DOCUMENT SELECTION USING MAPREDUCE

Yenumula B Reddy[$] and Desmond Hill[#]

Department of Computer Science, Grambling State University, Grambling, LA 71245, USA
$ybreddy@gram.edu; #desmondhill247@yahoo.com

ABSTRACT

Big data is used for structured, unstructured and semi-structured large volume of data which is difficult to manage and costly to store. Using explanatory analysis techniques to understand such raw data, carefully balance the benefits in terms of storage and retrieval techniques is an essential part of the Big Data. The research discusses the MapReduce issues, framework for MapReduce programming model and implementation. The paper includes the analysis of Big Data using MapReduce techniques and identifying a required document from a stream of documents. Identifying a required document is part of the security in a stream of documents in the cyber world. The document may be significant in business, medical, social, or terrorism.

KEYWORDS

HDFS, MapReduce, Big Data, Document identification, client node, data node, name node;

1. INTRODUCTION

Big data is a general term used for a large volume of data that are structured, unstructured and semi-structured data created or generated by a company [1]. This data cannot be loaded using any database models and is not possible to get results with available query languages. That means the big data cannot be processed using traditional tools. The data is not unique quantity in terms of many bytes (terra-bytes or petabytes or exabytes). It is continuously growing every minute. It is real big and grows in exabytes. The origins are from a variety of platforms. Volume changes quickly and cannot be predicted its growth. It is expensive to analyze, organize, and to prepare as useful data. Therefore, we need a special effort to prepare as meaningful by using new algorithms, special hardware, and software. There may not be a single tool to analyze and create big data as meaningful data. The tools may vary to analyze one type of data to another. The big data management requires generating high quality of data to answer the business queries.

The primary goal is to discover the repeatable business patterns, uncover the hidden patterns, and understand the unknown correlations and other useful information. The business giants have access to the information, but they do not know to get the value out of it. The traditional tools are not helpful due to semi-structured storage. The big data analytics may commonly use software tools such as predictive analytics and data mining. The technology associated with big data analytics includes NoSQL databases, Hadoop, and MapReduce [2]. There are open source frameworks that support processing of big data. The analysis requires MapReduce to distribute the work among a large number of computers and processes in real-time. Hadoop structure lowers the risk of catastrophic system failure, even the significant number of nodes become inoperative.

The Government and information technology (IT) managers are highly motivated to turn big data into an asset so that the data can be used to meet their business requirements. Today Hadoop framework and MapReduce offer new technology to process and transform big data into meaningful data. It is required to deploy the infrastructure differently to support the distributed processing and meet the real-time demands. The IT managers found that security and privacy are the major problems particularly while using a third party cloud service providers.

The structured and unstructured data comes from a variety of sources. Adoption of big data tools to process the big data is increasing. High priority is given to improve the big data formalizing and processing. The top data for transactions include business data documents, email, sensor data, image data, Weblogs, Internet search indexing, and attached files. The IT group found that interest in learning about technology, deploy the packages to process the data and adopt the infrastructure for better performance and affordable cost. They are in the process of implementing Apache Hadoop frameworks, commercial distributions of the Hadoop framework and other NoSQL databases.

Currently, priority is given to processing and analyzing unstructured data resources including Weblogs, social media, e-mail, photos, and videos. Unstructured emails are given priority to analyze and process. As a first step, the IT staff is working on batch processing and move to real-time process. The industries and Government are concerned about the scalability, low latency, and performance in storing and analyzing the data. Further, they are worried about the protection of data for third-party cloud providers. The data privacy, security, and interoperability standards are necessary for data and systems management.

The big data is useful if analyzed, stored in a meaningful way and accessed quickly. The main goal is to store the data to ensure that the information is accessible to business intelligence and analysis. It is required to design a tool to analyze the data and provide the answers with minimum efforts and time. The challenges include the size, structure, origin, variety, and continuous change in data. The data is real big in size (terra-bytes or eta-bytes) and unstructured. It contains text, figures, videos, tweets, Facebook posts, website clicks, and various types of data from a variety of websites. The origins are varied and come from a variety of platforms and multiple touch points. Data changes quickly in terms of format, size, and types of Websites. The software required to pull, sort, and make the data meaningful. There is no universal solution to make such data meaningful. The data produced is in universally available languages. Processing such data may need separate algorithms (depending upon the type of data). There are multiple predictions in years to come about the data coming from an unknown source and unstructured in nature. The following predictions include the data origin, type, size, and management.

- Massive data collection across multiple points
- Firmer Grip on the data collected by various groups
- Generalization of format for Internet data and data generated by business media
- Entering of social media as part of Big Data generation

The main purpose of big data management is to make sense of the collected data by analyzing and processing that collected data from various data collection points. Making sense of data means that the end point of processing of data must be able to answer the business queries. The queries include data mining related predictions, business queries, and management assistance.

The Hadoop project was adopted more in the DoD (Department of Defense) than in other agencies. The problem in Hadoop system design is the lack of reusability of existing applications since the

Apache Hadoop defines the new Java Application Programming Interfaces (API's) for data access. Research is in progress in understanding the Hadoop system design, usage, and security status. Protocol level modifications help to improve the security at the level.

Federated systems enable collaboration of multiple systems, networks, and organizations with different trust levels. Authentication and authorization procedures of a client must be kept separate from services. Traditional security is about protecting the resources within the boundary of the organization. The federated systems require the security specification for each function. Further, the federated systems change over time with new participants joining may not trustworthy. Therefore, individual protection domains are necessary for each entity. Boundary constraints are required depending upon the system. Therefore, monolithic security domain does not work in federated systems.

Existing federated security system separates the client access, associated authentication, and authorization procedures. The distribution of federated systems needs collaboration between networking, organizations and associated systems. Maintaining security among these is not an easy task since multiple entities are involved. The security threat may be unavoidable from a variety of sources. Therefore, a high-level coarse-grained security goals need to be specified in the requirements.

2. RECENT DEVELOPMENTS AND MAPREDUCE

Agarwal discussed the data backup in Hadoop file systems using snapshots [1]. The authors designed an algorithm for a selective copy on appends and low memory overheads. In this work, the snapshot tree and garbage collection was managed by Name-node. Shvachko et al. [2] discussed the architecture of Hadoop Distributed File Systems (HDFS) and the experiences to manage 25 petabytes of Yahoo data. The fault tolerant Google file system running on inexpensive commodity hardware with aggregate performance to a large number of clients was discussed in [3]. The paper discusses many aspects of design and provides report measurements from both micro-benchmarks and real world use. The MapReduce programming model was explained, and it is easy to use functions were discussed in [4]. The document discussed the experiences and lessons learned in implanting the model. Further, it discusses the impact of slow machines in redundant execution, locality of optimization and writing single copy of the immediate data to local disc, and saving the network bandwidth. Chang et al. [5] discussed the dynamic data control over data layout and format using Bitable as a flexible solution. They claimed that many projects were successfully using this model by adding more machines for process over the time.

The deployment of federated security architecture was discussed in Windows Communication Foundation (WCF) [6]. WCF provides support for building and deploying distributed systems that employ federated security. Domain/realm, Federation, and security token service consist of the primary security architecture of federated systems. The current mobile devices were implemented with the imitation of 60K tasks, but the next generation of Apache MapReduce supports 100K concurrent tasks [7]. The users in MapReduce specify the computation in terms of the map and a reduce function. The underlying scheme in MapReduce package automatically paralyzes the computation and schedules the parallel operations so that the task uses the network and computational facilities to perform the operations faster. An average of hundred thousand MapReduce jobs every day are executed on Google clusters.

Halevy et al. [8] suggested that representing the data of large data sources with a nonparametric model is needed compared to summarizing parametric model because the large data sources hold many details. The authors felt that choosing unsupervised learning on unlabeled data is more powerful than on labeled data. They pointed out that creating required data sets by automatically combining data from multiple tables is an active research area. Combining data from multiple tables with data from other sources, such as unstructured Web pages or Web search queries, is an important research problem. Various types of security policies such as local, component, generic, export and federate was discussed in [9]. The security policy generation enforcing also included in this study.

Hadoop security was discussed in the reports [10 - 15]. Reddy proposed the security model for Hadoop systems at access control and security level changes depending upon the sensitivity of the data. Authentication and encryption are the two Security levels for Big Data in Hadoop [11]. The author in [11] describes that Kerberos file system has better protection to keep the intruders away from accessing the file system. Chary et al. [12] discussed the security in distributed systems and current level of security in Hadoop systems that include the client access to Hadoop cluster using Kerberos Protocol and authorization to access.

O'Malley et al. [13-14] discussed the security threats from different user groups in Kerberos authentication system, the role of delegation token in Kerberos and MapReduce implementation details. The research emphasizes the need for internal and external security for Hadoop systems. The work describes the need for encryption, limiting access to specific users by application, isolation between customers, and information protection.

The prototype system called "Airavat" a MapReduce-based system that provides strong security and privacy guarantees for distributed computations of sensitive data was presented in [15]. The model provides a method to estimate the different parameters that should be used and tested on several different problems. This system has weak use cases, a complicated process in specifying the parameters and is not efficient in the general composition of MapReduce computations (the MapR function followed by another mapper). Some organizations raised critical questions about privacy and trust in the system.

Preserving the privacy of Big Data was discussed by McSherry [16]. McSherry's Privacy integrated queries (PINQ) presents an opportunity to establish a more formal and transparent basis for privacy technology. The algorithms designed help the users in increasing the privacy-preserving and increases the portability. Partha et al. [17] presented a system that learns for data-integrated queries which uses sequences of associations (e.g. foreign keys, links, schema, mappings, synonyms, and taxonomies). The associations create multiple ranked queries linking the matches to keywords. The queries are related to Web forms, and users have only to fill the keywords to get answers.

3. MAPREDUCE PROGRAMMING MODEL

One of the programming models in MapReduce is breaking the large problem into several smaller problems. The solutions to the problems are then combined to give a solution to the original problem. The process is similar to software engineering top down model. There are many questions arise in MapReduce application due to the following reasons.

- dynamic input
- nodes may fail

- number of smaller problems may exceed the number of nodes
- dependable sub-problems
- distribution of input to the smaller jobs
- coordination of nodes
- synchronization of the completed work

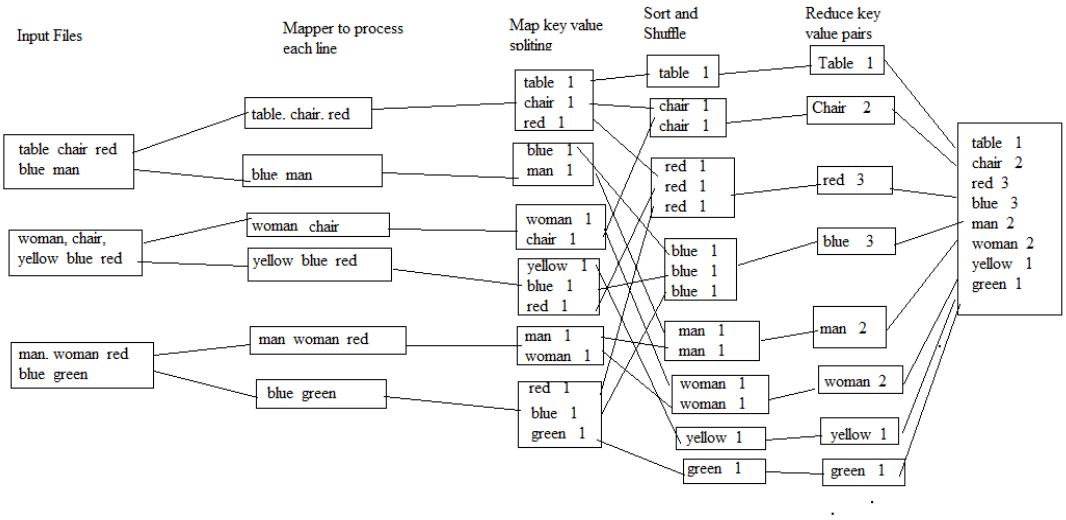


Figure 1: Map, Shuffle, and Reduce Phase

To solve these problems we can use MapReduce programming model and an associated implementation for processing and generating large data sets. MapReduce application executes in three steps. They are Map, Shuffle & Sort, and Reduce. A MapReduce job divides the input data-set into independent chunks that are processed by the map tasks in a completely parallel manner. A given input pair can have zero or more output pairs. The map outputs are merged and constructed with respect to the key values. These pairs are propagated to reduce function. The reduce function then merges these values to form a possibly smaller set of values. That is, the reduce function filters the map output and generates the results with respect to the key. The total functionality includes scheduling the tasks, monitoring them and re-executes the failed tasks. Figure 1 shows the Map, Shuffle, and Reduce phase.

4. IMPLEMENTATION

HDFS is a distributed file system on top of existing operating system. It runs on a cluster of commodity hardware and has the capability of handling node failures. The files are fragmented in blocks typically of size 64/128 MB. HDFS is divided into data nodes and name nodes. Name node is HDFS master node and data node is slave node. The name node controls the distribution of files into blocks and responsible for balancing. Data node reads and writes HDFS blocks, direct communication with the client and report awareness to the name node. Figure 2 shows the relation between Hadoop client, name node, and data node.

Command line interface, Java API, Web interfaces are known interfaces for HDFS. The following instructions are used for the HDFS command line.

Create a directory

```
$ hadoop fs -mkdir /user/idcuser/data
```

Copy a file from the local filesystem to HDFS

```
$ hadoop fs -copyFromLocal cit-Patents.txt /user/idcuser/data/.
```

List all files in the HDFS file system

```
$ hadoop fs -ls data/*
```

Show the end of the specified HDFS file

```
$ hadoop fs -tail /user/idcuser/data/cit-patents-copy.txt
```

Append multiple files and move them to HDFS (via stdin/pipes)

```
$ cat /data/ita13-tutorial/pg*.txt | hadoop fs -put - data/all_gutenberg.txt
```

File/Directory Commands:

copyFromLocal, copyToLocal, cp, getmerge, ls, lsr (recursive ls), moveFromLocal, moveToLocal, mv, rm, rmr (recursive rm), touchz, mkdir

Status/List/Show Commands:

stat, tail, cat, test (checks for existence of path, file, zero length files), du, dus

Misc Commands:

setrep, chgrp, chmod, chown, expunge (empties trashfolder)

The single node implementation was done on the Dell Precision T5500 with following configuration.

Hardware:

CPU: Intel (R) Xeon (R) 1.60 GHz

System Memory: 12 GB

GPU Chip: Two Quadro 4000

Software:

Ubuntu 14.04 (64-bit)

CUDA Version: 6.5

CUDA C, Python, FORTRAN

The computers have duel Quadro 4000 Graphics Processing Cards (GPU). The GPU facility was not used in the experiment. The research was conducted using Hadoop 2.6.0, JDK 7, and Python 3.4 on Dell Precision T5500 with Ubuntu 14.04 in Department of Computer Science Research Lab at

Grambling State University during spring and summer 2015. The process includes the following steps:

- Two computers were used to setup the single node Hadoop cluster
- Install Ubuntu 14.04, create Hadoop account. The Hadoop account helps to separate the Hadoop installation from other software applications and any other account running on the same machine.
- Configure the Secure Shell (SSH), generate key for the Hadoop account and enable SSH access to the local machine. Test the SSH setup by connecting the local machine with Hadoop account. Disable the IPv6 using text editor in /etc/sysctl.conf by modifying the code to keep running Hadoop package. Reboot the system.
- Update the required Hadoop core files as directed in document so that the single node clusters. Use JPS command to check the Hadoop single node cluster.

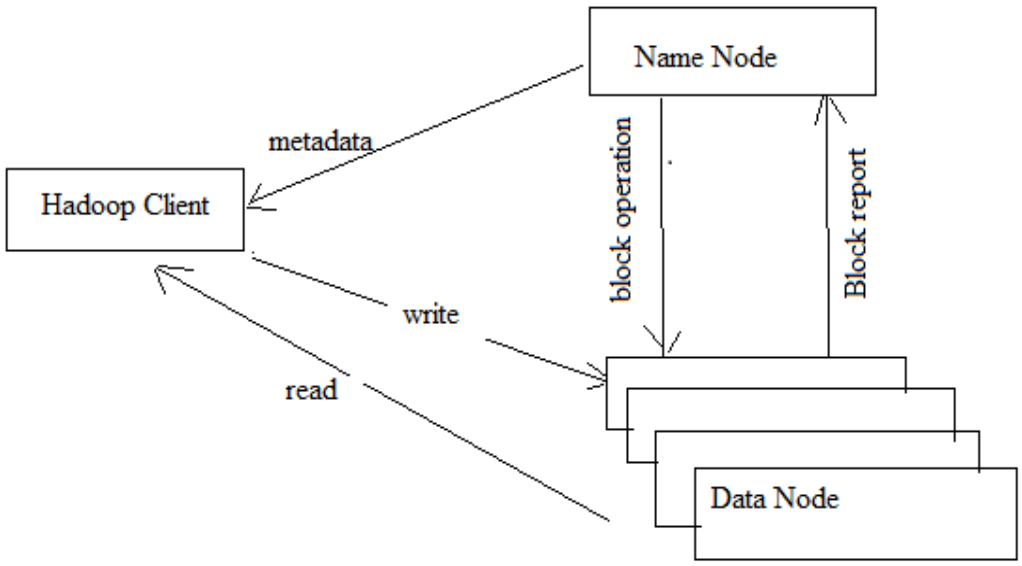


Figure 2: Relation between Hadoop client, Name Node, and Data Node

The code was developed to utilize the Hadoop's node distribution capabilities to analyze the text files. The analysis includes many times of each word repeats in the text file. Further, the program calculates the number of times the keyword repeats. Appendix A provides the Mapper and Reducer (Mapper.py and Reducer.py) written in Python. We provide the weight of each keyword generated by Mapper and Reducer file. The product of keyword with weight factor provides the importance of the keyword in the document. The sum of these products of all keywords with weight factors helps to select the document. For each document selection, we provide the threshold value. The programmer sets the threshold value. The threshold value determines the selection during the document screening process using MapReduce method. If the sum of the products with keywords exceeds the threshold, the document is required; otherwise the program rejects the document. The

search program in Appendix B helps to select the required document.

5. DISCUSSION OF RESULTS

To select a required document, we provide the keywords and their importance that varies between 0 and 1. We then take the important factor multiplied by the number of times keyword and sum the result of all keyword importance. If the sum is above the threshold value, we conclude that the document is required. The algorithm was coded in two stages. During the first stage, the reputation of the words and in the second stage the importance factor and selection of the document were coded in Python. We processed six text files to compare the results of the current experiment. The keywords and impact factor provided are: medicine (0.02), profession (0.025), disease (0.02), surgery (0.02), mythology (0.02), and cure (0.05). The size of each file in words, times taken in seconds to process and impact factors respectively are: (128,729; 0.1539; 5.39), (128,805; 0.1496; 0.62), (266,017; 0.13887, 0), (277,478; 0.1692; 6.02), (330,582; 0.1725; 7.93), and (409,113; 0.2032; 18.87). The threshold set was 10.0. Therefore, the file with impact factor 18.87 is selected as required file. If we lower the threshold to 5.0 another two files with impact factors 6.02 and 7.93 would become our required files.

6. CONCLUSIONS AND FUTURE RESEARCH

The current research discusses the implementation of Hadoop Distributed File system and selection of required document among the stream of documents (unstructured data). Setting up Hadoop single node cluster on two machines and selection of required document among the stream of text documents was completed. The process did not include any data models or SQL. It is simple Hadoop cluster, Map Reduce algorithm, required keys and their importance factor. The future research includes the multi-node cluster and High-performance computing using Graphics Processing Units (GPUs) for real-time response.

ACKNOWLEDGEMENTS

The research work was supported by the AFRL Collaboration Program – Sensors Research, Air Force Contract FA8650-13-C-5800, through subcontract number GRAM 13-S7700-02-C2.

REFERENCES

1. Sameer Agarwal., Dhruba Borthakur., and Ion Stoica., “Snapshots in Hadoop Distributed File System”, UC Berkeley Technical Report UCB/EECS, 2011.
2. Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler., “The Hadoop Distributed File System”, 26th IEEE (MSST2010) Symposium on Massive Storage Systems and Technologies, May, 2010.
3. Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung., “The Google File System”, *SOSP’03,* October 19–22, 2003.
4. Jeffrey Dean and Sanjay Ghemawat., “MapReduce: Simplified Data Processing on Large Clusters”, Proc. 6th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2004, San Francisco, USA, Dec. 2004.
5. F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, R. E. Gruber., “Bigtable : A Distributed Storage System For Structured Data”, ACM Transactions on Computer Systems (TOCS), Volume 26 Issue 2, June 2008.
6. Athontication, Microsoft Patterns & Practices, http://msdn.microsoft.com/en-us/library/ff649763.aspx 2012 [accessed: April 2013].

7. J. Dean and S. Ghemawat., “MapReduce: simplified data processing on large clusters”, CACM 50th anniversary issue, Vol. 51, issue 1, Jan 2008, pp. 107-113.
8. A. Halevy, P. Norvig and F. Pereira., “The Unreasonable Effectiveness of Data”, IEEE Intelligent Syst., 2009, pp. 8-12.
9. B. Thuraisingham., “Security issues for federated database systems”, Computers & Security, 13 (1994), pp. 509-525.
10. Yenumula B. Reddy., “Access Control for Sensitive Data in Hadoop Distributed File Systems”, Third International Conference on Advanced Communications and Computation, INFOCOMP 2013, November 17 - 22, 2013 - Lisbon, Portugal.
11. P. Ravi, “Security for Big Data in Hadoop”, http://ravistechblog.wordpress.com/tag/Hadoop-security/, April 15, 2013 [Retrieved - April 2013].
12. N. Chary., Siddalinga K M., and Rahman., “Security Implementation in Hadoop”, http://search.iiit.ac.in/cloud/presentations/28.pdf [accessed: January 2013].
13. O. O’Malle., K. Zhang., S. Radia., R. Marti., and C. Harrell., “Hadoop Security Design”, http://techcat.org/wp-content/uploads/2013/04/Hadoop-security-design.pdf, 2009, [accessed: March 2013].
14. D. Das., O. O’Malley., S. Radia., and K. Zhang., “Adding Security to Apache Hadoop”, hortonworks report, http://www.Hortonworks.com
15. I. Roy Srinath, T.V. Setty, A. Kilzer, V. Shmatikov, and E. Witchel, “Airavat: Security and Privacy for MapReduce”, 7th USENIX conference on Networked systems design and implementation (NSDI'10), 2010, Berkeley, CA.
16. Frank McSherry., “Privacy Integrated Queries: An Extensible Platform for Privacy-Preserving Data Analysis”, Proceedings of SIGMOD, 2009.
17. Partha Pratim Talukdar , Marie Jacob , Muhammad Salman Mehmood , Koby Crammer , Zachary G. Ives , Fernando Pereira , Sudipto Guha., “Learning to Create Data-Integrating Queries”, VLDB, 2008.

Appendix A

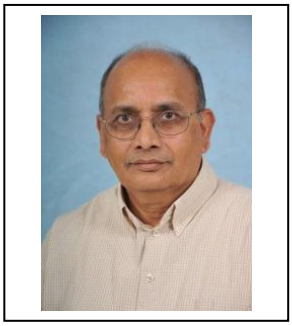
```
#!/usr/bin/python
import sys
import time
search_words = {'cure':0.05,'disease':0.02,'medicicne':0.02,
'mythology':0.02,'surgery': 0.02,'mythology': 0.02}
startTime = time.time()
counter = 0
for line in sys.stdin:
   for word in line.strip().split():
      for counter in range(len(search_words)):
         if word == search_words.keys()[counter]:
            print "%s\t%d" % (word, 1)
```

Appendix B

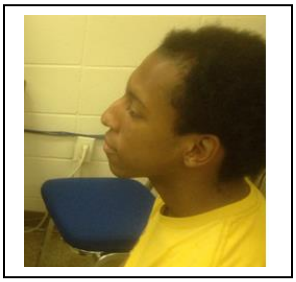
```
#!/usr/bin/python
import sys
import time

current_word = None
current_count = 1
word_names = []
word_amounts = []
x = 0
j = 0
importance = 0
search_importance = 0
totalTime = time.time()
print("\n-----------Word Occurrence--------------\n")
for line in sys.stdin:
    word, count = line.strip().split('\t')
    if current_word:
        if word == current_word:
          current_count += int(count)
        else:
          word_names.append(current_word)
          print "%s\t%d" % (current_word, current_count)
          word_amounts.append(current_count)
          current_count = 1
   current_word = word
if current_count > 0:
   word_names.append(current_word)
   print "%s\t%d" % (current_word, current_count)
   word_amounts.append(current_count)
#adds last searched word
print("\n-----------Word Importance---------- \n")
```

```
from word_mapper import search_words
from word_mapper import startTime
startTime = time.time() - startTime
for i in range(len(word_names)):
    importance = search_words[word_names[x]] * word_amounts[x]
    search_importance += importance
    print "%s\t%.2f" % (word_names[x], importance)
    x = x + 1
print("\n----------Search Stats---------- ")
print "%s%.2f" % ("\nSearch importance: ", search_importance)
totalTime = ((time.time() - totalTime) + startTime)
print "%s %.23f %s" % ("Search time: ", totalTime, "seconds.")
print("\n----------END----------")
```

Authors



Yenumula B. Reddy, Professor, IEEE Senior Member, IARIA Fellow and program coordinator of Computer Science, Grambling State University. ’ He is editorial board member of Journal BITM Transactions on EECC, Science Academy Transactions on Computer and Communications Networks (SATCCN), International Journal of Security, Privacy and Trust Management (IJSPTM), MASAUM Journal of Engineering and Applied Sciences (MJEAS), and International Journal of Engineering and Industries (IJEI). He is Editor-in-Chief of the International Journal of Security, Privacy, and Trust Management (IJSPTM).



Desmond Hill is an Undergraduate student at Department of Computer Science, Grambling State University. Currently, he is junior Computer Science Major and working in the Research group Hadoop Distributed File Systems.