Implementation of Page Ranking using Hadoop

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

By

Shriya Sharma (141337) and **Divjot Kaur (141356)**

Under the supervision of

Mr. Amit Jakhar

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work represented in this report entitled **"Implementation of Page Ranking Algorithm using Hadoop"** in partial fulfillment of the requirements for the award of the degree of **Bachelor in Technology in computer Science and Engineering** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2017 to May 2018 under the supervision of **Mr. Amit Jakhar** (Assistant Professor, Grade-II, Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)	(Student Signature)
Shriya Sharma, 141337	Divjot Kaur,141356

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
Supervisor Name
Designation
Department name
Dated:

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the completion of the project would be incomplete without the mention of the people who made it possible.

We would like to take the opportunity to thank and express our deep sense of gratitude to our faculty mentor Mr. Amit Jhakkar for providing their valuable guidance at all the stages of the study, their advice, constructive suggestions, positive and supportive attitude and continuous encouragement, without which it would have not been possible to complete the project. The blessing, help and guidance given by him from time to time shall carry us a long way in the journey of life in which we are about to embark.

We are obliged to all our faculty members of JUIT, for the valuable information provided by them in their respective fields. We are grateful for their cooperation during the period of the project. We hope that we can build upon the experience and knowledge that we have gained and make a remarkable mark in the future.

CONTENTS

List of Abbreviations

S.No	Abbreviation	Full Form
1.	DB	Database
2.	IR	Information Retrieval
3.	JDK	Java Development Kit
4.	JRE	Java Runtime Environment
5.	JVM	Java Virtual Machine
6.	PR	Page Rank
7.	WCM	Web Content Mining
8.	WPR	Weighted Page Rank
9.	WSM	Web Structure Mining
10.	WUM	Web Usage Mining
11.	WWW	World Wide Web
12.	WWWW	World Wide Web Worm

List of Figures

List of Tables

ABSTRACT

With the massive explosion of information these days and those relying increasingly on search-engines to get all sorts of information they want, it's far turning into more and more hard for the search-engines to supply most applicable records to the users. The most famous set of rules used in processing net information i.e. webpages, is web page rating set of rules which intends to determine the importance of a website by means of assigning a weighting cost based on any incoming link to those webpages. However, such big quantity of internet statistics may result in computational burden in processing those pages rank set of rules. To remember the ones burden, we bear in mind a refined algorithm over disbursed structures the use of Hadoop MapReduce framework i.e. MRPageRank. The Hadoop MapReduce is adopted because of its potential to technique large statistics over allotted machine without thinking about any detail troubles which includes scheduling, synchronizing and job failure. This algorithm has been decomposed into three tactics, every of which is carried out in a single Map and Reduce job. Parsing the webpage input to produce name of page as key and its outgoing links as value pair is the first part of the algorithm, including the calculation of the total weight of page without any outgoing node referred to as dangling node and quantity of all pages. Second is to calculate the probability of every page and calmly distribute this to each of the outgoing links. Each of the outgoing weight is shuffled and aggregated. This is based totally at the similarity of web page name to update a brand-new weighting price of each page. Third and the last part is to sort in the descending order based on their weighting values. For performance measurement of the proposed approach, response time and efficiency are considered.

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

With the appearance of digital information, retrieving data has become a bottleneck in information sharing and integration. The net has delivered significant impact in every component of human existence i.e. schooling, healthcare, enterprise, economic system etc. world extensive internet is considerable and growing at a remarkable priced WWW serves as a chief source of having such data.

The changing nature if internet i.e. masses of internet pages are deleted and delivered newly as a result retrieving green, relevant and significant data from those huge resources of facts is a completely challenging task. Due to the large length of the internet, similarly to any query made by way of a person variety of pages are being retrieved so that the result must be ordered in the collection in which the maximum relevant net Pages are at the top of the resulted list.

The internet contains millions and billions of web pages and multitudinous hyperlinks. These hyperlinks are analysed to discover concealed human interpretations that can be extremely valuable for inferring user behaviour. User interactions with search engines can serve as a valuable source of information for enhancing web search result ranking and can complement costlier explicit judgement.

One of the initial search-engines World Wide Web Worm came into existence in the early 1990s. It claimed that it indexed 110,000 web pages in 1994 and from then on, a variety of search engines claimed to be indexing documents up to a round figure of 100 million. A hike was seen in the user queries from 1500 per day to 20 million per day in the time span of 3 years; showing the exponential growth of internet users during the period of 1994 to 1997.

These search-engines did not take any kind of algorithms into account for the process of information retrieval. This resulted in a lot of challenges, but since the user, unaware of what was in the package in following years, was just satisfied with the results he was getting. The present scenario of Web Mining - an active research area, is mining of data available on WWW DB in the form of web pages to find hidden information. The Hidden information is the apprehension that could be contained in the content of web pages or in the link structure of WWW. As a result of the heterogeneity and lack of Web data structure, automated discovery of targeted or unanticipated information is a challenging task.

1.1.1 Web Mining

Web mining is the application of the data mining techniques that could be used to solve the information overload problems above directly or indirectly. Additionally, it identifies the pattern from WWW data and has proved to be very useful to e-commerce websites and e-services. It can be classified in three categories Web Content Mining(WCM), Web Structure Mining(WSM), and Web Usage Mining(WUM).

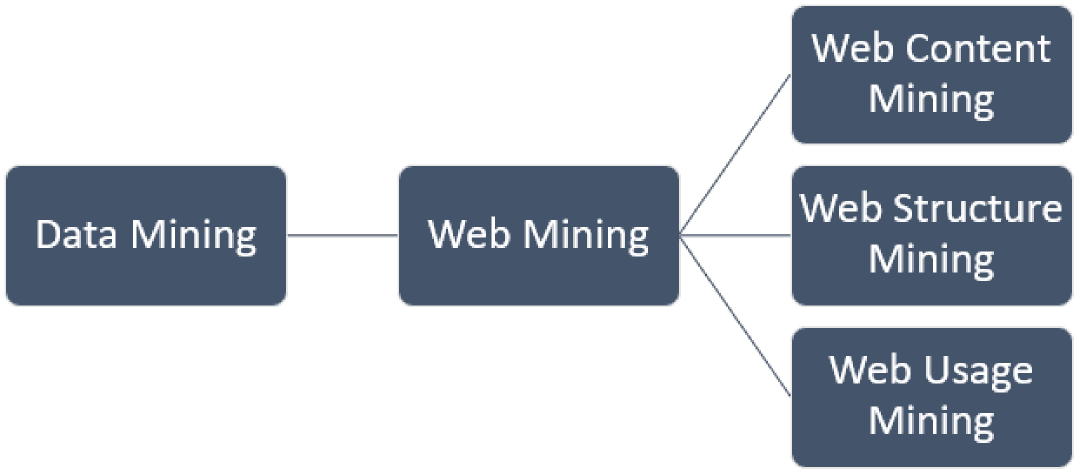


FIGURE 1.1: Classification of Web mining

- Web Content Mining

WCM is the mining, extracting and integrating relevant data and information from the content of web page. It is somehow related but still different from data and text mining, different in the context as it uses structured and unstructured data respectively. Web data is semi structured in nature. Hence, WCM uses the creative application of both data and text mining. There are two perspectives from which WCM is differentiated from other tools – Information Retrieval View and Database View.

- Web Structure Mining

  WSM is the process of analyzing the structure of a website i.e. nodes and connection structure through graph theory. We can get two things from this website first is how is the website connected from other websites and the document structure i.e. how the pages are connected.

- Web Usage Mining

  WUM looks on the patterns and information from the server logs. It extracts the patterns and checks on the activity of user from server for example, where is the user from, user clicked on what item and how many times. Basically, it checks every activity of user. This is done in three steps namely; preprocessing, pattern discovery and pattern analysis.[5]

These three categories have their own application areas such as site modification, web personalization, usage classification and characterization, ranking of pages etc. The ranking of pages is used by the search engines to find the most useful pages.

Some of the challenges confronted by Web Mining:

- Complexity of WWW: incorporates large amount of unstructured records and hence makes searching an incredibly complex task [4].
- Dynamic Nature of Internet
- User Diversity
- Relevant Information is time consuming

1.1.2 Page Ranking

With the drastic growth of the information on the internet, the ranking has become an undoubtedly hard task. Thus, the major goal of the search engines if to provide its users with an interactive interface which retrieves the relevant information in an ordered way. Page ranking has been classified into two categories by the search engine. The first is the query dependent, depends upon the frequency and the location of query terms. The second is question unbiased elements which take link recognition under consideration. Out of all the existing algorithms, Weighted Page Rank is the most appropriate to work with. [3]

Relevancy of the search engine consequences is measured with the aid of web page level keyword that is a vital aspect. End result set retrieved via serps are containing a huge quantity of vain net pages. they have shown that with the aid of retrieving the result which is relevant for a particular query increases the precision. And based totally on page level keywords search engines like google and yahoo are evaluated on instructional queries. A. Jain .et.al [5] have defined various algorithms used for link analysis like Page Rank (PR), Weighted Page Rank (WPR), Hyperlink-Induced Topic Search (HITS) and CLEVER and comparison is completed amongst those algorithms[1][2].

A paper was published by Sergey Brin et al. titled ‘Anatomy of a Large-Scale Hypertextual Web search-engine’ which described a search engine developed called Google and the algorithm that was developed for Google – PageRank algorithm. This search-engine was designed to scale efficiently to enormously large data sets. The growth seen in the web and the speed at which technology changes were the main factors responsible in developing Google.

PR works by means of counting the quantity and the quality of hyperlinks pointing to a page, so that it will determine a rough estimate of the way important the net-web page is. This algorithm works on an assumption that extra important web sites(webpages) are likely to get hold of more links from other webpages.

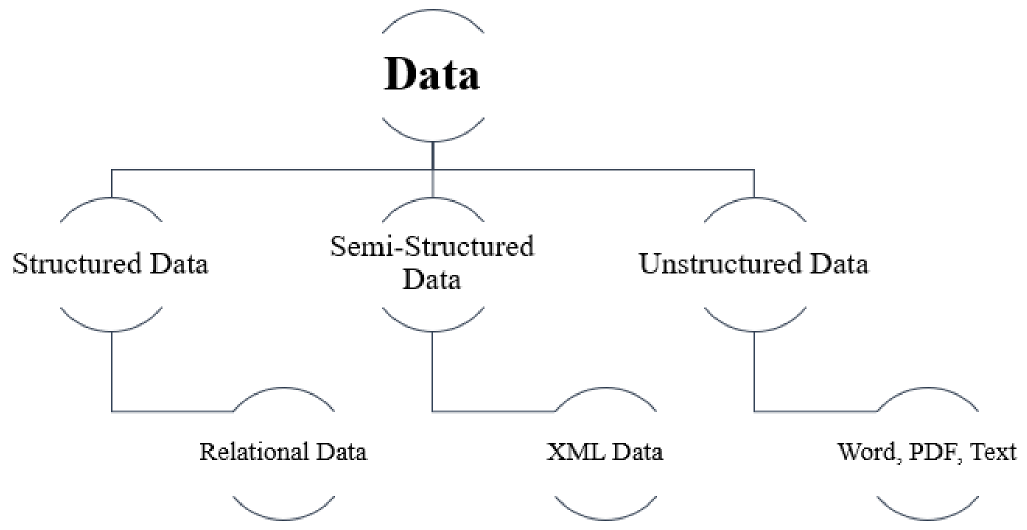


FIGURE 1.2: Classification of Data

1.1.3 Big Data

Taking literally, it is a collection of large datasets which are impossible to be processed the usage of conventional computing techniques. Big Data involves various tools, frameworks and techniques which are required to deal with huge volume, high velocity and extensible variety of data.

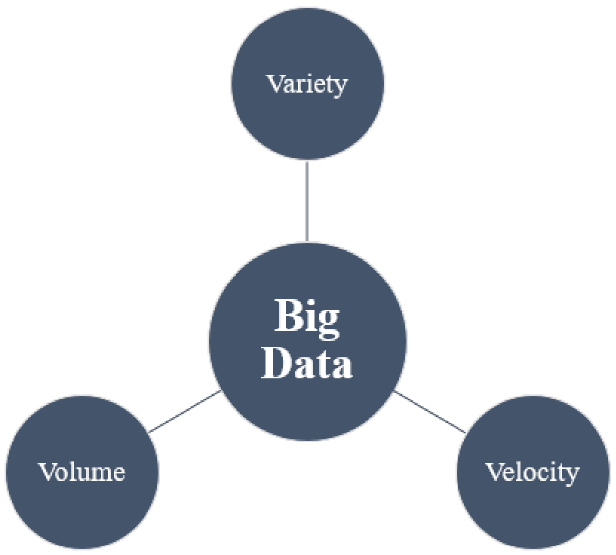


FIGURE 1.3: Three V’s of Big Data

Describing the data in 3 parameters-

- Velocity: The data is enormously increasing, and it has been estimated that the volume of data will double in every 2 years.
- Variety: As a result of millions and billions of chunks of data, traditional method of storing data in rows and columns proved to be fatigue. Data is structured and unstructured as well. Log files and CCTV footage is unstructured data whereas data like the transaction data of the bank can be saved in tables is structured data.
- Volume: It refers to the amount of data that is generated at an exponential rate and the growth in data storage as the data no longer comprises only of text data. It is common for enterprises to have terabytes & petabytes of storage systems. And as the DB grows, the architecture and applications need to be evaluated on a regular basis. This big volume of data represents Big Data.

1.1.4 Hadoop

Hadoop is a programming framework written in java which allows data distributed processing across clusters of computers using simple programming models. It is hosted by Apache Software Foundation [6] and is used for batch/offline processing. To scale up in Hadoop all we need to do is add modes in the cluster.

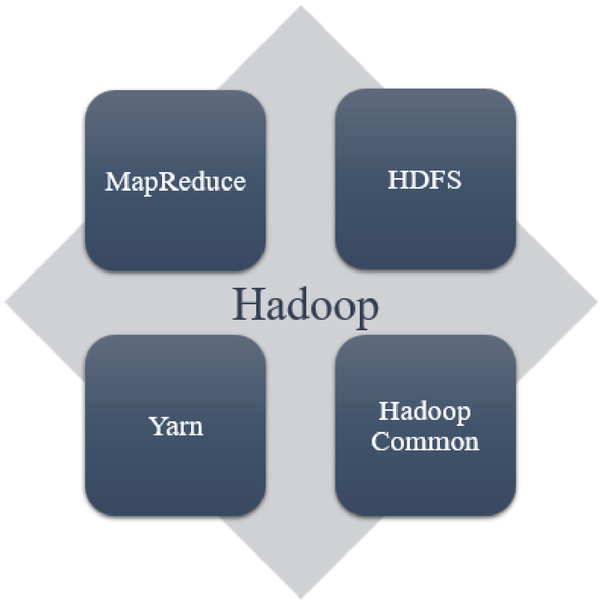


FIGURE 1.4: Modules of Hadoop

1.1.4.1 MapReduce

Hadoop provides the platform to work in clusters, and the clusters can store data and perform parallel computation as well. MR has a master-slave architecture where a single master node manages enormous slave nodes. In simple words, Hadoop can be termed as a combination of HDFS and MR. Mapper and Reducer and two platform independent and basic operations that MR framework provides us on data. In simpler words, MR is a programming model which is used to process large distributed datasets.

1.1.4.2 HDFS

Hadoop has another module, HDFS, which is a distributed file system. In HDFS, the data is distributed and replicated to stand by failure and fast availability to the parallel application. It uses commodity hardware which makes it cost effective and involves the concept of blocks, data node and name node.

- Blocks

  The minimum amount of data that HDFS can read or write is termed as a block. By default, these blocks are 128 MB and are configurable. The files are broken down into chunks which are block-sized, stored as independent units.

- Name node

  HDFS follows master-worker pattern to function, where the name node acts as the master. It is the controller and the manager of HDFS because of the fact that it knows status and the metadata of all the files in HDFS. Metadata information includes file permission, names and the location of each block. It is stored in the memory of name node itself to allow faster access to data.

- Data node

  Data nodes act as the worker nodes and they store and retrieve blocks when they are told to; by name done or client. They must report back to name node of the list of blocks which they are storing. Data node also works for block creation, deletion and replication because it is also a commodity hardware.

1.1.4.3 Yarn

Yarn is a core component to Hadoop and is added to provide improved performance. It stands for ‘Yet Another Resource Negotiator’. Yarn is the next generation of Hadoop’s computing platform which offers various advantages as compared to classic MR engine.

1.1.4.4 Hadoop Common

Hadoop common cites to the common utilities and libraries which supports other modules of Hadoop. It assumes that the hardware failures are common, and these should be handled automatically. Hadoop Common is the core of the framework as it provides us with crucial services and processes. It also contains all the vital JAR files and provides us with source code and documentation.

1.2 PROBLEM STATEMENT

The size of WWW is growing rapidly and at the same time, the number of queries, the search engines can handle has grown incredibly too. With increasing number of users on the web, the number of queries submitted to the search engines are also increasing exponentially. Therefore, the search engine must be able to process these queries efficiently. Thus, some web mining technique must be employed in order to extract only relevant documents from the database and provide intended information to the users.

Page Ranking is used to present the documents in a sequentially ordered manner, which can arrange the document taking various factors into account such as their relevance, importance, content score and uses web mining techniques to order them. Varying with the techniques we have some algorithms that rely on the link structure i.e. the popularity score of documents (WSM), whereas others search for the content (WCM), and some consider the combination of both. Following are the common page ranking algorithms that are in practice:

- Page Rank Algorithm
- Weighted Page Rank Algorithm
- Page content Algorithm
- HITS Algorithm

Taking topologies of the hyperlinks into account we get to know that WSM categorizes web pages and generates related patterns, such as similitude and the relationships between various websites. WCM works while considering the structure of the document (the internal-file degree) at the same time as WSM discovers the structure of the links among documents (the inter-report level). The number of links to a page i.e. inlinks and links from a page i.e. outlinks are significant in the concept of web mining and web mining techniques.

Hence, this is due to the facts that a well-liked web-page is often cited to by other pages and that a web-page which is important will contain a large number of outlinks. Such an idea lies behind a well-known search engine Google. The algorithm used by Google is PR Algorithm to rank its resulted pages.

1.3 OBJECTIVES

A system ranking web pages can have various applications. Some straight forward uses can be:

- Comparing two or more web pages
- Creating the outlinks and the inlinks
- Modified iterative PR Algorithm – To calculate the authoritative score for every particular node.
- Optimizing the data – as it can handle an enormous amount of data without any effect on the speed.

1.4 METHODOLOGY

Page Ranking is already a well-known algorithm, which is used by Google to rank websites in their output. It is the way of measuring the importance of a given webpage. Page ranking has been classified into two categories by the search engine. The first is the query dependent, depends upon the frequency and the positioning of query terms. The second one is query independent factors which take link popularity into account. Out of all the existing algorithms, Weighted Page Rank is the most appropriate to work with.

The relevancy of the search engine results is measured by Page level keyword that is an essential factor. Result set retrieved by search engines are containing a huge number of useless web pages. They have shown that by retrieving the result which is relevant for a particular query increases the precision. And based on the page level keywords, search engines are evaluated on informational queries.

We have opted Java as our software development programming language because it is scalable, platform independent and easy to use and access. Certain models already exist in Java also but again with many issues as discussed earlier in the report.

The project has been implemented using Big Data in Hadoop. Hadoop is a remarkable open-source platform and provides with the unique data management provisions

NetBeans:

Netbeans is helpful in providing us the following:

- Profiler – for our performance measurement.
- Debugger – for optimizing and removing some logical errors.
- Ant build tool – for making an executable JAR file.
- Javadoc generator – for building the map of packages, classes and functions used in the project.

Hadoop:

- Scalable
- Cost Effective
- Flexible
- Fast
- Resilient to Failure

Ubuntu:

- High customization
- Secure
- Low System requirements
- Open Source
- Comprehensive software update
- Integrated app store

1.5 ORGANIZATION

Hadoop is a programming framework written in java which allows data distributed processing across clusters of computers using simple programming models. It is hosted by Apache Software Foundation[Bigdata&hadoop:aSurvey] and is used for batch/offline processing. To scale up in Hadoop all we need to do is add modes in the cluster. Moreover, it incorporates like features to google file.

Hadoop can run in three different modes namely, standalone mode, pseudo-distributed mode and fully-distributed mode.

1.5.1 Standalone Mode

Hadoop has a default mode – standalone mode, which uses local file system for operations of input and output. This mode is much faster than the rest and the main purpose of this mode is to debug. HDFS is not supported. Further there is no custom configuration for files such as mapred-site.xml, core-site.xml and hdfs-site.xml.

1.5.2 Pseudo-Distributed Mode

There is no distinction between the master node and the slave node as it operates on a single node. Hence, commonly known as Single Node Cluster. The files demand custom configuration. In this case, all daemons run on one node, explaining that the master and the slave node are the same.

1.5.3 Fully-Distributed Mode

This is the mode which defines Hadoop and makes it unique. Fully-Distributed Mode is the phase of production for Hadoop as the data is used and distributed among several nodes on the cluster. Unlike Pseudo-Distributed Mode, here separate nodes are allocated as Master and Slave nodes.

CHAPTER 2: LITERAURE SURVEY & METHODOLOGY

2.1

TITLE: An Efficient Page Ranking Approach Based on Hybrid Model

Web mining is mining of data present in the web database. Web mining consists of Web Content Mining(WCM), Web Structure Mining(WSM), Web Usage Mining(WUM) [1]. Web content mining aims to mine useful information or knowledge from web page contents. Web structure mining tries to discover useful knowledge from the structure of hyperlinks. Web usage mining refers to the discovery of user access patterns from Web usage logs.

Lisa Rodrigues [1] proposed a Hybrid model combining the idea of link based mining and page level keyword search. First of all a database is created and a link graph is created to explain the linking of pages after creation of database. After that the work is divided into two modules:

1. Keyword matching module [1]: This aims at determining the pages based on number of times input keyword occurs in the particular page, matches the input keyword in dictionary and increases the counter by one.

2. Analysis module [1]: this keeps the database of pages in the form of table, finds the total and average count of keywords.

This paper has given a whole new approach to work on ranking schemes. The hybrid model takes into account all three mining techniques that makes it more useful. This is how it will provide more better and efficient results.

In internet these days it is very important to take into account the users interest hybrid model fulfills this demand and enhances the surfing.

Here is the flow chart of how the hybrid model works:

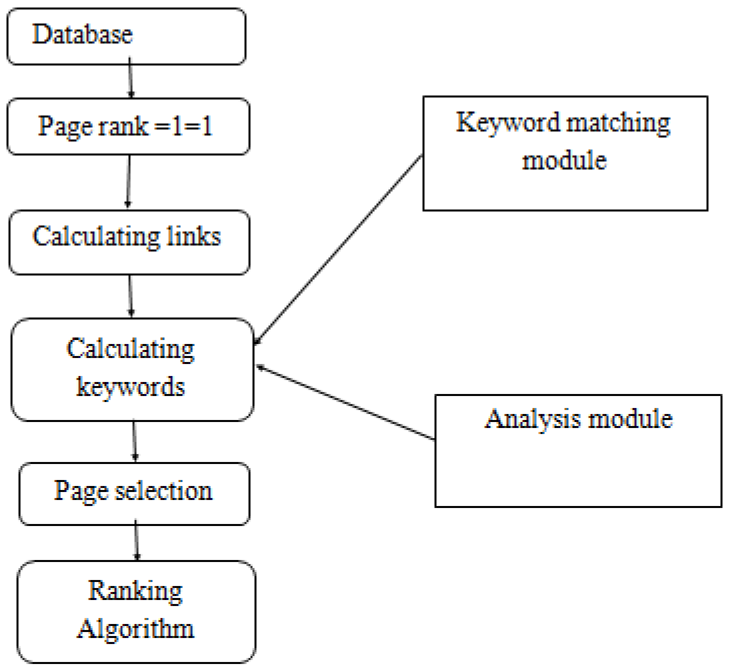


FIGURE 2.1: Working of Hybrid Model

2.2

TITLE: User preference based ranking algorithm

Internet consists of countless number of web pages. It's not easy to retrieve useful information. To overcome this problem search engines were developed for finding relevant information. These hyperlinks are analyzed to discover concealed human interpretations that can be extremely useful for inferring user behavior [2]. Authors have explained how the search engines work for better understanding.

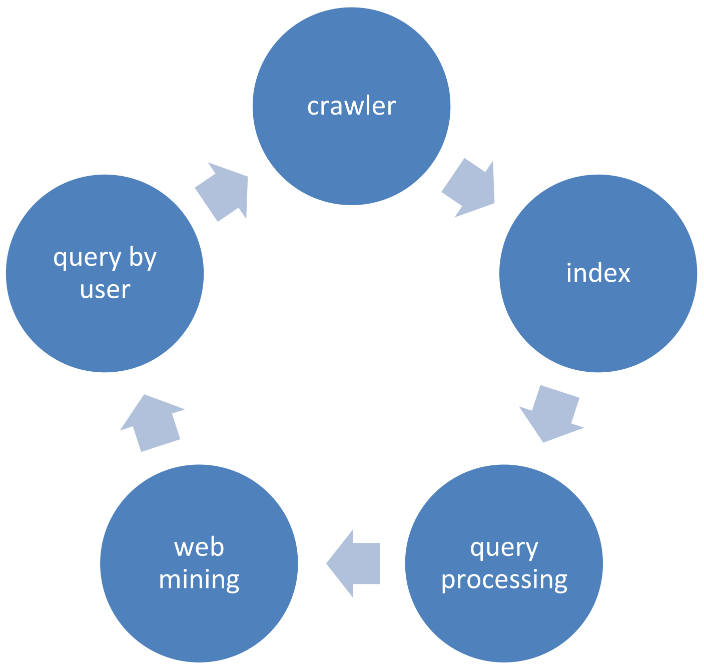


FIGURE 2.2: Working of a Search-Engine

After observing various ranking algorithm, for example HITS, Time Rank, Page Rank and Weighted Page Rank Dr. Daya Gupta [2] and Devika Singh [2] proposed User Preference Based Ranking Algorithm, which is an extension of Weighted Page Rank Algorithm. Algorithm take into account both Web Content Mining [1] and user preference [2]. It also takes into account the use of Software Agents and Web Crawlers.

Using the agents and crawlers they determined the page relevancy and incorporated user behavior based on the visits to any inlinks of a particular page. Algorithm first retrieves the web pages from crawler and creates the graph of those web pages.

$$ER(u) = (1-d) + d\sum_{v\in B(u)}(Vu * W^{in}_{(v,u)}W(v,u)in + W(v,u)out) * (Cw + Pw)/Tv \qquad [2]$$

Using this formula ranking of web pages is calculated keeping into account the content weight(weight of content with respect to the web page)[2] and probability weight(probability of query in terms of web page)[2].

2.3

TITLE: An Efficient Algorithm for Ranking Research Papers Based on Citation Network

Searching is keen part of our life these days. Similarly authors of this paper though out of the box and developed an algorithm that ranks the research paper based on their importance. They have modified the Page Rank algorithm and designed one that ranks the papers and give them an authoritative score. They also introduced time factor that helps in ranking. Researchers are more interested in conferences and some are in authors so keeping that in mind they have ranked even the conferences and authors for better results.

A.P.Singh, K.Shubhankar, V.pudi[3] propose an algorithm to rank the research papers based on their importance and relevance published in various fields over the years. Every research paper is linked with the other research paper, so accordingly a graph exists.

This algorithm keeps up the score of author, conference and research paper. This approach is entirely based on citation graph formed by papers [3]

According to their approach they first used the ranking algorithm to find the authorative score [3], by first creating the data structure of the papers cited. After that the iterative Page Rank algorithm is used[3]. Following algorithm starts by ranking every paper as 1, and then iteratively increases the rank.

Here is the formula used for iterations:

$$NewRank = 0.15 + 0.85 * \sum(PR[i\}OC[i] \qquad [3]$$

Now to calculate the Rank of conference they simply find the paper rank and multiply it to number of paper published in that conference. Similarly, to find score of conferences year wise paper rank is multiplied to paper presented in that given year.

Similarly, Author Rank is calculated, we calculate paper rank, conference rank and number of papers presented by the author. After multiplying them we get the final answer i.e. author score.

2.4

TITLE: Page Ranking Algorithm A Survey

Neelam Duhan, A.K. Sharma, Komal Kumar Bhatia [4] have explained the crawlers and structure of search engine and they are an important part of search engines. They have described the extraction of data, web mining and its techniques. Web mining have three techniques which are:

1. Web content mining
2. Web usage mining
3. Web structure mining

These days most search engines are tracking he users searches to make it more user friendly.

They have explained basic of search engine, how it works

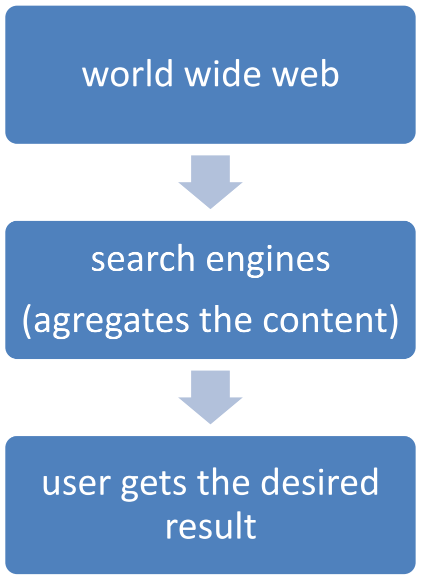


FIGURE 2.4.1: Idea of Search-Engine

Here is the comparison of their research on every page ranking algorithm:

Algorithms	**Description**	**Input Parameters**	**Complexity**	**Results**
Page Rank	Rank is calculated at indexing time.	Outlinks	O(log n)	Medium
Weighted Page Rank	This also calculates rank at indexing time, but pages are sorted according to importance	Outlinks and inlinks	<O(log n)	Higher than Page Rank
Page Content Rank	Calculate new rank every time of first n pages and relevant pages are shown.	Contents	O(m*)	Equal to WPR
HITS	Calculates hub and authority score	Outlinks, Inlinks, Content	<O(log n)	Less than PR

TABLE 2.4.1: Comparison of Ranking Algorithms

2.5

TITLE: A Novel Based Probability Algorithm for Page Ranking in Semantic Web Search Engine

In this paper authors proposes a new way of ranking the pages. He suggests that bases on user preference and scores to web page each web document will be given a new priority. This is basically an extension to Page Rank Algorithm

Basic idea of their project:

Search engine Components: a controlled web environment is created in OWL language [5] and some modifications are done afterwards to improve the quality of searching. Travel database is pasted into the environment created, same way ranking algorithm will be pasted. The GUI created will help to know the user query and provide us with the results.

Relation based ranking; introduction : all the information below is from the reference [5]. Let us assume that the user specifies the keyword 'homestay' and the keyword 'Italy' in the query interface. The associated concept with the keyword will be 'destination' for 'Italy' and 'accommodation' for 'Homestay'. But the exact relation resides only in the mind of user, we can just assume that one of the relation might exist between two. This is how pages are given priorities based on relation the software will analyze a relation and provide the optimum priority.

`

So, based on the relation between the concepts, a probability score will be created that can be easily calculated with the help of graph. It will check if the result is valid and important for user and hence provide user with the output.
Annotation Graph for Italy, Church and homestay.

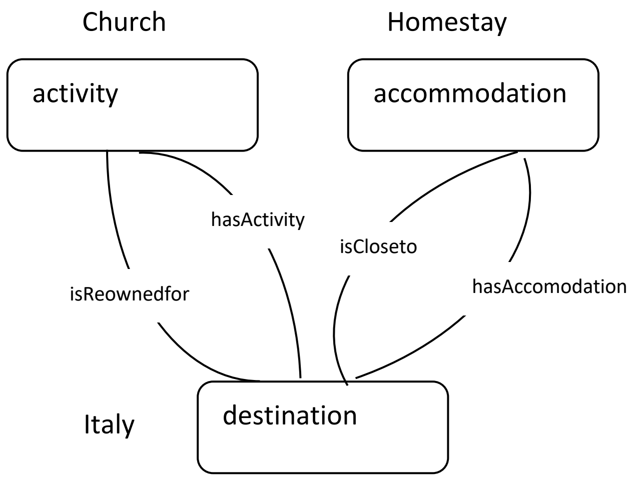


FIGURE 2.5.1: Annotation Graph for Italy, Church and homestay

The algorithm provided by them was pretty much accurate as compared to the google. They have given the comparison results with google. Ranking score was improved as checked with the Italy and homestay example. Ranking score was better and improved with the following algorithm.

This algorithm will provide better results without much interference of human. This algorithm is different than most of the semantic search engines algorithm and better than most of them, hence provide us with better and authentic results.

2.6

TITLE: Improving algorithm for calculation of page rank

World wide web is the most important part of our lives these days. So, it is very important that we find all the relevant information in it. As we know web pages are not fully structured, so we need a technique that can sort such type of data.

In this paper the authors have described about the web mining techniques and their importance in ranking the web pages for user to get the desired result. Authors have explained how web mining works. It is as follows:

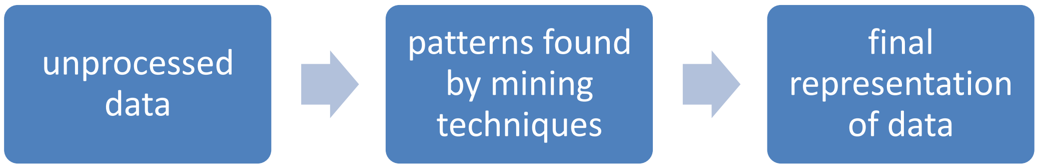

unprocessed data
patterns found by mining techniques
final representation of data

FIGURE 2.6.1: Working of Web Mining Techniques

Authors have explained the three techniques of web mining:

1. Web Content mining (processes data according to the content)
2. Web structure mining (generates the structural abstract of the data)
3. Web usage mining (keeps track of the pattern user searches)

Author have proposed the algorithm based on the research of three main ranking algorithm namely Weighted page rank, HITS and Page rank algorithm. They have explained working of all these algorithms, their formulas and derived the formula that will improve the Page Rank algorithm

In order to improve the page rank algorithm, authors have deeply studied the disadvantages of the page rank algorithm and improved the following disadvantages:

1. Page Rank algorithm takes into account only the inlinks, authors counted the inlinks as well as outlinks of the web pages

2. Page rank uses only two web mining techniques i.e. web structure mining to create structured data and web content mining to find the keywords in the web pages.
   In order to provide with better results, they have also included web usage mining to keep track of users visits.

The final equation of proposed algorithm is:

$$pageRank(u) = (1-d) + d \sum_{v \in R(u)} \frac{\left(V_x * x * W^{in}_{(v,u)} + y * W^{out}_{(v,u)}\right) pageRank(v)}{TL(v)}$$

[6]

After the experimental analysis they have derived that the proposed algorithm works more efficiently than the previous existing one.

2.7

TITLE: Implementation in Hadoop

This paper is divided into 4 parts i.e.

Introduction; where they have explained about World Wide Web and need for ranking and how map reduce works in Hadoop. System model and annotations: this part describes about the page rank algorithm, how it works and basics.

Implementation: this part consists of three algorithms i.e. first to parse web pages, second to implement mapper class and third to implement reducer class. After that is shows how to implement this algorithm in Hadoop with map reduce. Numerical results: in this part they have shown the final results of algorithm where they have shown the execution time of mapper and reducer

The aim of this paper is to implement page rank algorithm over multiple machine in Hadoop. Authors have chosen Hadoop Map reduce due to its various advantages like:

1. It can process large data.
2. It can handle both structured and non-structured data.
3. It is failure tolerant
4. No need to learn new languages you can easily code in java.

They have divided the whole algorithm into three parts i.e. first part checks the raw web pages, second part implements raw web pages and the third one to implement reducer class.

They have implemented java code over a distributed system and got the reasonable results to conclude the paper successfully.

CHAPTER 3: SYSTEM DESIGN

There exist a number of algorithms of the similar concept, i.e. they rank webpages based on their importance and significance. Some of the well-known algorithms are

- Text-Based Ranking Algorithm
  - → It is a traditional way which ranks the pages based on their textual content and number of terms that match with the query string.

- Page Rank Algorithm
  - → This algorithm takes link structure into account to determine the importance of web pages.

- Weighted Page Rank Algorithm
  - → It is an extension of the Page-Rank algorithm and allocates a higher value of rank to the more significant pages. With this amount of benefits, it is accompanied by many limitations such as relevancy is ignored, and the method computes score at a single level.

- Page Content Rank Algorithm
  - → This algorithm uses WCM technique. The term specified in the given query determines the page importance.

- HITS Algorithm
  - → It is a hybrid model which is search query dependent.

We have opted Java as our software development programming language because of the sole reason that it is scalable, platform independent and easy to use and access.

Page Rank Algorithm exist with the following steps:

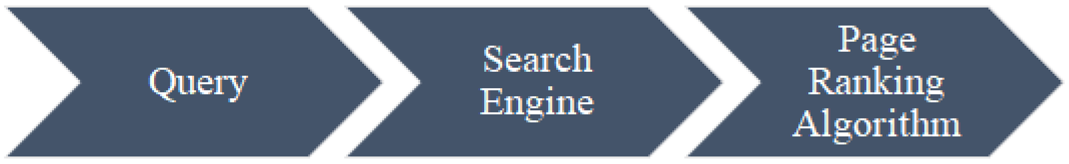


FIGURE 3.1: Existing Model

To make it more scalable and to try to make it handle more heavy datasets, we have implemented it using Big Data. Some other factors such as cost efficiency and how fast Hadoop handles data are also responsive for the implementation of Page Rank Algorithm in Hadoop.

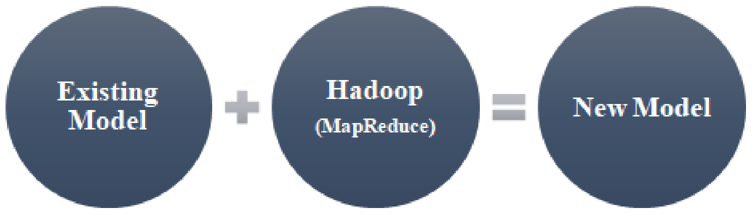


FIGURE 3.2: Proposed Model

Our software is made in two major steps that too in four phases. The first step being the development of the java code in NetBeans. The java code contains 1 main configuration class, 3 mappers and 2 reducer classes. The reason behind a lesser number of reducer class is that the desired output is achieved after the third mapper class. Hence, no need to implement another reducer code. Build the project and create a jar file.

Before moving towards the second step we need to install ubuntu with 1 processor and 2.5 GB memory. Now, the second major step was the installation of Hadoop in a virtual environment i.e. ubuntu. VM Ware was installed as we are using pseudo-distributed mode to let Hadoop know that we are executing the program over different systems.

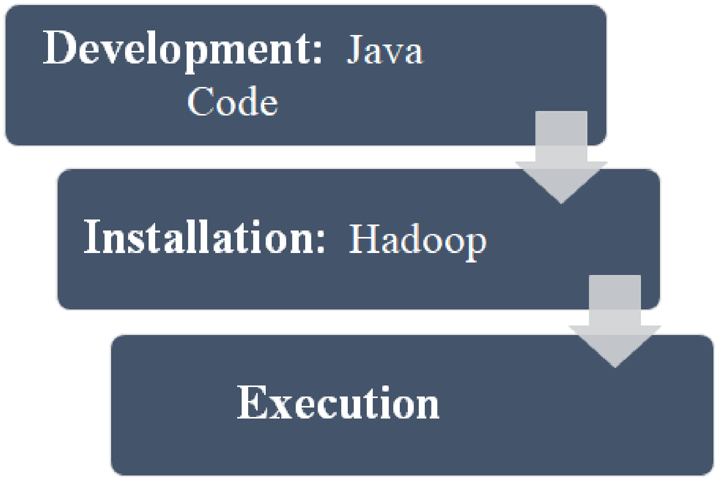


FIGURE 3.3: Project Phases

Installation of Hadoop has three phases namely setup, server login, file editing and executing.

- Hadoop Setup
  1. Unzip and create a folder of Hadoop:

     tar zxvf jdk-7u80-linux-x64.tar.gz
  2. Setup JDK and JRE:

     sudo apt-get update

     sudo apt-get install openjdk-8-jre
  3. Open Server and Client:

     sudo apt-get install openssh-server openssh-client
  4. Using res algorithm to generate key pairs:

     sudo apt-get install openssh-server openssh-client
  5. Generate public and private key for authentication:

     ssh-keygen -t rsa -P ""
  6. Authorization of key for unrestricted access:

     cat $HOME/.ssh/id_rsa.pub>>$HOME/.ssh/authorized_keys

- Server Login
    1. To login into the server:

       ssh localhost

- File Editing
    1. Open up the file named Hadoop_env.sh to set the java path:

       sudo gedit hadoop_env.sh
    2. For internal working and generation of files we need to edit core-site:

       sudo gedit core-site.xml
    3. To set replication and to make directories; set replication factor to 1; make namenade and datanode directories:

       sudo gedit hdfs-site.xml
    4. Similarly edit mapred-site.sh and yarn-site.sh
    5. After editing file, we need to edit bash. As

       sudo nano ~/.bashrc
    6. After creating and editing all the files, we have to format the namenode. This is done in order to format all of the space and let Hadoop create its own clusters:

       bin/hadoop namenode -format

- Executing
    1. Select and download the particular dataset on which you need to perform page ranking algorithm.
    2. Before executing, start all the files that have been edited by using the command:

       sbin/start-all.sh
    3. To check if the nodes are ready:

       jps
    4. To execute the code:

       bin/hadoop jar PageRank.jar --input /project/web-Google.txt --ouput /project-out/
    5. Open Mozilla Firefox
    6. Open utilities in localhost/50070

The final phase is the fixing the errors, if any.

For our software program improvement, we've used Iterative model of SDLC (software improvement life Cycle). This software engineering life cycle starts with a quite simple implementation of the small set of the software program requirements and next it iteratively complements the evolving variations until the whole and integrated device is implemented and geared up to be deployed.

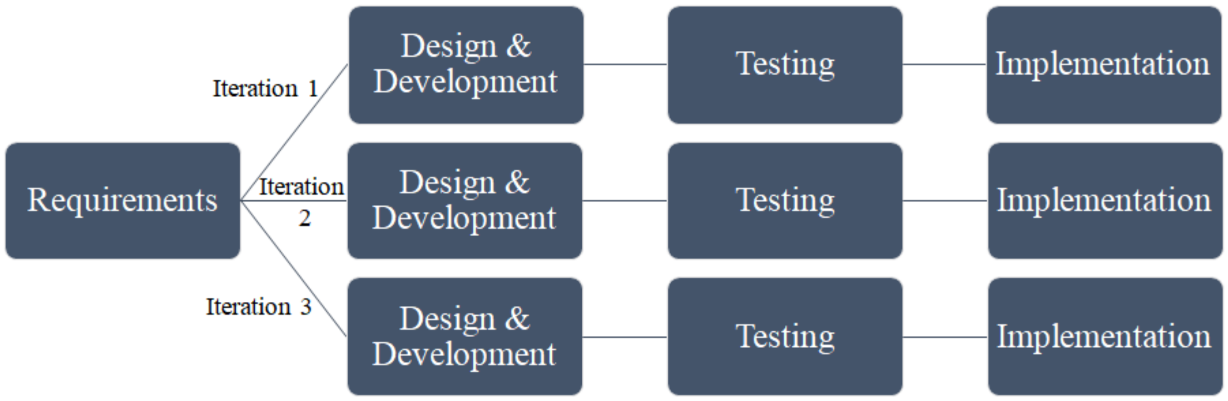
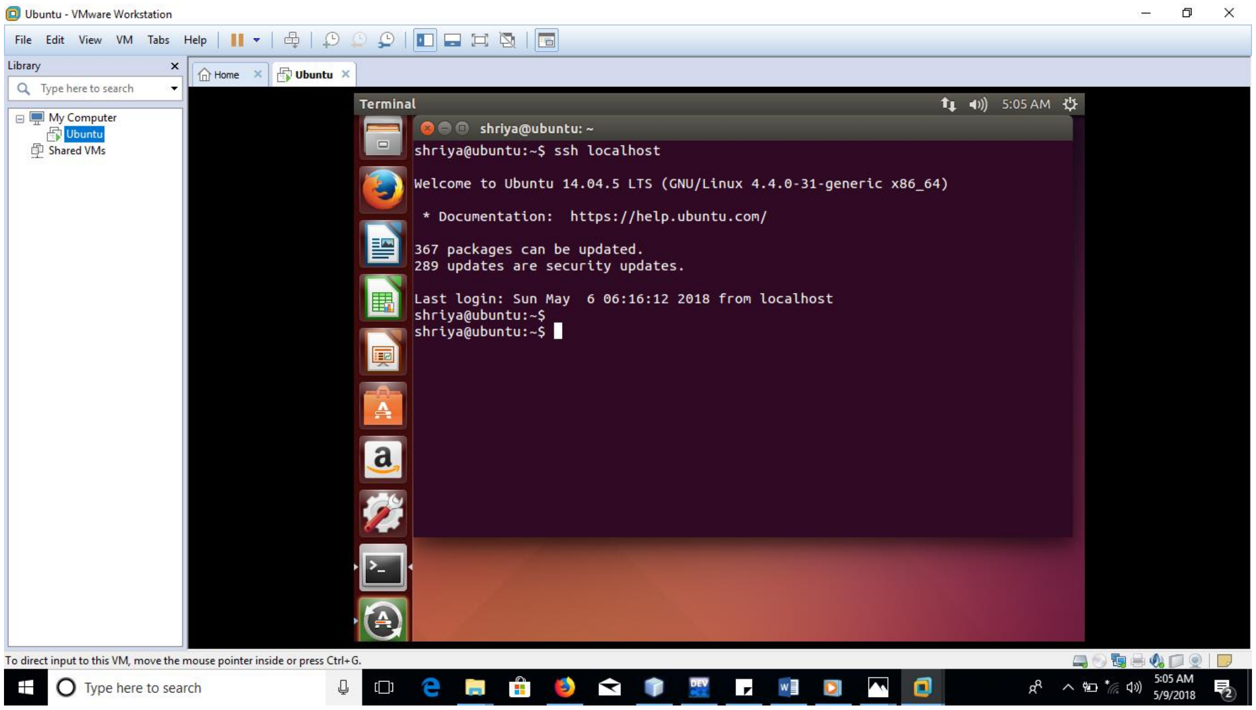


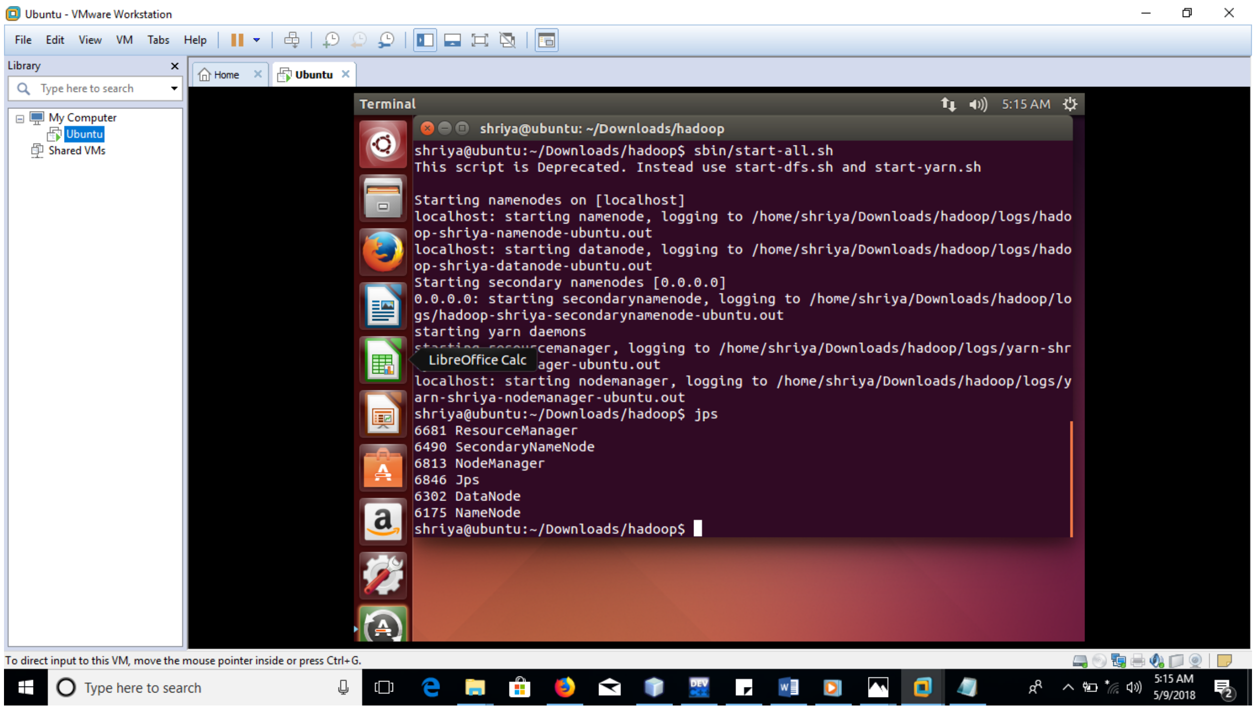
FIGURE 3.4: Iterative SDLC Model

OUTPUTS



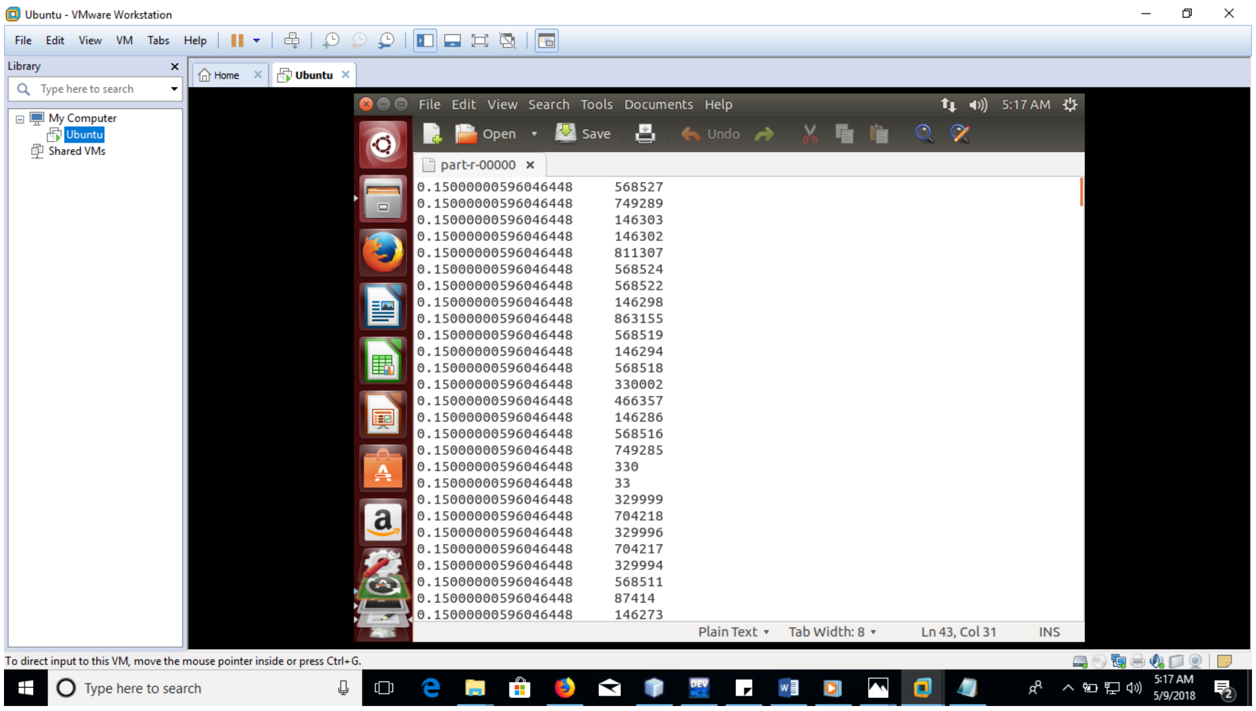
```
shriya@ubuntu:~$ ssh localhost
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 4.4.0-31-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

367 packages can be updated.
289 updates are security updates.

Last login: Sun May  6 06:16:12 2018 from localhost
shriya@ubuntu:~$
shriya@ubuntu:~$
```

FIGURE 3.5: Server Login



```
shriya@ubuntu:~/Downloads/hadoop$ sbin/start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh

Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/shriya/Downloads/hadoop/logs/hado
op-shriya-namenode-ubuntu.out
localhost: starting datanode, logging to /home/shriya/Downloads/hadoop/logs/hado
op-shriya-datanode-ubuntu.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/shriya/Downloads/hadoop/lo
gs/hadoop-shriya-secondarynamenode-ubuntu.out
starting yarn daemons
starting resourcemanager, logging to /home/shriya/Downloads/hadoop/logs/yarn-shr
iya-resourcemanager-ubuntu.out
localhost: starting nodemanager, logging to /home/shriya/Downloads/hadoop/logs/y
arn-shriya-nodemanager-ubuntu.out
shriya@ubuntu:~/Downloads/hadoop$ jps
6681 ResourceManager
6490 SecondaryNameNode
6813 NodeManager
6846 Jps
6302 DataNode
6175 NameNode
shriya@ubuntu:~/Downloads/hadoop$
```

FIGURE 3.6: Node formation


Ubuntu - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Shared VMs
Home
Ubuntu
File Edit View Search Tools Documents Help
5:17 AM
Open
Save
Undo
part-r-00000
Plain Text
Tab Width: 8
Ln 43, Col 31
INS
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
Type here to search
5:17 AM
5/9/2018

```
0.15000000596046448	568527
0.15000000596046448	749289
0.15000000596046448	146303
0.15000000596046448	146302
0.15000000596046448	811307
0.15000000596046448	568524
0.15000000596046448	568522
0.15000000596046448	146298
0.15000000596046448	863155
0.15000000596046448	568519
0.15000000596046448	146294
0.15000000596046448	568518
0.15000000596046448	330002
0.15000000596046448	466357
0.15000000596046448	146286
0.15000000596046448	568516
0.15000000596046448	749285
0.15000000596046448	330
0.15000000596046448	33
0.15000000596046448	329999
0.15000000596046448	704218
0.15000000596046448	329996
0.15000000596046448	704217
0.15000000596046448	329994
0.15000000596046448	568511
0.15000000596046448	87414
0.15000000596046448	146273
```

FIGURE 3.7: Output- PageRankings

CHAPTER 4- PERFORMANCE ANALYSIS

Performance of algorithm is measured by its execution time. Surgey Brin and Larry Page[2] have developed the Page Rank algorithm. The original algorithm has some disadvantages such as:

1. it uses only two techniques of web mining i.e. web content mining and web structure mining.
2. It calculates only the inlinks of any web page

Due to these disadvantages page rank algorithm was underestimated.

We covered up one disadvantage i.e. we used outlinks and inlinks to calculate ranks of the nodes(webpages)

Also, we used dampening factor as 0.5 instead of 0.85, so it provides us with better results.

Here is the comparison of the nodes with d=0.85 and 0.5

pages	D=0.85	D=0.5
Page A	0.65	1.08
Page B	0.93	1.26
Page C	0.60	0.66

TABLE 4.1: Comparison of dampening factor

Reducing the dampening factor as you can see increases the rank to more accurate position and is easy to calculate while iterations.

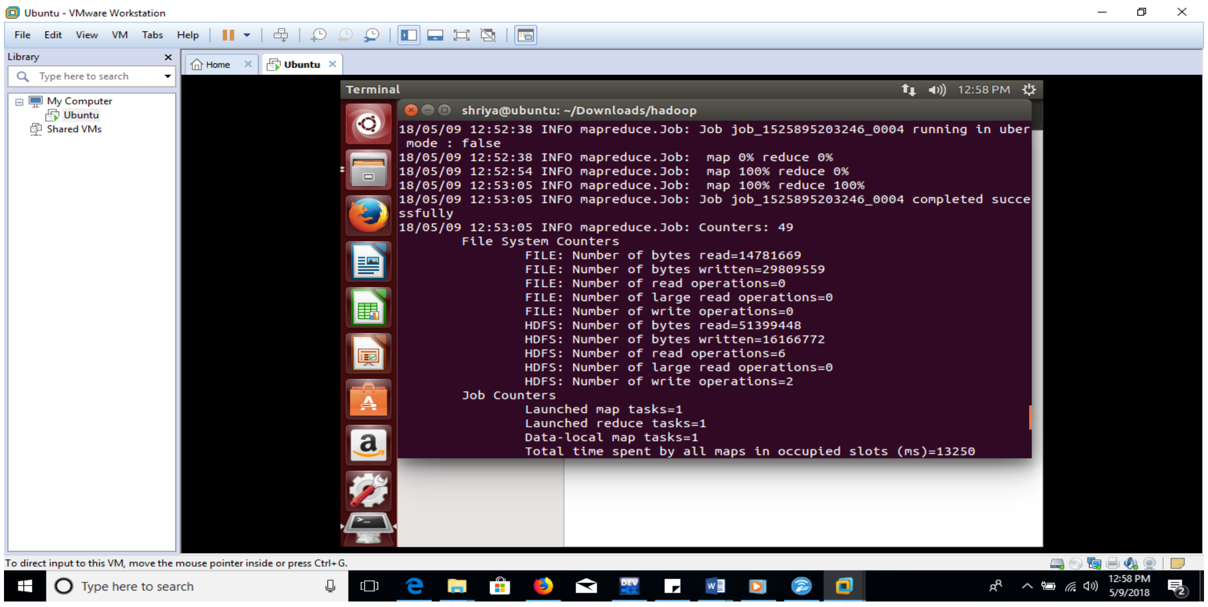
Analysis of output:

We have considered three datasets to understand the results in better way, which are:

1. Google
2. Stanford
3. Berksten

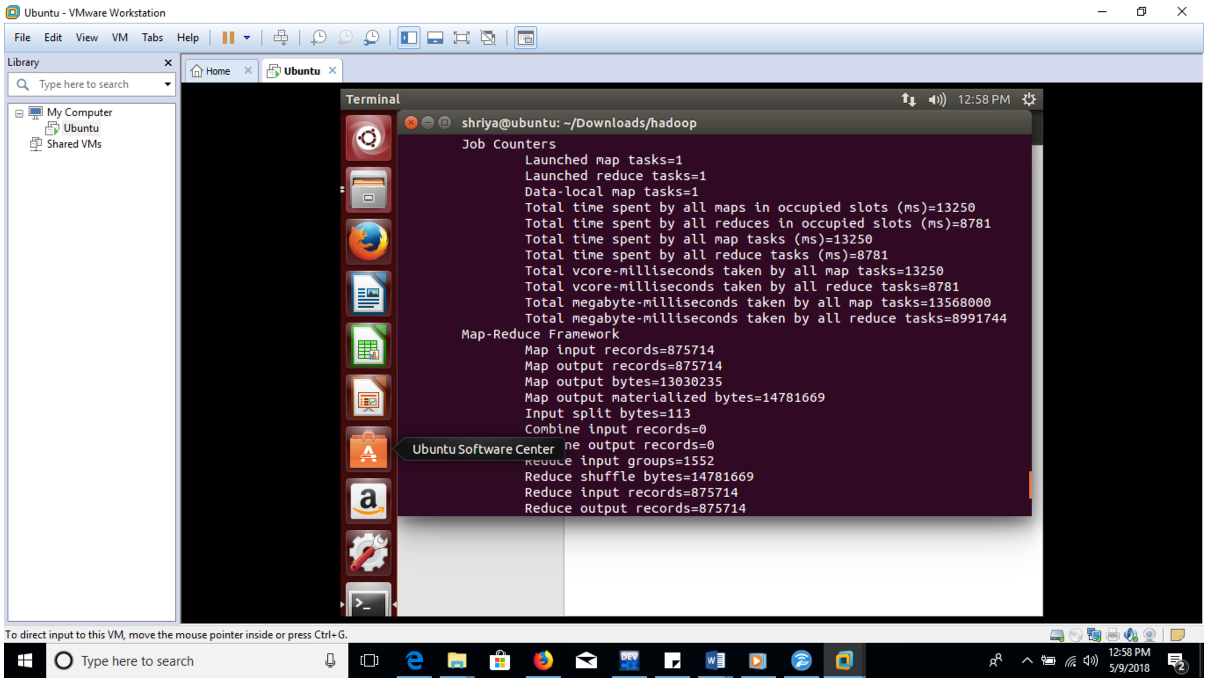
We recorded their output and found the comparison between three and came to the conclusion that google data set gave us the best results

4.1 Analysis of Google Data Set



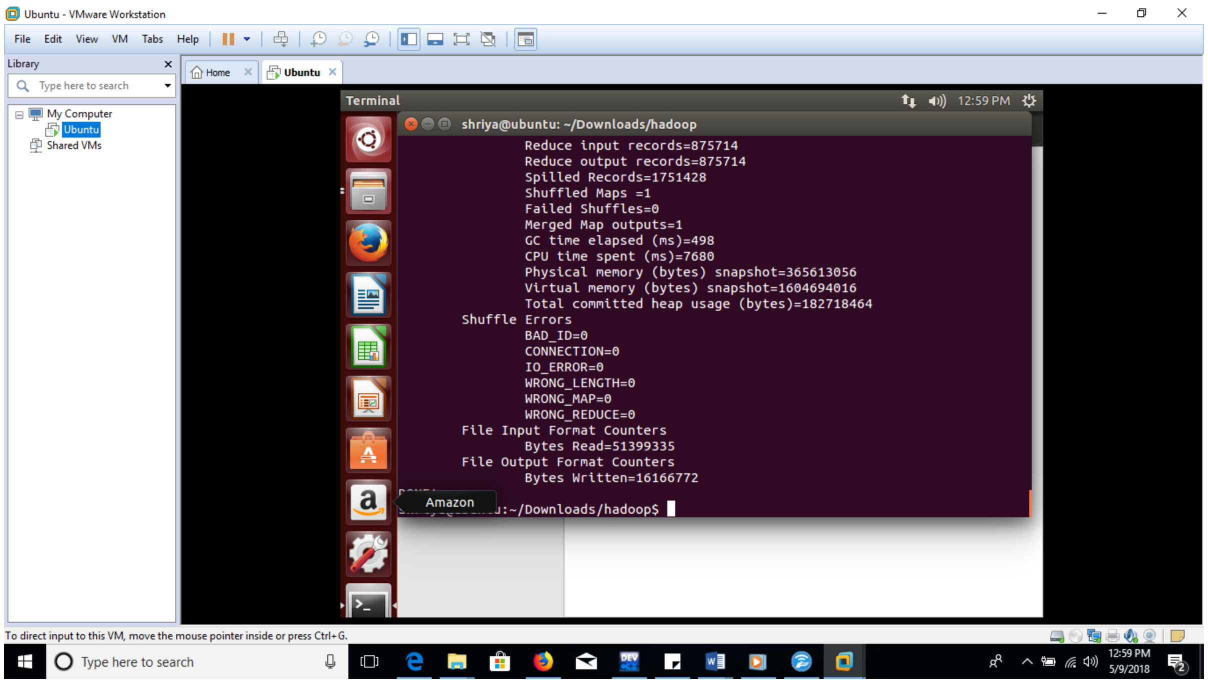
```
shriya@ubuntu: ~/Downloads/hadoop
18/05/09 12:52:38 INFO mapreduce.Job: Job job_1525895203246_0004 running in uber
 mode : false
18/05/09 12:52:38 INFO mapreduce.Job:  map 0% reduce 0%
18/05/09 12:52:54 INFO mapreduce.Job:  map 100% reduce 0%
18/05/09 12:53:05 INFO mapreduce.Job:  map 100% reduce 100%
18/05/09 12:53:05 INFO mapreduce.Job: Job job_1525895203246_0004 completed succe
ssfully
18/05/09 12:53:05 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=14781669
                FILE: Number of bytes written=29809559
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=51399448
                HDFS: Number of bytes written=16166772
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=13250
```

FIGURE 4.1: Output- Google Dataset(I)



```
shriya@ubuntu: ~/Downloads/hadoop
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=13250
                Total time spent by all reduces in occupied slots (ms)=8781
                Total time spent by all map tasks (ms)=13250
                Total time spent by all reduce tasks (ms)=8781
                Total vcore-milliseconds taken by all map tasks=13250
                Total vcore-milliseconds taken by all reduce tasks=8781
                Total megabyte-milliseconds taken by all map tasks=13568000
                Total megabyte-milliseconds taken by all reduce tasks=8991744
        Map-Reduce Framework
                Map input records=875714
                Map output records=875714
                Map output bytes=13030235
                Map output materialized bytes=14781669
                Input split bytes=113
                Combine input records=0
                      ne output records=0
                Reduce input groups=1552
                Reduce shuffle bytes=14781669
                Reduce input records=875714
                Reduce output records=875714
```

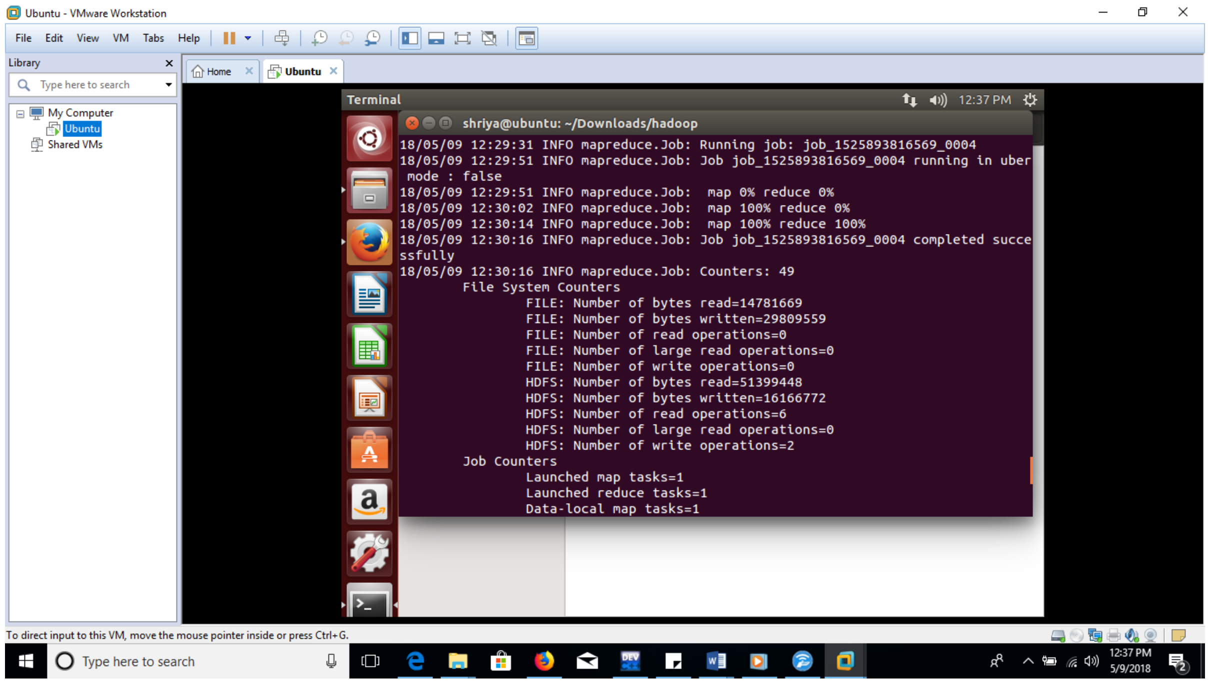
FIGURE 4.2: Output- Google Dataset(II)


Ubuntu - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Shared VMs
Home
Ubuntu
Terminal
12:59 PM
shriya@ubuntu: ~/Downloads/hadoop
Amazon
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
Type here to search
12:59 PM
5/9/2018

```
                Reduce input records=875714
                Reduce output records=875714
                Spilled Records=1751428
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=498
                CPU time spent (ms)=7680
                Physical memory (bytes) snapshot=365613056
                Virtual memory (bytes) snapshot=1604694016
                Total committed heap usage (bytes)=182718464
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=51399335
        File Output Format Counters
                Bytes Written=16166772
DONE!
shriya@ubuntu:~/Downloads/hadoop$
```

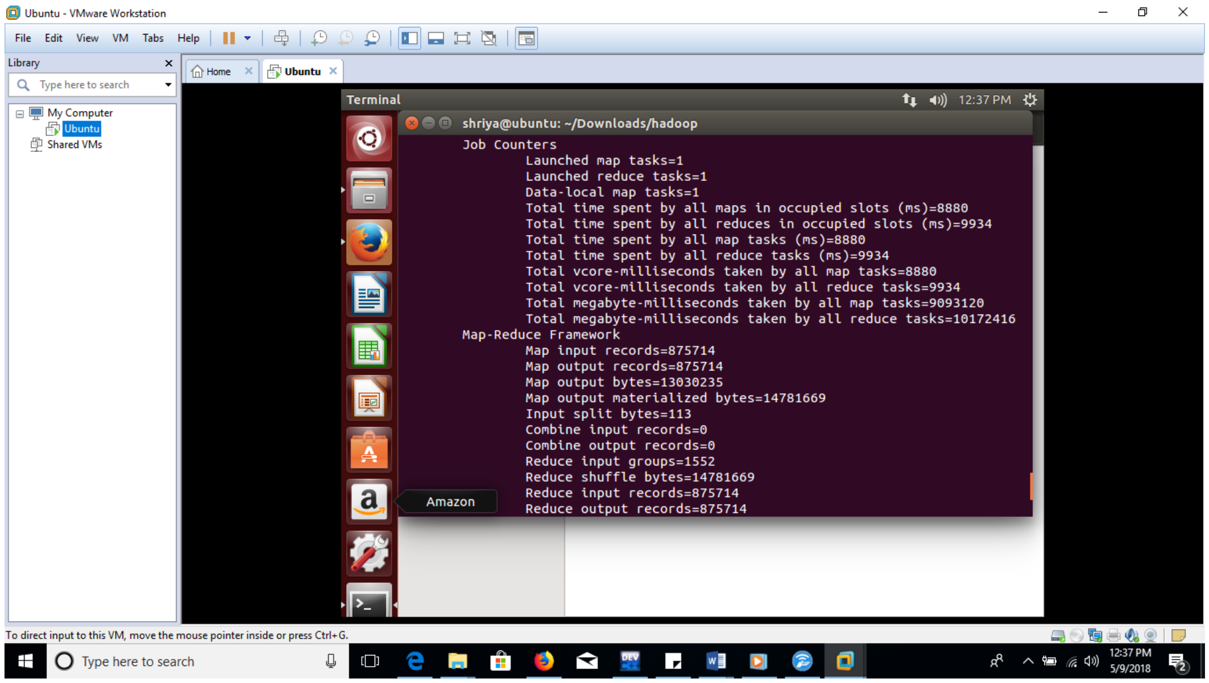
FIGURE 4.3: Output- Google Dataset(III)

4.2 Analysis of Stanford Dataset



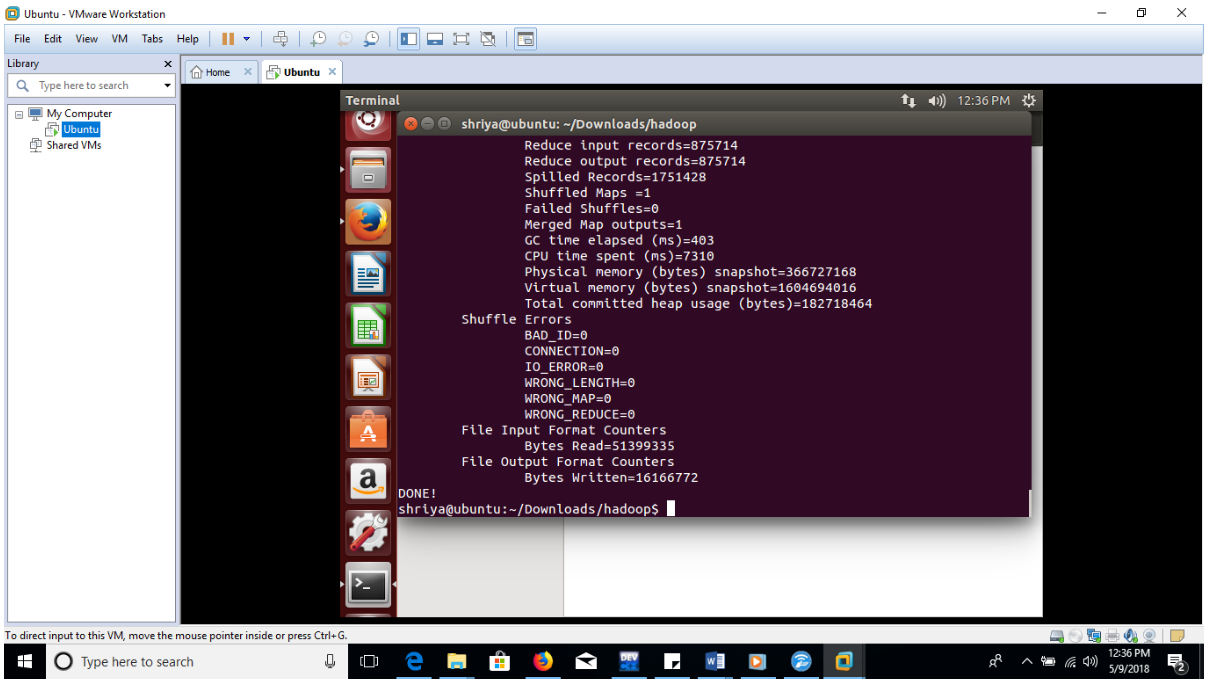
```
shriya@ubuntu: ~/Downloads/hadoop
18/05/09 12:29:31 INFO mapreduce.Job: Running job: job_1525893816569_0004
18/05/09 12:29:51 INFO mapreduce.Job: Job job_1525893816569_0004 running in uber
 mode : false
18/05/09 12:29:51 INFO mapreduce.Job:  map 0% reduce 0%
18/05/09 12:30:02 INFO mapreduce.Job:  map 100% reduce 0%
18/05/09 12:30:14 INFO mapreduce.Job:  map 100% reduce 100%
18/05/09 12:30:16 INFO mapreduce.Job: Job job_1525893816569_0004 completed succe
ssfully
18/05/09 12:30:16 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=14781669
                FILE: Number of bytes written=29809559
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=51399448
                HDFS: Number of bytes written=16166772
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
```

FIGURE 4.4: Output- Stanford Dataset (I)



```
shriya@ubuntu: ~/Downloads/hadoop
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=8880
                Total time spent by all reduces in occupied slots (ms)=9934
                Total time spent by all map tasks (ms)=8880
                Total time spent by all reduce tasks (ms)=9934
                Total vcore-milliseconds taken by all map tasks=8880
                Total vcore-milliseconds taken by all reduce tasks=9934
                Total megabyte-milliseconds taken by all map tasks=9093120
                Total megabyte-milliseconds taken by all reduce tasks=10172416
        Map-Reduce Framework
                Map input records=875714
                Map output records=875714
                Map output bytes=13030235
                Map output materialized bytes=14781669
                Input split bytes=113
                Combine input records=0
                Combine output records=0
                Reduce input groups=1552
                Reduce shuffle bytes=14781669
                Reduce input records=875714
                Reduce output records=875714
```

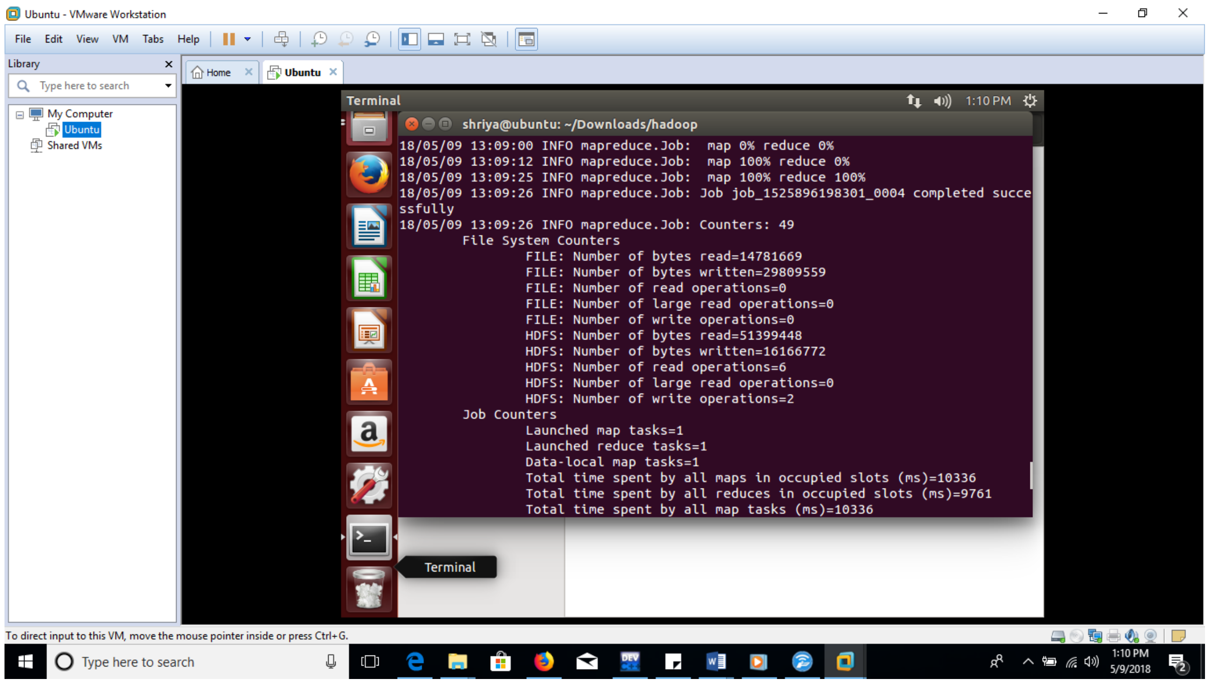
FIGURE 4.5: Output- Stanford Dataset (II)



```
                Reduce input records=875714
                Reduce output records=875714
                Spilled Records=1751428
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=403
                CPU time spent (ms)=7310
                Physical memory (bytes) snapshot=366727168
                Virtual memory (bytes) snapshot=1604694016
                Total committed heap usage (bytes)=182718464
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=51399335
        File Output Format Counters
                Bytes Written=16166772
DONE!
shriya@ubuntu:~/Downloads/hadoop$
```

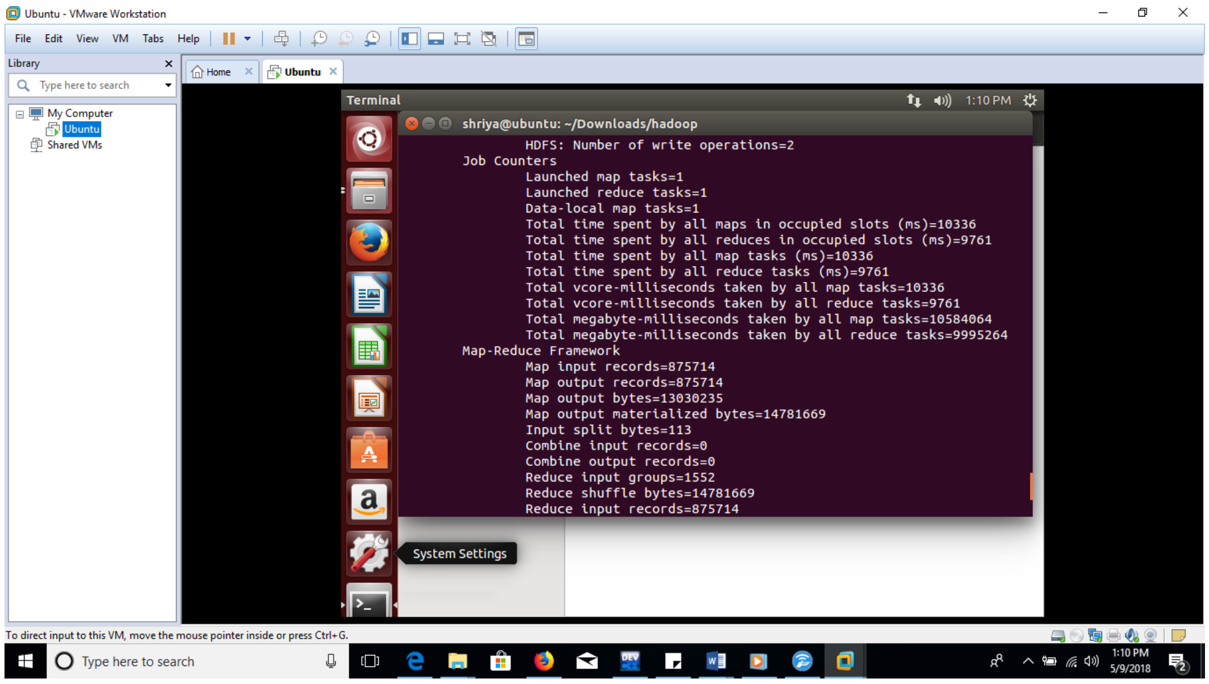
FIGURE 4.6: Output- Stanford Dataset (III)

4.3 Analysis of Berksten Dataset


Ubuntu - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Shared VMs
Home
Ubuntu
Terminal
1:10 PM
shriya@ubuntu: ~/Downloads/hadoop
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
Type here to search
1:10 PM
5/9/2018

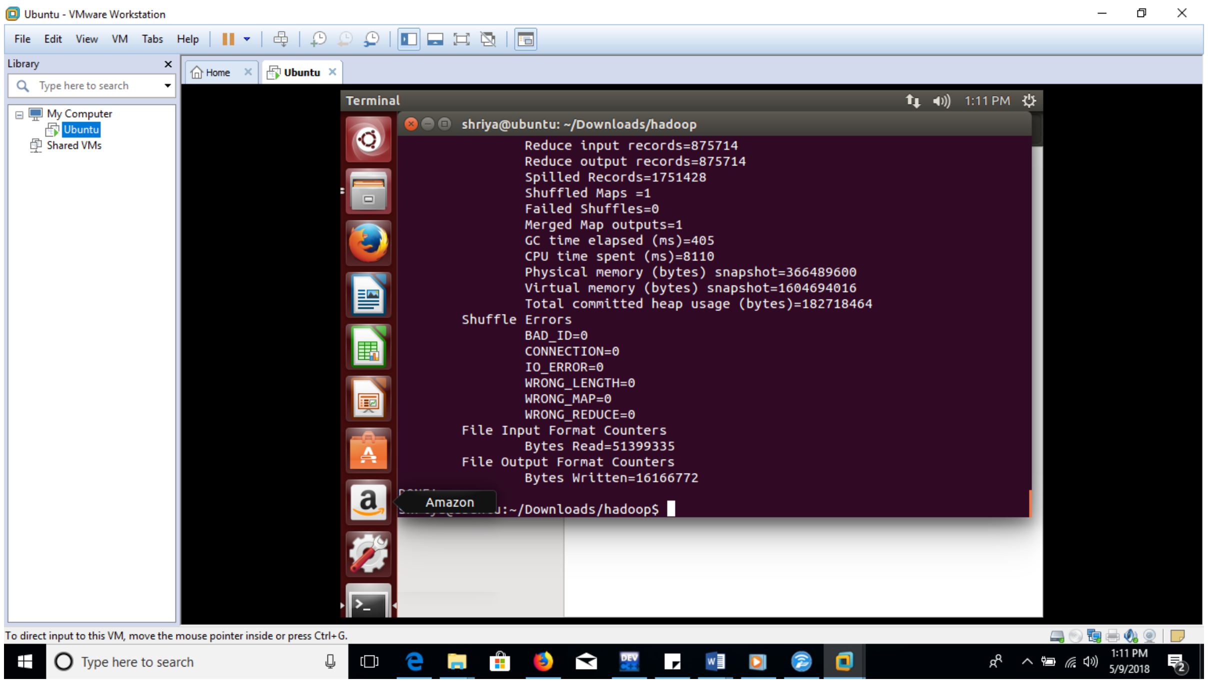
```
18/05/09 13:09:00 INFO mapreduce.Job:  map 0% reduce 0%
18/05/09 13:09:12 INFO mapreduce.Job:  map 100% reduce 0%
18/05/09 13:09:25 INFO mapreduce.Job:  map 100% reduce 100%
18/05/09 13:09:26 INFO mapreduce.Job: Job job_1525896198301_0004 completed succe
ssfully
18/05/09 13:09:26 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=14781669
                FILE: Number of bytes written=29809559
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=51399448
                HDFS: Number of bytes written=16166772
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=10336
                Total time spent by all reduces in occupied slots (ms)=9761
                Total time spent by all map tasks (ms)=10336
```

FIGURE 4.7: Output- Berkstein Dataset (I)


Ubuntu - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Shared VMs
Home
Ubuntu
Terminal
1:10 PM
shriya@ubuntu: ~/Downloads/hadoop
System Settings
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
Type here to search
1:10 PM
5/9/2018

```
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=10336
                Total time spent by all reduces in occupied slots (ms)=9761
                Total time spent by all map tasks (ms)=10336
                Total time spent by all reduce tasks (ms)=9761
                Total vcore-milliseconds taken by all map tasks=10336
                Total vcore-milliseconds taken by all reduce tasks=9761
                Total megabyte-milliseconds taken by all map tasks=10584064
                Total megabyte-milliseconds taken by all reduce tasks=9995264
        Map-Reduce Framework
                Map input records=875714
                Map output records=875714
                Map output bytes=13030235
                Map output materialized bytes=14781669
                Input split bytes=113
                Combine input records=0
                Combine output records=0
                Reduce input groups=1552
                Reduce shuffle bytes=14781669
                Reduce input records=875714
```

FIGURE 4.8: Output- Berkstein Dataset (II)


Ubuntu - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Shared VMs
Home
Ubuntu
Terminal
1:11 PM
shriya@ubuntu: ~/Downloads/hadoop
Amazon
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
Type here to search
1:11 PM
5/9/2018

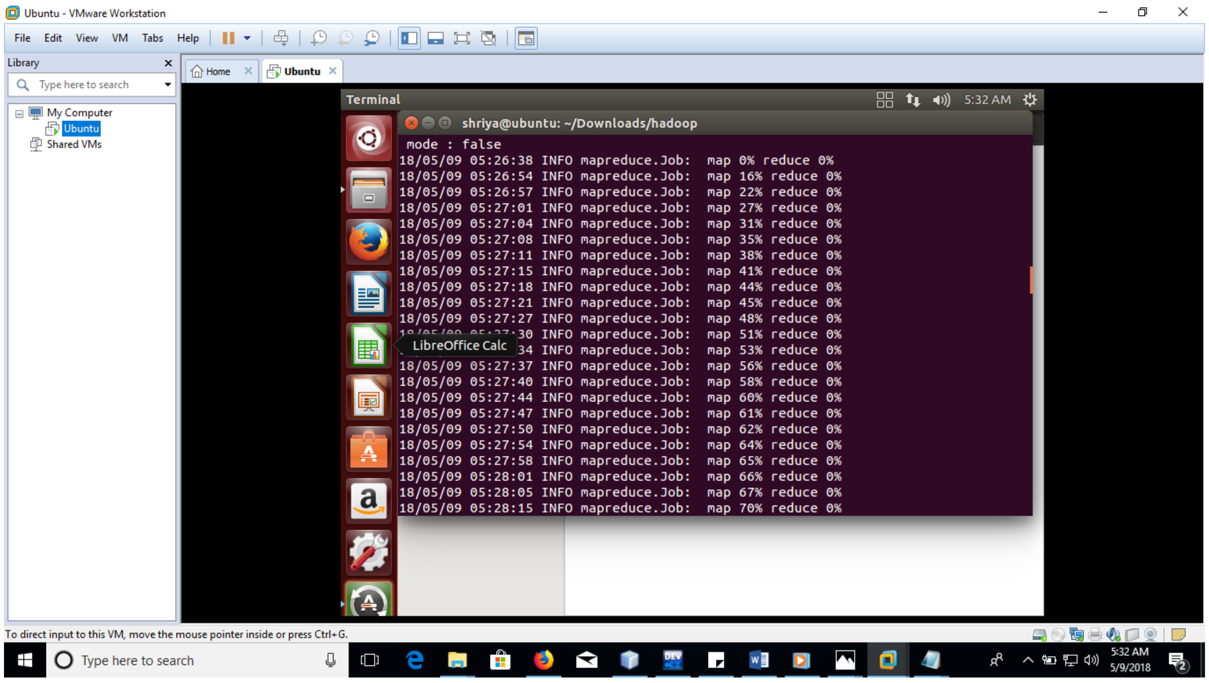
```
                Reduce input records=875714
                Reduce output records=875714
                Spilled Records=1751428
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=405
                CPU time spent (ms)=8110
                Physical memory (bytes) snapshot=366489600
                Virtual memory (bytes) snapshot=1604694016
                Total committed heap usage (bytes)=182718464
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=51399335
        File Output Format Counters
                Bytes Written=16166772
[illegible]
[illegible]:~/Downloads/hadoop$
```

FIGURE 4.9: Output- Berkstein Dataset (III)

	Google	Stanford	Berksten
Gc time elapsed	498	403	405
Cpu time	7680	7310	8110

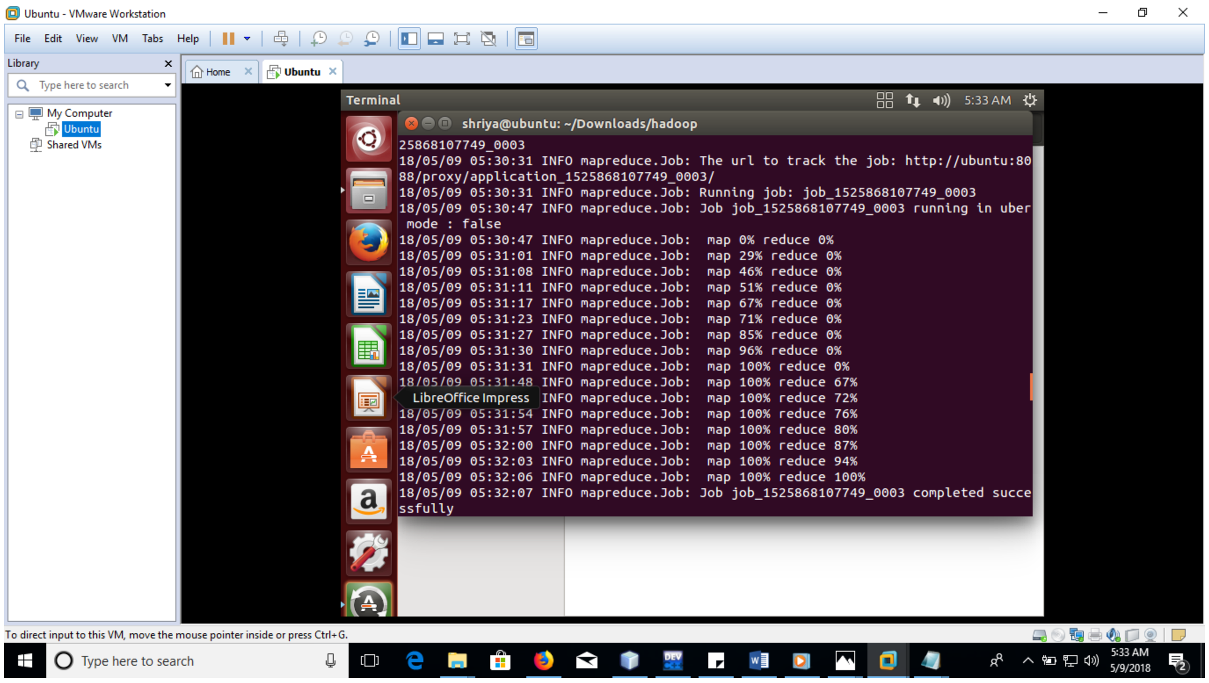
TABLE 4.2: Execution-time of each dataset

Mapper and Reducer jobs are as follows:


Ubuntu - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Shared VMs
Home
Ubuntu
Terminal
5:32 AM
shriya@ubuntu: ~/Downloads/hadoop
LibreOffice Calc
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
Type here to search
5:32 AM
5/9/2018

```
 mode : false
18/05/09 05:26:38 INFO mapreduce.Job:  map 0% reduce 0%
18/05/09 05:26:54 INFO mapreduce.Job:  map 16% reduce 0%
18/05/09 05:26:57 INFO mapreduce.Job:  map 22% reduce 0%
18/05/09 05:27:01 INFO mapreduce.Job:  map 27% reduce 0%
18/05/09 05:27:04 INFO mapreduce.Job:  map 31% reduce 0%
18/05/09 05:27:08 INFO mapreduce.Job:  map 35% reduce 0%
18/05/09 05:27:11 INFO mapreduce.Job:  map 38% reduce 0%
18/05/09 05:27:15 INFO mapreduce.Job:  map 41% reduce 0%
18/05/09 05:27:18 INFO mapreduce.Job:  map 44% reduce 0%
18/05/09 05:27:21 INFO mapreduce.Job:  map 45% reduce 0%
18/05/09 05:27:27 INFO mapreduce.Job:  map 48% reduce 0%
18/05/09 05:27:30 INFO mapreduce.Job:  map 51% reduce 0%
                 34 INFO mapreduce.Job:  map 53% reduce 0%
18/05/09 05:27:37 INFO mapreduce.Job:  map 56% reduce 0%
18/05/09 05:27:40 INFO mapreduce.Job:  map 58% reduce 0%
18/05/09 05:27:44 INFO mapreduce.Job:  map 60% reduce 0%
18/05/09 05:27:47 INFO mapreduce.Job:  map 61% reduce 0%
18/05/09 05:27:50 INFO mapreduce.Job:  map 62% reduce 0%
18/05/09 05:27:54 INFO mapreduce.Job:  map 64% reduce 0%
18/05/09 05:27:58 INFO mapreduce.Job:  map 65% reduce 0%
18/05/09 05:28:01 INFO mapreduce.Job:  map 66% reduce 0%
18/05/09 05:28:05 INFO mapreduce.Job:  map 67% reduce 0%
18/05/09 05:28:15 INFO mapreduce.Job:  map 70% reduce 0%
```

FIGURE 4.10: Mapper-Reducer for Job1


Ubuntu - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu
Shared VMs
Home
Ubuntu
Terminal
5:33 AM
shriya@ubuntu: ~/Downloads/hadoop
LibreOffice Impress
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
Type here to search
5:33 AM
5/9/2018

```
25868107749_0003
18/05/09 05:30:31 INFO mapreduce.Job: The url to track the job: http://ubuntu:80
88/proxy/application_1525868107749_0003/
18/05/09 05:30:31 INFO mapreduce.Job: Running job: job_1525868107749_0003
18/05/09 05:30:47 INFO mapreduce.Job: Job job_1525868107749_0003 running in uber
 mode : false
18/05/09 05:30:47 INFO mapreduce.Job:  map 0% reduce 0%
18/05/09 05:31:01 INFO mapreduce.Job:  map 29% reduce 0%
18/05/09 05:31:08 INFO mapreduce.Job:  map 46% reduce 0%
18/05/09 05:31:11 INFO mapreduce.Job:  map 51% reduce 0%
18/05/09 05:31:17 INFO mapreduce.Job:  map 67% reduce 0%
18/05/09 05:31:23 INFO mapreduce.Job:  map 71% reduce 0%
18/05/09 05:31:27 INFO mapreduce.Job:  map 85% reduce 0%
18/05/09 05:31:30 INFO mapreduce.Job:  map 96% reduce 0%
18/05/09 05:31:31 INFO mapreduce.Job:  map 100% reduce 0%
18/05/09 05:31:48 INFO mapreduce.Job:  map 100% reduce 67%
                  INFO mapreduce.Job:  map 100% reduce 72%
18/05/09 05:31:54 INFO mapreduce.Job:  map 100% reduce 76%
18/05/09 05:31:57 INFO mapreduce.Job:  map 100% reduce 80%
18/05/09 05:32:00 INFO mapreduce.Job:  map 100% reduce 87%
18/05/09 05:32:03 INFO mapreduce.Job:  map 100% reduce 94%
18/05/09 05:32:06 INFO mapreduce.Job:  map 100% reduce 100%
18/05/09 05:32:07 INFO mapreduce.Job: Job job_1525868107749_0003 completed succe
ssfully
```

FIGURE 4.10: Mapper-Reducer for Job2

CHAPTER 5: CONCLUSION

5.1 CONCLUSION

In this paper a new and modified ranking algorithm is proposed which considers user preference with handy considerations of the content of web page to efficiently rank the web pages. Comparison of the proposed algorithm with one of the page ranking algorithms i.e. Weighted Page Rank Algorithm. Preference based algorithm makes search navigation easier. Ranking algorithms which consider web structure mining are comparatively less relevant to user query as they have no idea about user trends on the topic.

Ranking algorithm based on web content mining ignores importance of webpage. On the other hand, our proposed algorithm tries to overcome the above-mentioned limitations by taking both user tends and content of webpage hand in hand. Moreover, it avoids similarity in ranking and is more dynamic in nature.

5.2 FUTURE SCOPE

Currently, Google and many other search-engines rank documents on the web and exhibits the link as a result to its users. In Future, it might aggregate information hidden in these links that might be relevant to the user query and present that information directly instead of links. This idea proves to be detrimental to the search-engine and the resources like Stack Over Flow, which is among the top ranked websites by google and many other search-engines. Hence, not an idea which could viably be used commercially. However, development of some unique applications that might use this idea can occur.

In past few years, Google has tried to create some amendments and result in some internal versions of its search-engine that work wholly without links. Although the outputs are horrendous for now. But this also implies that the value which links hold have come under question at google. Links will start losing their weight due to the folding in of some other metrics and hike in algorithmic and manual sanction.

A Russian search-engine, Yandex, is attempting to work without links replacing it with vertical metrics. It is a publicly-traded and an utmost popular search engine.

REFERENCES

Books and Research Papers:

[1] Neelam Duhan et al, "Page ranking algorithm: A survey" in IEEE International Advance Computing Conference, Patiala, 2009.

[2] C. Deisy et al, "A Novel Relation-Based Probability Algorithm for PageRanking in Semantic Web Search Engine" presented at Department of Computer Science and Engineering, Thiagarajar College of Engineering, Madurai, TamilNadu, 2011.

[3] AdityaPratap Singh et al, "An Efficient Algorithm for Ranking Research Papers Based on Citation Network" in 3rd Conference on Data Mining and Optimization (DMO), Selangor, Malaysia, 28-29 June,2011.

[4] Lissa Rodrigues et al, "Hybrid Model for Improvised Page Ranking Algorithm" in International Conference on Control, Instrumentation, Communication and Computational Technologies (lCCICCT), Mumbai, Maharashtra, 2015.

[5] Lissa Rodrigues et al, "An Efficient Page Ranking Approach Based on Hybrid Model" in Second International Conference on Advances in Computing and Communication Engineering, Mumbai, Maharashtra, 2015.

[6] Dr. Daya Gupta et al, "User Preference Based Page Ranking Algorithm" in International Conference on Computing, Communication and Automation, Delhi, India, 2016.

Websites:

https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm

https://intellipaat.com/tutorial/big-data-and-hadoop-tutorial/hadoop-multi-node-clusters/

https://www.dezyre.com/hadoop-tutorial/hadoop-mapreduce-wordcount-tutorial

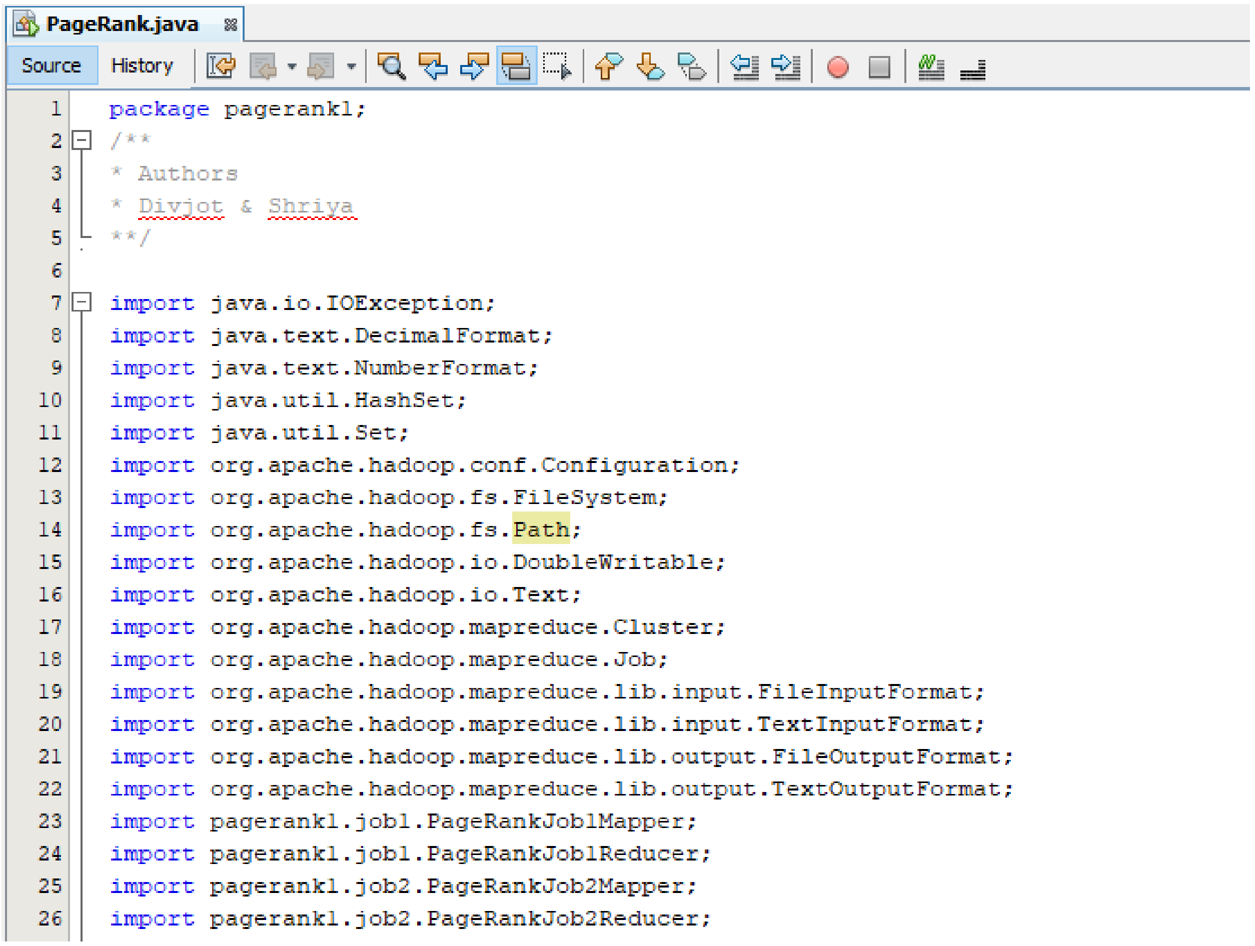
https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm

www.thecloudavenue.com/2016/04/pagerank-on-english-wiki-data-using.html

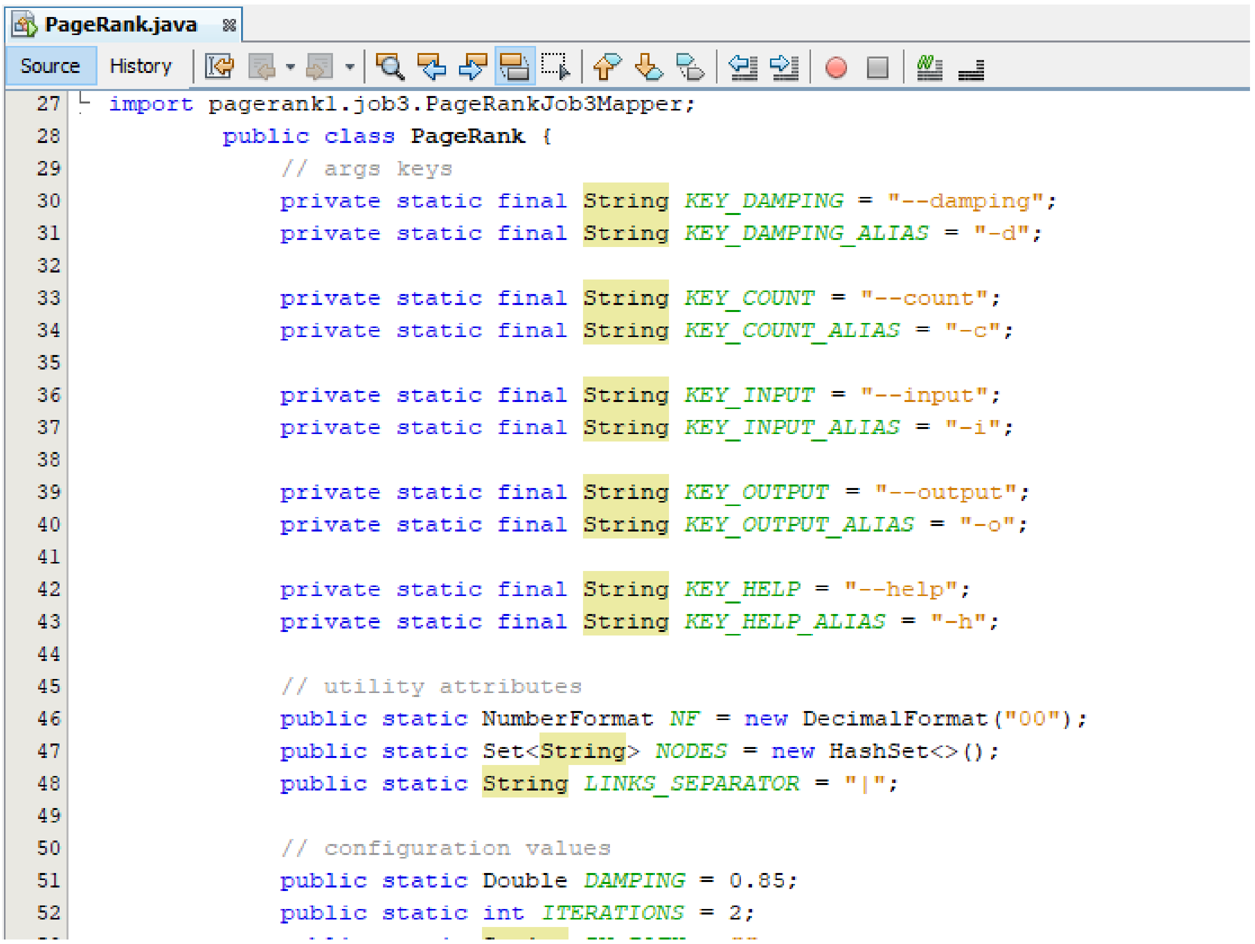
http://www.geeksforgeeks.org/page-rank-algorithm-implementation/
https://www.youtube.com/watch?v=EDcXRPKk7Qk&t=2906s
https://www.youtube.com/watch?v=-YEcJquYsFo&t=2192s
https://www.youtube.com/watch?v=8jtkR_h0aUw

Appendices

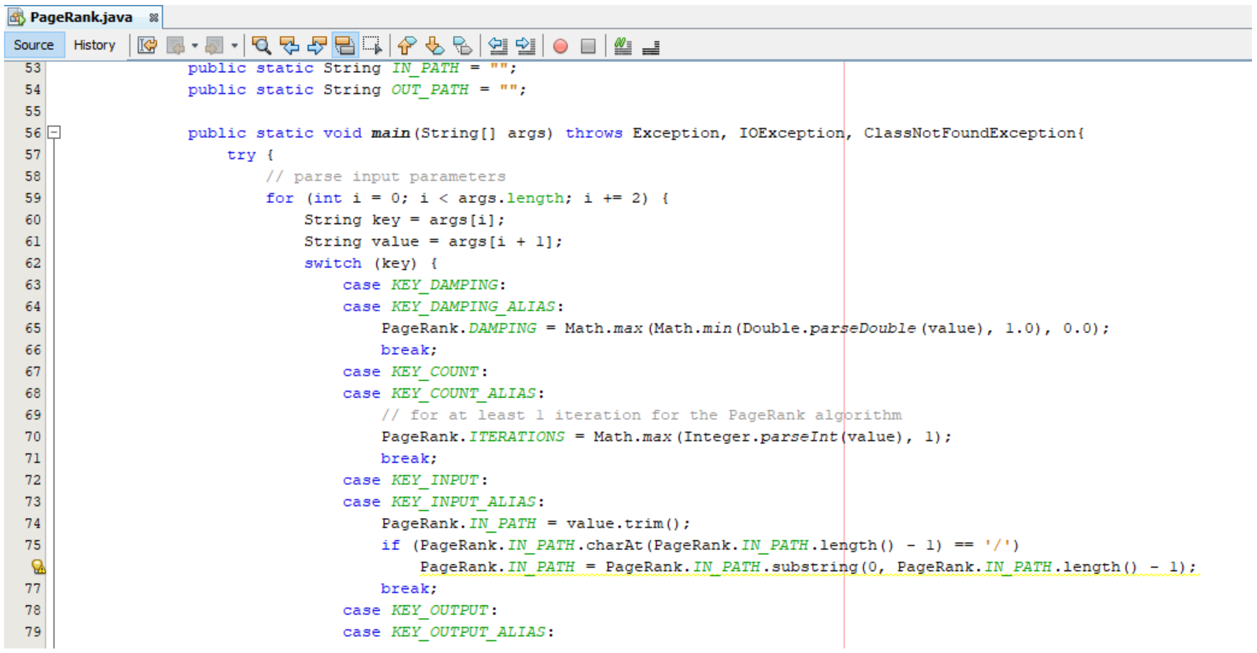
1. Configuration Class



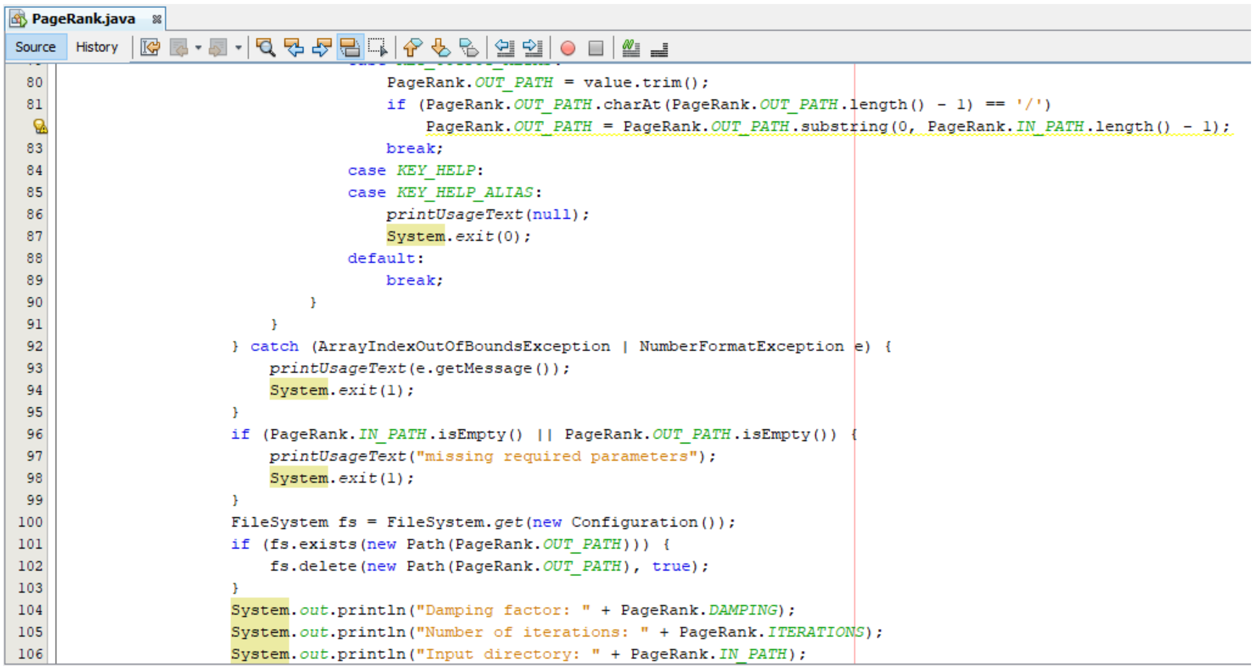
```java
package pagerank1;
/**
 * Authors
 * Divjot & Shriya
 **/

import java.io.IOException;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.HashSet;
import java.util.Set;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Cluster;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import pagerank1.job1.PageRankJob1Mapper;
import pagerank1.job1.PageRankJob1Reducer;
import pagerank1.job2.PageRankJob2Mapper;
import pagerank1.job2.PageRankJob2Reducer;
```



```java
import pagerank1.job3.PageRankJob3Mapper;
        public class PageRank {
            // args keys
            private static final String KEY_DAMPING = "--damping";
            private static final String KEY_DAMPING_ALIAS = "-d";

            private static final String KEY_COUNT = "--count";
            private static final String KEY_COUNT_ALIAS = "-c";

            private static final String KEY_INPUT = "--input";
            private static final String KEY_INPUT_ALIAS = "-i";

            private static final String KEY_OUTPUT = "--output";
            private static final String KEY_OUTPUT_ALIAS = "-o";

            private static final String KEY_HELP = "--help";
            private static final String KEY_HELP_ALIAS = "-h";

            // utility attributes
            public static NumberFormat NF = new DecimalFormat("00");
            public static Set<String> NODES = new HashSet<>();
            public static String LINKS_SEPARATOR = "|";

            // configuration values
            public static Double DAMPING = 0.85;
            public static int ITERATIONS = 2;
```

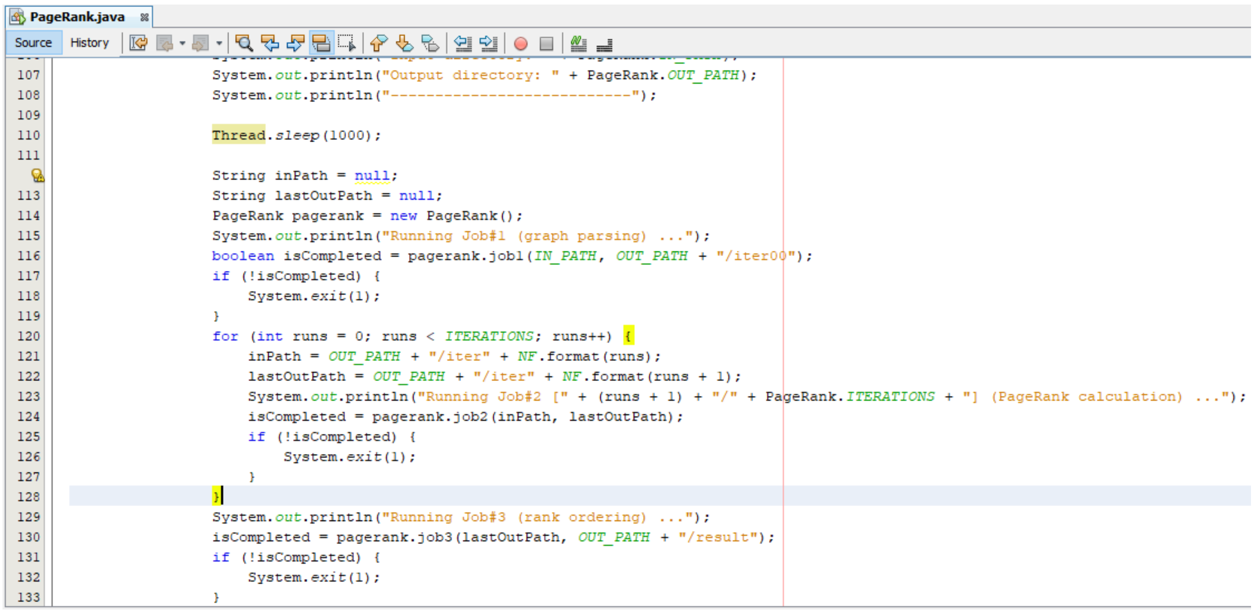


```java
            public static String IN_PATH = "";
            public static String OUT_PATH = "";

            public static void main(String[] args) throws Exception, IOException, ClassNotFoundException{
                try {
                    // parse input parameters
                    for (int i = 0; i < args.length; i += 2) {
                        String key = args[i];
                        String value = args[i + 1];
                        switch (key) {
                            case KEY_DAMPING:
                            case KEY_DAMPING_ALIAS:
                                PageRank.DAMPING = Math.max(Math.min(Double.parseDouble(value), 1.0), 0.0);
                                break;
                            case KEY_COUNT:
                            case KEY_COUNT_ALIAS:
                                // for at least 1 iteration for the PageRank algorithm
                                PageRank.ITERATIONS = Math.max(Integer.parseInt(value), 1);
                                break;
                            case KEY_INPUT:
                            case KEY_INPUT_ALIAS:
                                PageRank.IN_PATH = value.trim();
                                if (PageRank.IN_PATH.charAt(PageRank.IN_PATH.length() - 1) == '/')
                                    PageRank.IN_PATH = PageRank.IN_PATH.substring(0, PageRank.IN_PATH.length() - 1);
                                break;
                            case KEY_OUTPUT:
                            case KEY_OUTPUT_ALIAS:
```


PageRank.java

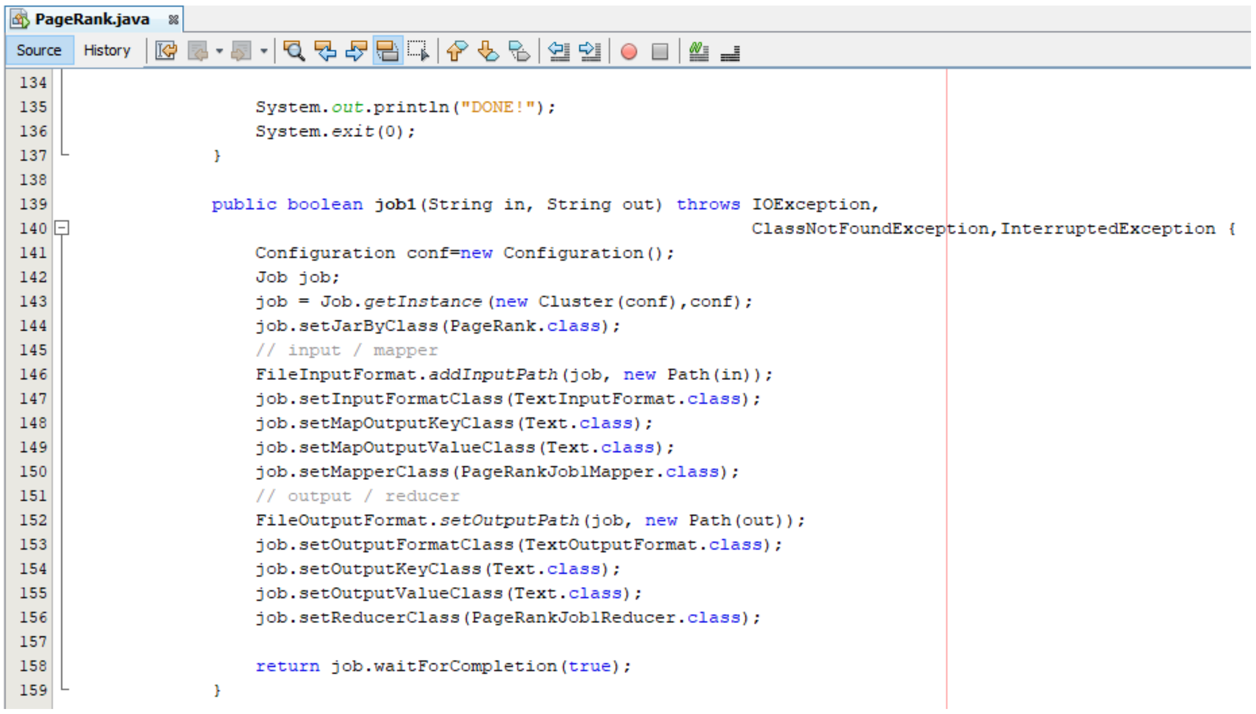
Source History

```
                        PageRank.OUT_PATH = value.trim();
                        if (PageRank.OUT_PATH.charAt(PageRank.OUT_PATH.length() - 1) == '/')
                            PageRank.OUT_PATH = PageRank.OUT_PATH.substring(0, PageRank.IN_PATH.length() - 1);
                        break;
                    case KEY_HELP:
                    case KEY_HELP_ALIAS:
                        printUsageText(null);
                        System.exit(0);
                    default:
                        break;
                }
            }
        } catch (ArrayIndexOutOfBoundsException | NumberFormatException e) {
            printUsageText(e.getMessage());
            System.exit(1);
        }
        if (PageRank.IN_PATH.isEmpty() || PageRank.OUT_PATH.isEmpty()) {
            printUsageText("missing required parameters");
            System.exit(1);
        }
        FileSystem fs = FileSystem.get(new Configuration());
        if (fs.exists(new Path(PageRank.OUT_PATH))) {
            fs.delete(new Path(PageRank.OUT_PATH), true);
        }
        System.out.println("Damping factor: " + PageRank.DAMPING);
        System.out.println("Number of iterations: " + PageRank.ITERATIONS);
        System.out.println("Input directory: " + PageRank.IN_PATH);
```


PageRank.java

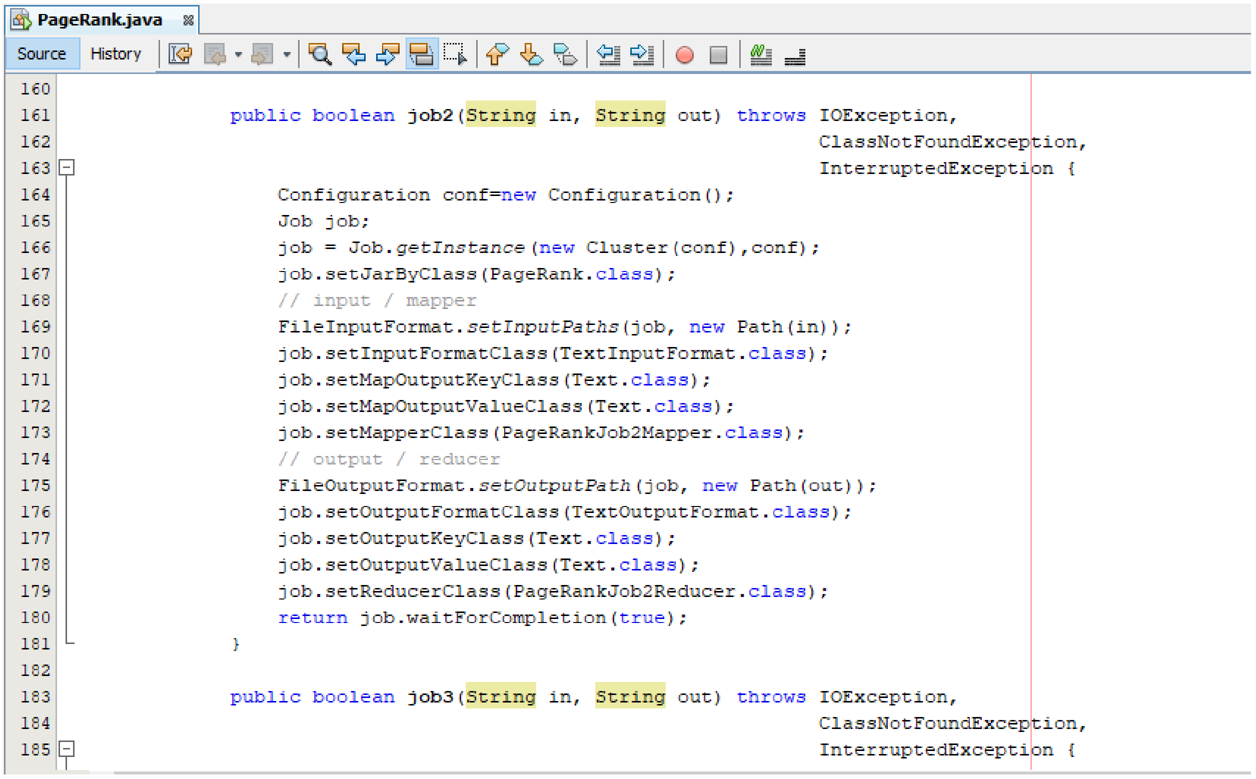
Source History

```
        System.out.println("Output directory: " + PageRank.OUT_PATH);
        System.out.println("---------------------------");

        Thread.sleep(1000);

        String inPath = null;
        String lastOutPath = null;
        PageRank pagerank = new PageRank();
        System.out.println("Running Job#1 (graph parsing) ...");
        boolean isCompleted = pagerank.job1(IN_PATH, OUT_PATH + "/iter00");
        if (!isCompleted) {
            System.exit(1);
        }
        for (int runs = 0; runs < ITERATIONS; runs++) {
            inPath = OUT_PATH + "/iter" + NF.format(runs);
            lastOutPath = OUT_PATH + "/iter" + NF.format(runs + 1);
            System.out.println("Running Job#2 [" + (runs + 1) + "/" + PageRank.ITERATIONS + "] (PageRank calculation) ...");
            isCompleted = pagerank.job2(inPath, lastOutPath);
            if (!isCompleted) {
                System.exit(1);
            }
        }
        System.out.println("Running Job#3 (rank ordering) ...");
        isCompleted = pagerank.job3(lastOutPath, OUT_PATH + "/result");
        if (!isCompleted) {
            System.exit(1);
        }
```


PageRank.java

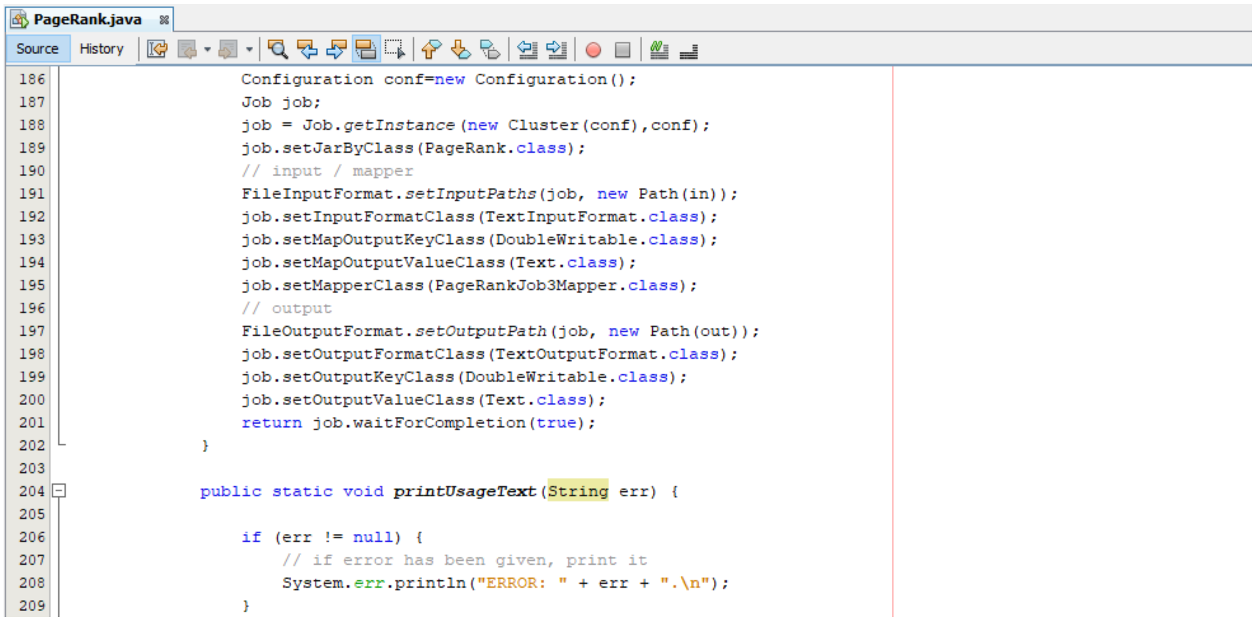
Source History

```
            System.out.println("DONE!");
            System.exit(0);
        }

        public boolean job1(String in, String out) throws IOException,
                                                          ClassNotFoundException,InterruptedException {
            Configuration conf=new Configuration();
            Job job;
            job = Job.getInstance(new Cluster(conf),conf);
            job.setJarByClass(PageRank.class);
            // input / mapper
            FileInputFormat.addInputPath(job, new Path(in));
            job.setInputFormatClass(TextInputFormat.class);
            job.setMapOutputKeyClass(Text.class);
            job.setMapOutputValueClass(Text.class);
            job.setMapperClass(PageRankJob1Mapper.class);
            // output / reducer
            FileOutputFormat.setOutputPath(job, new Path(out));
            job.setOutputFormatClass(TextOutputFormat.class);
            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(Text.class);
            job.setReducerClass(PageRankJob1Reducer.class);

            return job.waitForCompletion(true);
        }
```

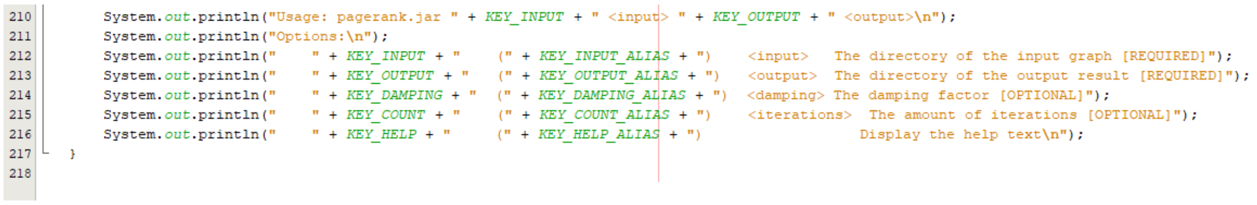

PageRank.java

Source History

```

        public boolean job2(String in, String out) throws IOException,
                                                          ClassNotFoundException,
                                                          InterruptedException {
            Configuration conf=new Configuration();
            Job job;
            job = Job.getInstance(new Cluster(conf),conf);
            job.setJarByClass(PageRank.class);
            // input / mapper
            FileInputFormat.setInputPaths(job, new Path(in));
            job.setInputFormatClass(TextInputFormat.class);
            job.setMapOutputKeyClass(Text.class);
            job.setMapOutputValueClass(Text.class);
            job.setMapperClass(PageRankJob2Mapper.class);
            // output / reducer
            FileOutputFormat.setOutputPath(job, new Path(out));
            job.setOutputFormatClass(TextOutputFormat.class);
            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(Text.class);
            job.setReducerClass(PageRankJob2Reducer.class);
            return job.waitForCompletion(true);
        }

        public boolean job3(String in, String out) throws IOException,
                                                          ClassNotFoundException,
                                                          InterruptedException {
```



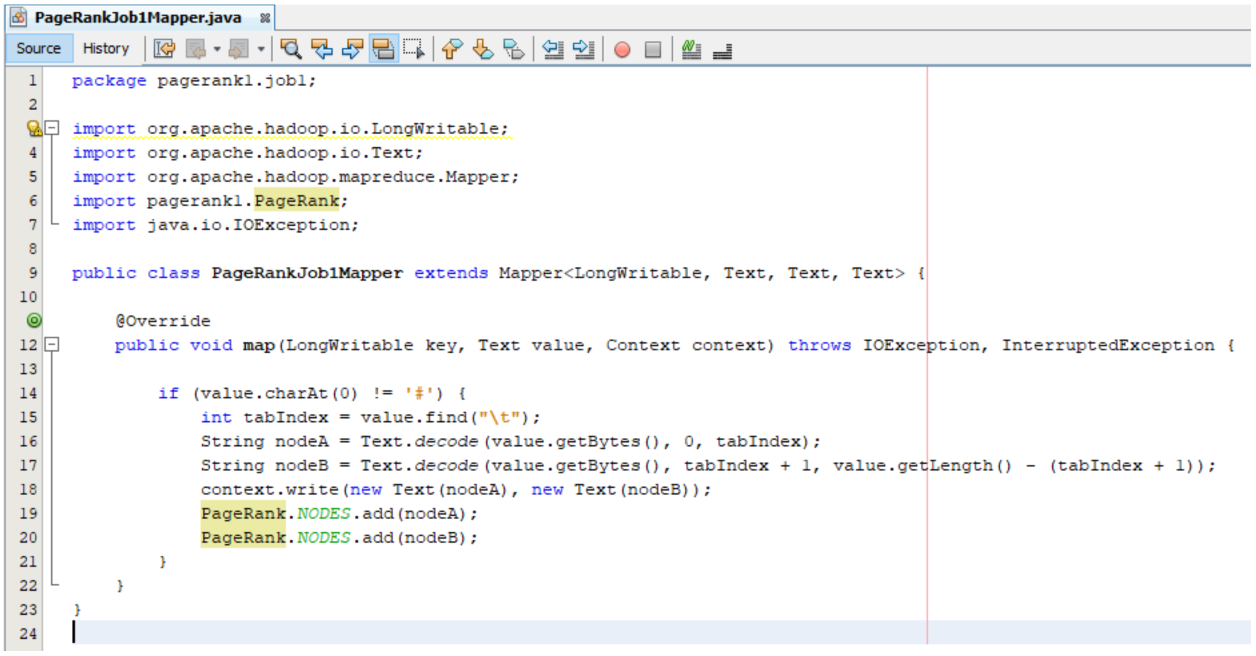
```java
        Configuration conf=new Configuration();
        Job job;
        job = Job.getInstance(new Cluster(conf),conf);
        job.setJarByClass(PageRank.class);
        // input / mapper
        FileInputFormat.setInputPaths(job, new Path(in));
        job.setInputFormatClass(TextInputFormat.class);
        job.setMapOutputKeyClass(DoubleWritable.class);
        job.setMapOutputValueClass(Text.class);
        job.setMapperClass(PageRankJob3Mapper.class);
        // output
        FileOutputFormat.setOutputPath(job, new Path(out));
        job.setOutputFormatClass(TextOutputFormat.class);
        job.setOutputKeyClass(DoubleWritable.class);
        job.setOutputValueClass(Text.class);
        return job.waitForCompletion(true);
    }

    public static void printUsageText(String err) {

        if (err != null) {
            // if error has been given, print it
            System.err.println("ERROR: " + err + ".\n");
        }
```



```java
        System.out.println("Usage: pagerank.jar " + KEY_INPUT + " <input> " + KEY_OUTPUT + " <output>\n");
        System.out.println("Options:\n");
        System.out.println("    " + KEY_INPUT + "    (" + KEY_INPUT_ALIAS + ")    <input>   The directory of the input graph [REQUIRED]");
        System.out.println("    " + KEY_OUTPUT + "   (" + KEY_OUTPUT_ALIAS + ")   <output>  The directory of the output result [REQUIRED]");
        System.out.println("    " + KEY_DAMPING + "  (" + KEY_DAMPING_ALIAS + ")  <damping> The damping factor [OPTIONAL]");
        System.out.println("    " + KEY_COUNT + "    (" + KEY_COUNT_ALIAS + ")    <iterations>  The amount of iterations [OPTIONAL]");
        System.out.println("    " + KEY_HELP + "     (" + KEY_HELP_ALIAS + ")                  Display the help text\n");
    }
```

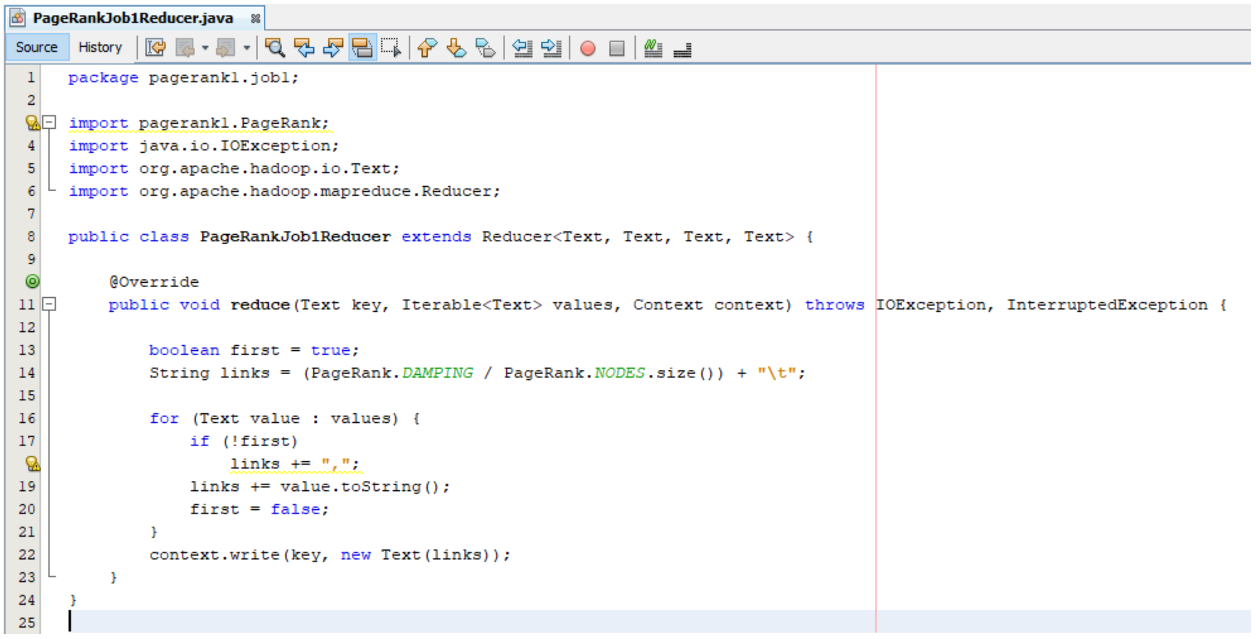
2. Mapper Reducer Class1

Mapper Class



```java
package pagerank1.job1;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import pagerank1.PageRank;
import java.io.IOException;

public class PageRankJob1Mapper extends Mapper<LongWritable, Text, Text, Text> {

    @Override
    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

        if (value.charAt(0) != '#') {
            int tabIndex = value.find("\t");
            String nodeA = Text.decode(value.getBytes(), 0, tabIndex);
            String nodeB = Text.decode(value.getBytes(), tabIndex + 1, value.getLength() - (tabIndex + 1));
            context.write(new Text(nodeA), new Text(nodeB));
            PageRank.NODES.add(nodeA);
            PageRank.NODES.add(nodeB);
        }
    }
}
```

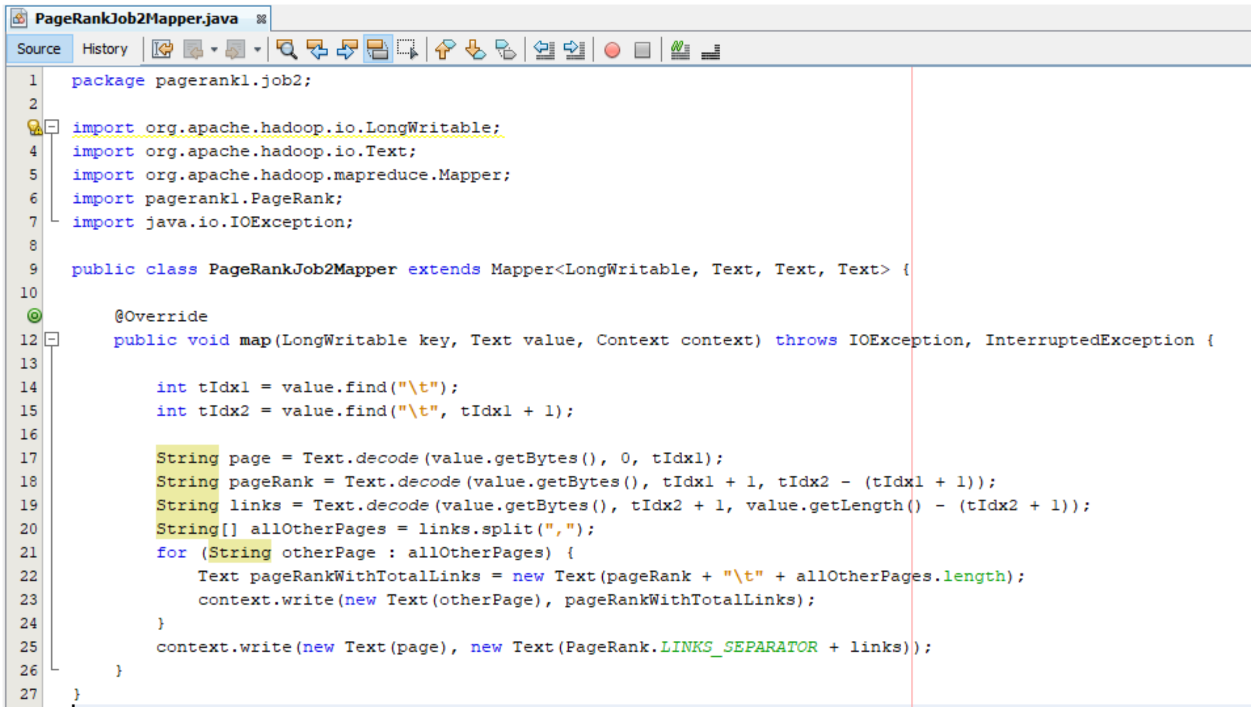
Reducer Class



```java
package pagerank1.job1;

import pagerank1.PageRank;
import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class PageRankJob1Reducer extends Reducer<Text, Text, Text, Text> {

    @Override
    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

        boolean first = true;
        String links = (PageRank.DAMPING / PageRank.NODES.size()) + "\t";

        for (Text value : values) {
            if (!first)
                links += ",";
            links += value.toString();
            first = false;
        }
        context.write(key, new Text(links));
    }
}
```

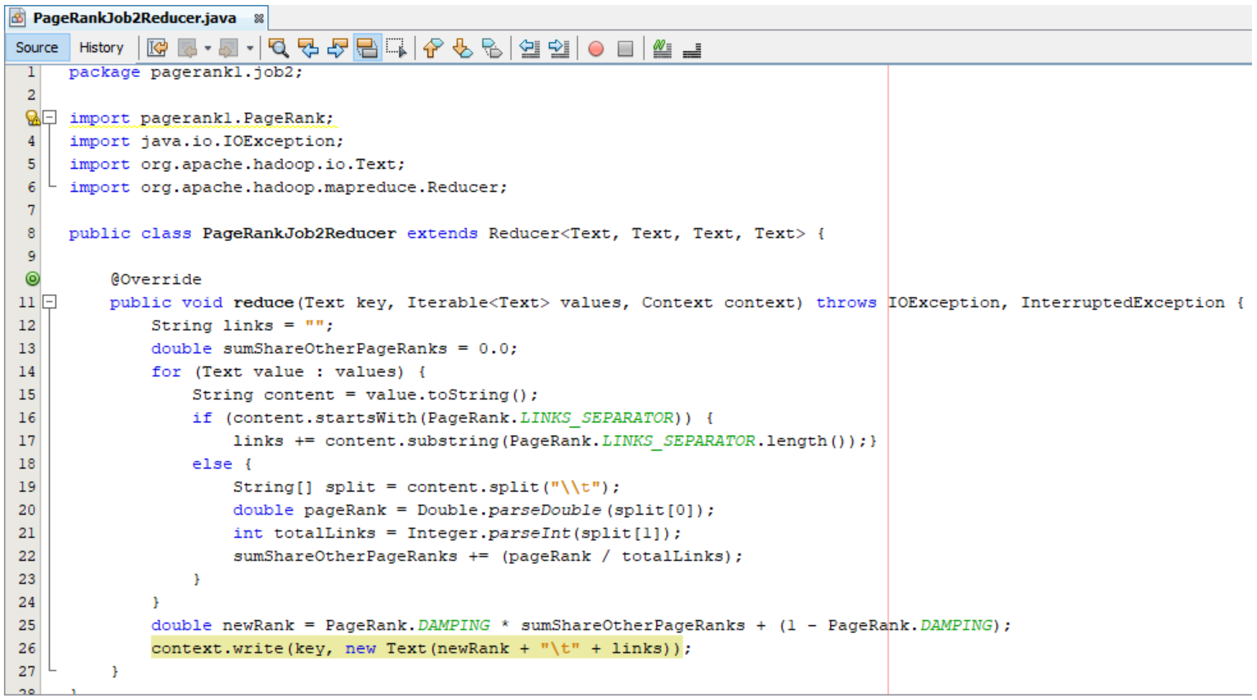
3. Mapper Reducer Class2

Mapper Class



```java
package pagerank1.job2;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import pagerank1.PageRank;
import java.io.IOException;

public class PageRankJob2Mapper extends Mapper<LongWritable, Text, Text, Text> {

    @Override
    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

        int tIdx1 = value.find("\t");
        int tIdx2 = value.find("\t", tIdx1 + 1);

        String page = Text.decode(value.getBytes(), 0, tIdx1);
        String pageRank = Text.decode(value.getBytes(), tIdx1 + 1, tIdx2 - (tIdx1 + 1));
        String links = Text.decode(value.getBytes(), tIdx2 + 1, value.getLength() - (tIdx2 + 1));
        String[] allOtherPages = links.split(",");
        for (String otherPage : allOtherPages) {
            Text pageRankWithTotalLinks = new Text(pageRank + "\t" + allOtherPages.length);
            context.write(new Text(otherPage), pageRankWithTotalLinks);
        }
        context.write(new Text(page), new Text(PageRank.LINKS_SEPARATOR + links));
    }
}
```

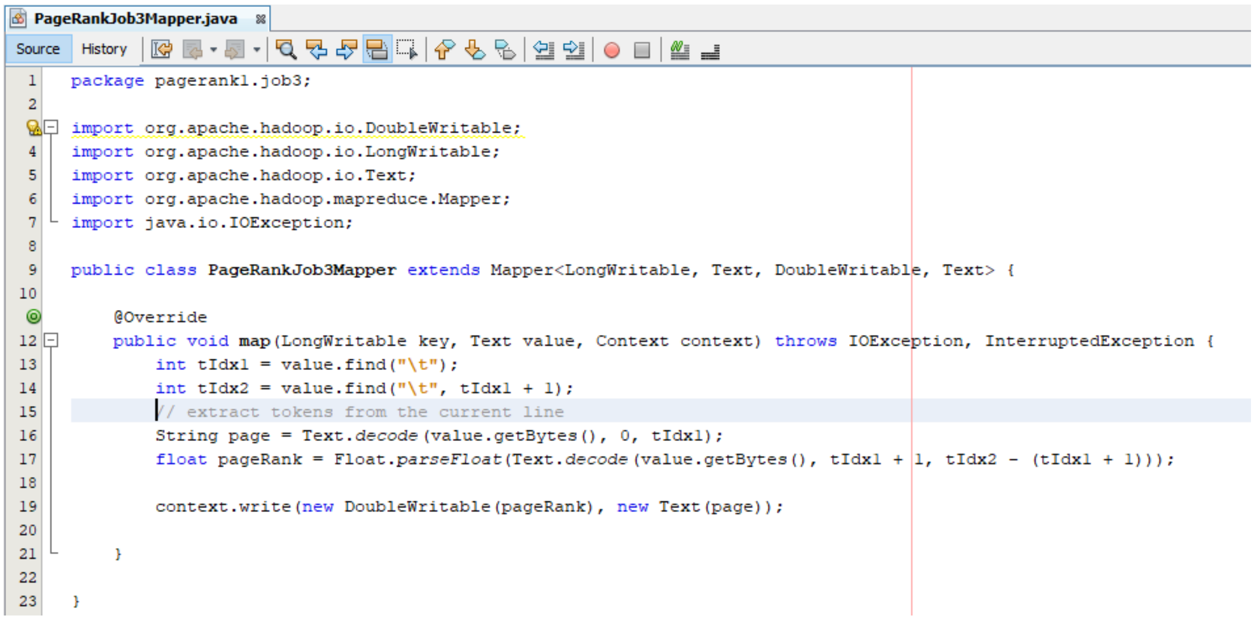
Reducer Class



```java
package pagerank1.job2;

import pagerank1.PageRank;
import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class PageRankJob2Reducer extends Reducer<Text, Text, Text, Text> {

    @Override
    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
        String links = "";
        double sumShareOtherPageRanks = 0.0;
        for (Text value : values) {
            String content = value.toString();
            if (content.startsWith(PageRank.LINKS_SEPARATOR)) {
                links += content.substring(PageRank.LINKS_SEPARATOR.length());}
            else {
                String[] split = content.split("\\t");
                double pageRank = Double.parseDouble(split[0]);
                int totalLinks = Integer.parseInt(split[1]);
                sumShareOtherPageRanks += (pageRank / totalLinks);
            }
        }
        double newRank = PageRank.DAMPING * sumShareOtherPageRanks + (1 - PageRank.DAMPING);
        context.write(key, new Text(newRank + "\t" + links));
    }
}
```

4. Mapper Class 3

Mapper Class



```java
package pagerank1.job3;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;

public class PageRankJob3Mapper extends Mapper<LongWritable, Text, DoubleWritable, Text> {

    @Override
    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        int tIdx1 = value.find("\t");
        int tIdx2 = value.find("\t", tIdx1 + 1);
        // extract tokens from the current line
        String page = Text.decode(value.getBytes(), 0, tIdx1);
        float pageRank = Float.parseFloat(Text.decode(value.getBytes(), tIdx1 + 1, tIdx2 - (tIdx1 + 1)));

        context.write(new DoubleWritable(pageRank), new Text(page));

    }

}
```