich discovery from data is imperative and extremely useful. With advances in technologies in the past decade, the amount of data generated and recorded has grown enormously in every field of science. This extraordinary amount of data provides unprecedented opportunities for data-driven decision making and knowledge discovery. However, the task of analyzing such large scale data set poses significant challenges and calls for innovative analytical/statistical methods specifically designed for faster speed and higher efficiency. The chapter has elaborated a few methods and tools to tackle the big data problem for complex computer modeling, analysis and discovery of knowledge.

While developing methodologies for big data and exploiting the existing ones for discovery of knowledge, the authors foresee a number of applied research problems to be addressed in the next decade or so. These include (1) The problem of sustainable computing in data centers, provided by Amazon, Google, Microsoft, IBM and particularly a host of small and medium scale players, (2) The problem of flood forecasting using complex computer modeling and big data methods (3) Prediction of pest infestation in agriculture, thereby measuring the reduction in crop quality or quantity, (4) Identifying optimal number of control factors/parameters in the context of designing products in the industry, with applications in pharmaceutical industry, Nano-Engineering and bio-plastic productions and (5) using methods like SVD to identify single valued and most critical aspect from multi dimensional and voluminous data.

References

1 Gouri, G., Saha, S., Mathur, A., Venkatagiri, S., Vadakkepat, S., Narasimhamurthy, A., Daya Sagar, B.S. (June 2016). ScientoBASE: A Framework and Model for Computing Scholastic Indicators of non local influence of Journals via Native Data Acquisition algorithms. Journal Of Scientometrics. 1-51. doi:10.1007/s11192-016-2006-2
http://link.springer.com/article/10.1007/s11192-016-2006-2

2 Ginde, G., Saha, S., Balasubramaniam, Chitra., R.S, Harsha., Mathur, A., Daya Sagar, B. S., Narsimhamurthy, A. (August 2015). Mining massive databases for computation of scholastic indices - Model and Quantify internationality and influence diffusion of peer-reviewed journals. Proceedings of the Fourth National Conference of Institute of Scientometrics, SIoT.

3 Ginde, G. (2016). Visualisation of massive data from scholarly Article and Journal Database A Novel Scheme. CoRR abs/1611.01152

4 Ryza, S., Laserson, U., Owen, S., & Wills, J. (2015). Advanced Analytics with Spark. O'rielly publications.

5 Leskovec, J., Rajaraman, A., Ullman, J. D., (2011). Mining Massive Data Sets. Cambridge University Press.

6 Shukla, N. (2014). Haskell Data Analysis Cookbook. Packt Publishing

7 Bora K., Saha S., Agrawal S., Safonova M., Routh S., Narasimhamurthy A. M. (2016). CD-HPF: New Habitability Score Via Data Analytic Modeling. Journal of Astronomy and Computing. Preprint arxiv:1604.01722v1.

8 Mitchell, T. M., (1997). Machine Learning. McGraw-Hill.

9 Agarwal, B., Ravikumar, A., Saha, S., (2016a), Big Data Analytics – A Novel Approach to Big Data Veracity using Crowdsourcing Techniques. Unpublished manuscript.

10 Agarwal, B., Ravikumar, A., Saha, S., (2016b). A Novel Approach to Big Data Veracity using Crowdsourcing Techniques and Bayesian Predictors. Proceedings of the 9th Annual ACM India Conference (pp. 153-160).

11 Owen, S., & Owen, S. (2012). Mahout in action.

12 Spark, A. (2015). Apache Spark: Lightning-fast cluster computing.

13 Gartner, data. http://www.gartner.com/it-glossary/big-Intel.BigThinkers on Big Data,
http://www.intel.com/content/www/us/en/bigdata/big-thinkers-on-big-data.html, 2012.

14 Zicari, Roberto. Big Data: Challenges And Opportunities. 1st ed. 2012. Web. 12 Feb. 2017.

15 Wang, L., Wang, G., & Alexander, C. A. (2015). Big Data and Visualization: Methods, Challenges and Technology Progress. Digital Technologies, 1(1), 33-38. doi:10.12691/dt-1-1-7

16 Maniyam, S. (2015, February 23). Moving From Hadoop to Spark - SF Bay Area ACM. Retrieved February 12, 2017, from http://www.sfbayacm.org/sites/default/files/hadoop_to_spark-v2.pdf

17 L, P. C., & Zhang, C. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. Information Sciences, 275, 314-347. Retrieved February 12, 2017, from http://dx.doi.org/10.1016/j.ins.2014.01.015

18 Kalman, D. (1996). A Singularly Valuable Decomposition: The SVD of a Matrix. The College Mathematics Journal, 27, 2-23. Retrieved February 12, 2017, from http://www.jstor.org/stable/2687269

19 Golub, G. H., & Van Loan, C. F. (2012). Matrix Computations (Third ed.). John Hopkins University Press.

20 White, T. (2012). Hadoop: The Definitive Guide(Third ed.). O'rielly publications.

21 Ding, M., Zheng, L., Lu, Y., Li, L., Guo, S., & Guo, M. (2011, November). More convenient more overhead: the performance evaluation of hadoop streaming. In Proceedings of the 2011 ACM Symposium on Research in Applied Computation (pp. 307-313). ACM.

22 Wilkinson, J. H., & Wilkinson, J. H. (1965). The algebraic eigenvalue problem (Vol. 87). Oxford: Clarendon Press

23 Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity.

24 Hey, T., Tansley, S., & Tolle, K. M. (2009). The fourth paradigm: data-intensive scientific discovery (Vol. 1). Redmond, WA: Microsoft research.

25 Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D.B., Amde, M., Owen, S., Xin, D., Xin, R., Franklin, M.J., Zadeh, R., Zaharia, M., Talwalkar, A. (2016). Mllib: Machine learning in apache spark. Journal of Machine Learning Research, 17(34), 1-7.

Apendix 1

Spark Implementation and configuration

We have already discussed about the architecture and features of Apache Spark. Let us now in detail see the practical implementation of the Spark on a cluster along with its respective configuration.

Building Apache Spark Using Maven

As mentioned before building Apache Spark is very crucial and using a prebuilt version of Spark is not advised for our purpose. This is mainly because the prebuilt version of spark doesn't come with the necessary BLAS and ARPACK libraries which are required. This causes Spark's MLlib library to use a built in F2J implementation of most of the Machine learning algorithms this can seriously affect the performance of the task at hand as we deal with big data an F2J implementation is ill advised. Building from source also ensures that it perfectly integrates Scala with Spark. First Download the Spark source code from the Spark official website and extract the tar file.

Apache Maven

The Maven-based build is the build of reference for Apache Spark. Building Spark using Maven requires Maven 3.3.9 or newer and Java 7+.

Setting up Maven's Memory Usage

You'll need to configure Maven to use more memory than usual by setting MAVEN_OPTS:

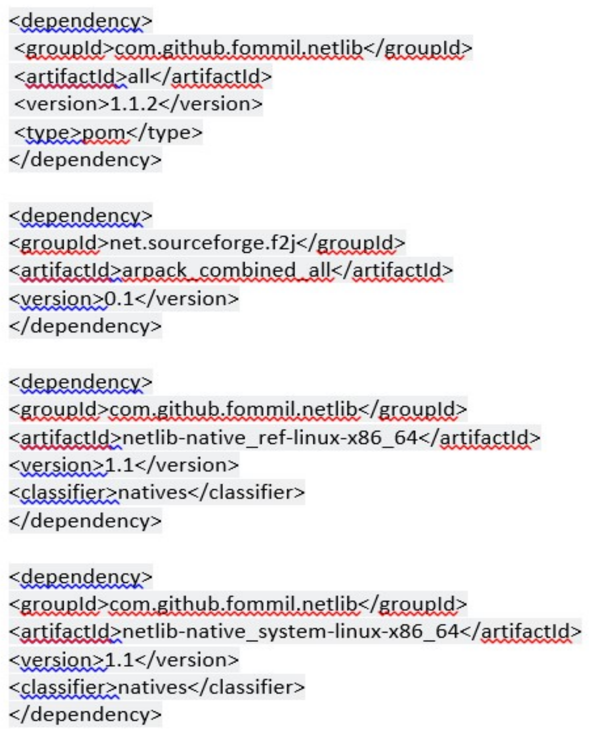
```
export MAVEN_OPTS="-Xmx2g -XX:ReservedCodeCacheSize=512m"
```

Adding the BLAS AND ARPACK dependencies

OpenBLAS is an optimized BLAS library .First install the openblas library to your system using the following command.

```
sudo apt-get install libopenblas-dev
```

This will install the FORTRAN libraries for the basic linear algebra subprograms packages which are required for the best performance of some the machine learning algorithms which will be used such as Eigen value decomposition etc. ARPACK, the ARnoldi PACKage, is a numerical software library written in FORTRAN 77 for solving large scale eigenvalue problems. This can be done by including a few of the dependencies to the pom file in the respective mllib and mllib-local directory . The following snippets of code are to be added to the dependencies in those pom files



```
<dependency>
 <groupId>com.github.fommil.netlib</groupId>
 <artifactId>all</artifactId>
 <version>1.1.2</version>
 <type>pom</type>
</dependency>

<dependency>
<groupId>net.sourceforge.f2j</groupId>
<artifactId>arpack_combined_all</artifactId>
<version>0.1</version>
</dependency>

<dependency>
<groupId>com.github.fommil.netlib</groupId>
<artifactId>netlib-native_ref-linux-x86_64</artifactId>
<version>1.1</version>
<classifier>natives</classifier>
</dependency>

<dependency>
<groupId>com.github.fommil.netlib</groupId>
<artifactId>netlib-native_system-linux-x86_64</artifactId>
<version>1.1</version>
<classifier>natives</classifier>
</dependency>
```

These dependencies ensure that while building it extracts the necessary libraries from the sources specified. This will ensure that we won't get implementation errors when spark is trying to run the SVD operation on big data and also keeps the performance very optimal .Now navigate to the spark folder and follow the build instructions given below

build/mvn

Building spark using maven or sbt is preferred because it gives you the additional customization options which can help you choose specific libraries and packages that can be used for any specific task, by changing and adding those specific dependencies to the pom file. Spark comes with an easy default build for maven, we could also use sbt to get the same results to help in building it. This will automatically set up the required environment such as the Scala version and more which Spark utilizes. You can build spark using the following command:

```
./build/mvn -Pyarn -Phadoop-2.7 -Dhadoop.version=2.7.0 -DskipTests clean package
```

This should be executed in the spark directory wherever it was extracted.

Required cluster configurations for Spark

Let us now discuss the configuration of the cluster for the Spark job using the Spark shell, we first have to set the required options to the Scala shell to give the cluster settings in order for it to run. Let's first discuss the hardware settings of each of the worker but also let's see the required changes to make in the configuration file before running.

Note: Due to hardware limitations we are using a 48 gb server so all the worker nodes will be launched on the server itself in a cluster mode. First make sure that there is a copy of all of the given files

slaves.template to slaves

spark-env.sh.template to spark-env.sh

spark-defaults.conf.template to spark-defaults.conf

Now add the following lines to the following files respectively

In slaves add

```
# A Spark Worker will be started on each of the machines listed below.

 master
```

In spark-env.sh add

```
export SPARK_MASTER_MEMORY="8g"
export SPARK_DRIVER_MEMORY="48g"
export SPARK_WORKER_INSTANCES="6"
export SPARK_EXECUTOR_INSTANCES="6"
export SPARK_WORKER_MEMORY="8g"
export SPARK_EXECUTOR_MEMORY="8g"
export SPARK_WORKER_CORES="16"
export SPARK_EXECUTOR_CORES="16"
```

In spark-default.conf add

```
spark.master                spark://master:7077
```

Note : We are assuming the named IP address of the system to be master , please do change the name if you're using any other name in the /etc/hosts file for your respective IP address accordingly Starting your cluster

In our demonstration cause of hardware constrains we are using only one system to launch all our worker nodes and master node alike so we will be using the following command

```
/<sparkdirectory>/sbin/start-all.sh
```

In case you have other workers in other slave computers then make sure you start them from other computers respectively by using

```
/<sparkdirectory>/sbin/start-slaves.sh <master-IPaddress>
```

Then check : master:8080 in any web browser to see if your workers have successfully started

For the code used check it out on GitHub :

https://github.com/rahul-aedula95/Mahout-spark_SVD

https://github.com/rahul-aedula95/singval

Key Terminology & Definitions

Apache Software Foundation Projects: Apache provides support for the Apache Community of open-source software projects. There are over 140 different Apache project communities that provide free software for the public benefit. About 35 open source Apache projects are dedicated to Big data, namely Hadoop, Spark, Pig, Mahout, Hive, Sqoop, Storm etc.

Data collection/acquisition: Data collection is the process of gathering and measuring information on targeted variables in an established systematic fashion, which then enables one to answer relevant questions and evaluate outcomes. The data collection is a primary component of research which is common to all fields of study including physical and social sciences, humanities and business.

Data Cleansing: This is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database.

Data visualization: This is a general term that describes any effort to help people understand the significance of data by placing it in a visual context. Patterns, trends and correlations that might go undetected in text-based data can be exposed and recognized easier with data visualization software.

Data analysis: This is a process of inspecting, cleaning, transforming, and modeling data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making.

Machine Learning: This is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data.

AWS: Amazon Web Service is a platform which provides cloud data storage and remote computing services provided by Amazon.

CLUSTER BY: This is a Hive command used as CLUSTER BY x which ensures that each of the N reducers get a Non-overlapping Ranges. Then sorts those ranges at the reducers.

CSV: Comma separated Values is a file format primarily used in storing data in a tabular format. Each line of the CSV file indicates an entry in the record and the different fields of the record are separated using delimited values in this specific format which is comma (,) .

Hadoop: Hadoop is a framework by the Apache Software Foundation which provides an environment for distributed computing over a cluster of working nodes. This is primarily designed to handle batch processing for Big Data.

HDF5: Hierarchical Data Format is a file format extensively used in storing data which involves metadata. Most of these formats are popularly used in instrumentation data storage.

Hive: Hive is a tool provided by the Apache foundation which can be used in the Hadoop environment. This tool brings important elements such as Data Warehousing and SQL synchronously with Hadoop Map Reduce.

HSSF: This is an API provided by the Apache POI project. This provides the pure implementation of Java regarding Excel sheets. It provides functions to create, modify, write and read into Excel documents.

JHDF5: Also known as HDF5 for Java is a package which provides a high level interface which work over other packages which includes functionality and support mechanisms for handling and manipulating HDF5 file format.

JSON: Java Script Object Notation is a Data interchange format. It is extensively used for denoting data with various fields and it very easy to work with as there are many packages and API which are supporting it.

Mahout: Mahout is a tool provided by the Apache Software Foundation which integrates Machine learning and its relevant algorithms to work in a Hadoop environment. It tries to utilize the potential of the distributed computation of a Hadoop Cluster.

MYSQLdb: This is a package which provides API's in Python to communicate with the MYSQL database. We can perform various databases manipulation such as querying and inserting records et cetera.

NoSQL: Traditionally termed as Non SQL which is used to associate data storing mechanism which holds data without involving tables and other relational database concepts.

NUMPY: This is a package or library provided by Python which includes several Matrix and Array Manipulation functions which make it much simpler to use. It is widely used by data science community for high end Matrix operations.

ORDER BY: This is a Hive command which is derived from SQL, this basically performs sorting of data in ascending or descending order. In hives it guarantees the total ordering of data.

Pandas: This is a high performance library provided by Python for easy to use functions which help in data analysis and data structuring.

PCA: Principle Component Analysis is a statistical method used to obtain a number of uncorrelated data sets. The main purpose of PCA is to identify maximum variance using less number of prominent components to represent it.

POI: This is an Apache Project which was developed for the primary purpose of modifying Microsoft Office files. It provides a set of functions and API's which can interact with various different file formats for modifying, such include HSSF which is used to handle Excel files in Java.

RDBMS: Relational Database Management Systems is a program which acts as an interface to a Relational Databases. A Relational Database is a storage unit which holds data in a tabular format and maintains a type of link or relationship among the various fields of data.

RDD: Resilient Distributed Data set can be defined as a data structure widely used in Spark. Its main characteristics are it is fault tolerant, it can be parallelized easily and it is an immutable collection of objects.

Scikit-Learn: This is a Machine learning Library made for Python which is built using NUMPY, SCIPY and MATPLOTLIB. This favors an easy implementation of Machine Learning Algorithms such as Classification, Regression and Dimensionality Reduction without any cumbersome code.

Spark: Apache Spark is an Open source project which is built as a framework for data processing more specifically big data. It is designed in a manner to enhance clustering capabilities to perform more complex algorithms much faster and more resource efficient compared to the other frameworks.

Sqoop: Sqoop is a tool which is built by the Apache Software Foundation. Its main purpose is transfer data from a Relational Database to a NoSQL database. It is effectively used in the Hadoop Ecosystem to transfer data from MYSQL and such databases to Hive and PIG.

UNION ALL: This is a command used in Hive which is used to combine all results of Select statements into one result.

Authors Bio:

Gouri Ginde, MTech: Gouri Ginde holds Master's Degree in Computer Science and Engineering. She has over 10 years of industry experience broadly based on Big data, Data analytics and Machine learning. She is currently working as a Research Associate at Centre for Applied Mathematical Modeling and Simulation (CAMMS), Department of Computer Science and Engineering, PESIT Bangalore South Campus, Bangalore

E-Mail: gouriginde@pes.edu

Affiliation: Department of Computer Science and Engineering
PESIT South Campus Bangalore, India

Rahul Aedula: Rahul Aedula is currently pursuing his Bachelors of Engineering in Computer Science at PESIT South Campus, Bangalore. He is currently working as an undergraduate research assistant in PESIT. His area of research includes Data Science, Machine Learning and Big Data Analytics.

E-Mail: rahulaedula95@gmail.com

Affiliation: Department of Computer Science and Engineering
PESIT South Campus Bangalore, India

Snehanshu Saha, PhD: Dr. Saha holds Master's Degree in Mathematical Sciences at Clemson University and Ph.D. from the Department of Mathematics at the University of Texas at Arlington in 2008. He is a Professor of Computer Science and Engineering at PESIT South since 2011 and heads the Centre for Applied Mathematical Modeling and Simulation. He has published 40 peer-reviewed articles in International journals and conferences and been IEEE Senior member and ACM professional member since 2012. Snehanshu's current and future research interests lie in Data Science, Machine Learning, Big data and applied computational modeling.

E-Mail: snehanshusaha@pes.edu

Affiliation: Department of Computer Science and Engineering
PESIT South Campus Bangalore, India

Archana Mathur: Archana Mathur has completed her Bachelors in Engineering in Computer science from Rajasthan University and received Masters in Web Technologies from Bangalore University, Bangalore. She is working as Assistant Professor in PES Institute of Technology Bangalore South Campus and has 8 years of teaching and research experience in academics. She is pursuing PhD in scientometrics and Machine Learning and published in high impact factor journals.

E-Mail: archanamathur@pes.edu

Affiliation: Department of Computer Science and Engineering
PESIT South Campus Bangalore, India

Sudeepa Roy Dey : Sudeepa Roy Dey, Faculty, PESIT-BSC is a Bachelors and Master's degree holder in Computer Science and Engineering. She has eight years of teaching and research experience. Her research areas include Machine learning, Cloud Computing and Optimization techniques.

E-Mail: sudeepar@pes.edu

Affiliation: Department of Computer Science and Engineering
PESIT South Campus Bangalore, India

Gambhire Swati Sampatrao : Gambhire Swati Sampatrao, faculty, PESIT-BSC holds Bachelor's and Master's degree in Computer Science and Engineering. She has a teaching experience of 6 years in various technical institutions. Her research areas include Cloud Computing, Machine Learning and convex optimization and graph theoretic algorithms.

E-Mail: swatigambhire@pes.edu

Affiliation: Department of Computer Science and Engineering
PESIT South Campus Bangalore, India

BS Daya Sagar, PhD: B. S. Daya Sagar received Ph.D. from the Faculty of Engineering, Andhra University, India, in 1994. Sagar is a Full Professor at Systems Science and Informatics Unit (SSIU) of Indian Statistical Institute—Bangalore Centre, India. Earlier, he worked in at

National University of Singapore and Multimedia University, Malaysia during 1998-20007. His research interests include GIS, Mathematical Morphology, and Fractals. He has published over 75 papers in journals, and has authored and/or guest edited 9 book and/or theme issues for journals. He authored a book entitled "Mathematical Morphology in Geomorphology and GISci," CRC Press: Boca Raton, 2013, p. 546. He is an elected Fellow of Royal Geographical Society (1999), Indian Geo-physical Union (2011). He received Dr. Balakrishna Memorial Award from Andhra Pradesh Akademi of Sciences in 1995, Krishnan Gold Medal from Indian Geo-physical Union in 2002, and 'Georges Matheron Award-2011 (with Lecturership)" of International Association for Mathematical Geosciences.

E-Mail: bsdsagar@isibang.ac.in

Affiliation: Systems Science and Informatics Unit,
Indian Statistical Institute-Bangalore Centre