BRAC UNIVERSITY

Detecting Document Similarity in Large Document Collection using MapReduce and the Hadoop Framework.

A Thesis submitted in partial fulfillment of the
Requirement for the degree Bachelor of Science

in

Computer Science and Engineering

by

AnikMomtaz (ID: 08201002)
SadikaAmreen (ID: 09101003)

Supervisor:

Dr. Mumit Khan

December, 2012

CERTIFICATE

This is to certify that the thesis entitled "Detecting Document Similarity in Large Document Collection using MapReduce and the Hadoop Framework" is submitted by Anik Momtaz (ID: 08201002) and Sadika Amreen (ID: 09101003) to the Department of Computer Science and Engineering, School of Engineering and Computer Science, BRAC University, for the award of the degree in Bachelor of Science in Computer Science and Engineering is a bonafide record of work carried out by them under my supervision. The contents of this thesis have not been submitted to any other Institute or University for the award of any degree or Diploma.

_____

Imran Ahmed
Co-Supervisor

_____

Dr. Mumit Khan
Supervisor

_____

Prof. Mohammad Zahidur Rahman
Dean

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

We would like to thank our adviser Dr. Mumit Khan for his guidance. We have learned a great amount working on this thesis and much of that is due to him. We would like to thank our teacher Imran Ahmed who was always willing to answer our questions, who worked alongside us as a guide during our workand who provided a great deal of advice.

We would also like to thank our teachers AnnajiatAlimRasel and Mohammad ShamsulKaonain for providing us with support and facilities as well as for ushering us through the final stage of graduation.

Finally, we would like to thank our families; without the love and support of which none of this would have been possible.

ABSTRACT OF THE THESIS

Detecting Document Similarity in Large Document Collection using MapReduce and the Hadoop Framework.

By

AnikMomtaz (ID: 08201002)
Sadika Amreen (ID: 09101003)

Bachelor of Science in Computer Science
BRAC University, 2012

Dr. Mumit Khan, Supervisor

The everlasting necessity to process data is only becoming more and more challenging due to the exponential growth of the data itself. We are talking about exabytes, zettabytes and even yottabytes of data; generally referred to as Big Data. Hence, the conventional processing methods of data have become obsolete when handling Big Data. It is simply not feasible to use a single machine to analyze data of such tremendous volume.

This is where Hadoop comes in. Simply put, using the Hadoop Distributive File System (HDFS), an enormous chunk of data can be divided into smaller pieces and be distributed amongst multiple machines referred to as nodes to parallel process them using a technique called MapReduce.

The potential for such a concept is limitless. However, for our thesis, we have used the HDFS to identify similarities between multiple documents. The initial idea was to make an algorithm to detect full or partial plagiarism in documents as there are countless materials of interest readily available on the internet.

However, upon successfully being able to implement an algorithm for the English language, we realized that there is no record of any work on document similarity detection carried on upon Bangla language. Therefore, with some modifications to our existing algorithm to fit our specifications (as the Bangla language is completely different from the English language as far as construction is concerned), we were able to develop an algorithm to detect document similarities on a broad scale using the Ferret model.

Chapter 1

Introduction

Data is growing fast. About ninety percent of the world's existing data was created in the last year.The amount of digital content on the web is now close to five hundred billion gigabytes and the number is expected to double within a year. The explosion of mobile networks, cloud computing and new technologies has given rise to incomprehensibly large worlds of information. The necessity to manage efficiently the exponentially growing dataset is increasing everyday. Data not only needs to be processed and analyzed fast, a guarantee of a reliable backup such that there is no data loss, also needs to be provided.

The good news is that Big Data is here. The bad news is that we are struggling to store and analyze it. The problem is simple: while the storage capacities of hard drives have increased massively over the years, access speeds – the rate at which data can be read from drives have not kept up. It takes a long time to read all data on a single drive and writing is even slower. The obvious way to reduce the time is to read from multiple disks at once. Working in parallel would mean saving a lot more time through a trade off with resource and computational power. Processing and analyzing data in the minimum possible time is of utmost importance nowadays.

The traditional data management tools such as the RDBMS, no longer prove to be adequate in handling this explosion in data. To keep up with large scale distributed data generation, a large scale distributed data storage is needed which should be scalable as well.

This report gives an overview of the new ways to handle such large datasets by iterating over theMapReduce technique. It also focuses on the Hadoop framework and the Hadoop Distributed File System which uses the MapReduce algorithm to manage the extensively large amounts of data by splitting up huge datasets across multiple servers and parallely processing each part and then combining the answers of each part to produce the final answer. The report then extensively covers Document Similarity Detection using

MapReduce, beginning with the various techniques that are applicable to detect similarity between documents and then focuses on an analysis of a particular approach namely the Ferret Model.

Document similarity detection tools guarantee the privacy protection of documents. As the Internet holds large repositories of data which is easily accessible to all, the problem of plagiarism is likely to arise. With this rising problem, we need to develop a tool that not only protects the copyright of documents, but does them at a fast rate. Data repositories building at an exponential rate means that we need to run a sample of text against huge datasets for similarity detection. The MapReduce approach is a faster technique of doing such jobs on large sets of data as it exploits the advantage of parallel computation. This thesis mainly deals with detecting syntactical similarity between documents in large document sets that will prevent plagiarized documents to be acknowledged.

Tools for detecting document similarity such as plagiarism detectors for the English Language have already been created through several research works. However, such tools are not available for the Bangla Language which will enable us to compare pairs of texts or documents in Bangla in large document collections. The need and unavailability of such detectors is the motivation behind our work. Our research therefore focuses on developing a Text Similarity Detector using the Ferret Model for the Bangla Language.

1.1 What is Big Data?

Bigdata is a loosely defined term used to describe data sets so large and complex that they become awkward to work with using hands on database-management tools or computer software tools as they lack in parallelization capabilities. The data is too big, moves too fast, or does not fir the structures of the database architectures. Big data sizes are a constantly moving target, as of 2012 ranging from a few dozen terabytes to many petabytes in a single dataset. What is considered “big data” varies depending on the capabilities of the organization managing the set. For some organizations, facing hundreds of gigabytes of data for the first time may trigger a need to reconsider data management options. For others, it may take tens or hundreds of terabytes before data

size becomes a significant consideration. With this difficulty, a new platform of big data tools, which uses an alternative to the conventional method, has arisen to handle sensemaking over large quantities of data, as in the Apache HadoopBig Data Platform.

Big Data can be described as:

1. ***Velocity*** – how fast the data is coming in and how fast the data is must be processed to meet demand.For example, analyze 500 million daily call detail records in real time to predict customer churn faster. Reacting quickly enough to deal with velocity is a challenge to most organizations.

2. ***Variety*** – all types of data are now being captured for analysis (structured, semi-structured and unstructured) such as text, sensor data, audio, video, click streams, log files etc. By some estimates, 80 percent of an organization's data is not numeric! But it still must be included in analysis and decision making.

3. ***Volume*** – Many factors contribute to the increase in data volume – transactions-based data through the year, text data constantly streaming in from social media, increasing amount of sensor data being collected, etc.In the past, excessive data volume created a storage issue but with today's decreasing storage costs, other issues emerge, including how to determine relevance amidst the large volumes of data and how to create value from data that is relevant.

4. ***Complexity*** – involves everything from moving operational data into big data platforms and the difficulty in managing the data in multiple sites and geographies.

1.2 How much data are we talking about?

A few examples: Google grew from processing 100 TB of data a day with MapReduce in 2004 to processing 20 PB a day with MapReduce in 2008. In April 2009, a blog post was written about eBay's two enormous data warehouses: one with 2 petabytes of user data, and the other with 6.5 petabytes of user data spanning 170 trillion records and growing by 150 billion new records per day. Shortly thereafter, Facebook revealed similarly impressive numbers, boasting of 2.5 petabytes of user data, growing at about 15 terabytes

per day. Petabyte datasets are rapidly becoming the norm, and the trends are clear: our ability to store data is fast overwhelming our ability to process what we store.

Back in the year 2000 Google had a problem of adjusting the internet over every couple of days. They needed to process the data and there was no way to build an index over the whole index in a reasonable amount of time using the commercially available tools. They designed and build this infrastructure called MapReduce to handle the huge amounts of data that were not only generated by humans but also machines.

Simply put, data becomes "big data" when it basically outgrows our current ability to process it, store it, and cope with it efficiently. Storage has become very cheap in the past decade, which means it has become easy to collect mountains of data. However, our ability to actually process the mountains of data quickly has not scaled as fast. Traditional tools to analyze and store data -- SQL databases, spreadsheets were not designed to deal with vast data problems.

1.3 Solving the Big Data Issues

The fact that the existing set of data is already very large and that data is growing at an exponential rate, not only from humans but also machines and satellites generated lead us to the question of how we can tackle this issue of handling such huge amounts of data. Two factors are needed to be addressed here:

- Fast data processing i.e. search and analysis
- Reliable data storage that guarantees no data will be lost i.e. a reliable backup storage

Since the old data handling concept of using Relational Databases have proved to be inadequate in speed and backup, distinct approaches that run counter to traditional approaches of computing was needed to be adopted. MapReduce (which will be discussed further in detail later in this report) is one such approach which harnesses the power of parallel data processing and maximum CPU utilization.

To address the reliability of data storage, the concept of scaling ‘out’ and not ‘up’ is used. A large number of commodity low-end servers (i.e., the scaling ‘out’ approach) is preferred over a small number of high-end servers (i.e., the scaling ‘up’ approach). The

latter approach of purchasing symmetric multi-processing (SMP) machines with a large number of processor sockets is not cost effective, since the costs of such machines do not scale linearly (i.e., a machine with twice as many processors is often significantly more than twice as expensive).

Chapter 2

Background

2.1 Apache Hadoop

Apache Hadoop is an open source software framework that supports data-intensive distributed applications, licensed under the Apache v2 license. Hadoop was created by Doug Cutting, the creator of Apache Lucene, the widely used text search library. Hadoop was invented by Google and wasderived from Google'sMapReduce and Google File System (GFS).

In a nutshell: Hadoop provides a reliable shared storage and an analysis system.

1. The shared storage is the large scale distributed file system called the HadoopDistributed File System (HDFS) which is a disk on every file server with a software infrastructure to spread the data out among all of them.

2. The analysis of the system is performed by MapReducewhich is an algorithm that enables analysis to be run in parallel across several or all of those servers on extremely large datasets.

Although Hadoop is best known for MapReduce and its distributed file system (HDFS) the other subprojects provide complementary services, or build on the core to add higher level abstractions. The subprojects, and where they sit in the technology stack, are shown in Figure 1.1.

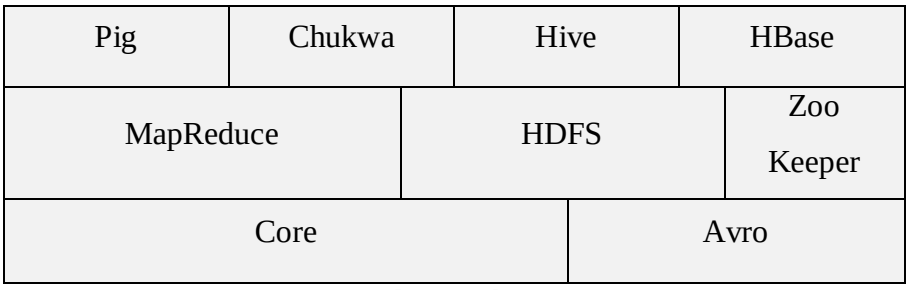


Fig: 1.1 Hadoop Subprojects

In April 2008, Hadoop broke a world record to become the fastest system to sort a terabyte of data. Running on a 910-node cluster, Hadoop sorted one terabyte in 209 seconds (just under 3.5mins), beating the previous year's winner of 297 seconds. In November of the same year, Google reported that its MapReduce implementation sorted one terabyte in 68 seconds. In May 2009, it was announced that a team at Yahoo! used Hadoop to sort one terabyte in 62 seconds.

2.2 The HDFS Architecture

HDFS stores file system metadata and application data separately. The metadata or file definitions are stored on a dedicated server called the ***NameNodes*** and the application data are stored on servers called the***DataNodes***. All servers are fully connected and communicate with each other using TCP-based protocols.

The Namenodes are independent and don't require coordination with each other. The DataNodes in HDFS do not use data protection mechanisms such as RAID to make the data durable. Instead, like GFS, the file content is replicated on multiple DataNodes for reliability. While ensuring data durability, this strategy has the added advantage that data transfer bandwidth is multiplied, and there are more opportunities for locating computation near the needed data.

2.2.1 The NameNode

The HDFS namespace is a hierarchy of files and directories. Files and directories are represented on the NameNode by *inodes*, which record attributes like permissions, modification and access times, namespace and disk space quotas. The file content is split into large blocks (typically 128 megabytes, but user selectable file-by-file) and each block of the file is independently replicated at multiple DataNodes (typically three, but user selectable file-by-file). The NameNode maintains the namespace tree and the mapping of file blocks to DataNodes (the physical location of file data).

An HDFS client wanting to read a file first contacts the NameNode for the locations of data blocks comprising the file and then reads block contents from the DataNode closest to the client. When writing data, the client requests the NameNode to nominate a suite of three DataNodes to host the block replicas. The client then writes data to the DataNodes

in a pipeline fashion. The current design has a single NameNode for each cluster. The cluster can have thousands of DataNodes and tens of thousands of HDFS clients per cluster, as each DataNode may execute multiple application tasks concurrently.

HDFS keeps the entire namespace in RAM. The inode data and the list of blocks belonging to each file comprise the metadata of the name system called the *image*. The persistent record of the image stored in the local host's native files system is called a *checkpoint*. The NameNode also stores the modification log of the image called the *journal* in the local host's native file system. For improved durability, redundant copies of the checkpoint and journal can be made at other servers. During restarts the NameNode restores the namespace by reading the namespace and replaying the journal. The locations of block replicas may change over time and are not part of the persistent checkpoint.

2.2.2 The DataNode

Each block replica on a DataNode is represented by two files in the local host's native file system. The first file contains the data itself and the second file is block's metadata including checksums for the block data and the block's *generation stamp*.

During startup each DataNode connects to the NameNode and performs a *handshake*. The purpose of the handshake is to verify the *namespace ID* and the *software version* of the DataNode. If either does not match that of the NameNode the DataNode automatically shuts down.

The namespace ID is assigned to the file system instance when it is formatted. The namespace ID is persistently stored on all nodes of the cluster. Nodes with a different namespace ID will not be able to join the cluster, thus preserving the integrity of the file system. A DataNode that is newly initialized and without any namespace ID is permitted to join the cluster and receive the cluster's namespace ID.

After the handshake the DataNode*registers* with the NameNode. DataNodes persistently store their unique *storageIDs*. The storage ID is an internal identifier of the DataNode, which makes it recognizable even if it is restarted with a different IP address or port. The storage ID is assigned to the DataNode when it registers with the NameNode for the first time and never changes after that.

A DataNode identifies block replicas in its possession to the NameNode by sending a *block report*. A block report contains the *block id*, the *generation stamp* and the length for each block replica the server hosts. The first block report is sent immediately after the

DataNode registration. Subsequent block reports are sent every hour and provide the NameNode with an up-to date view of where block replicas are located on the cluster.
During normal operation DataNodes send *heartbeats* to the NameNode to confirm that the DataNode is operating and the block replicas it hosts are available. The default heartbeat interval is three seconds. If the NameNode does not receive a heartbeat from a DataNode in ten minutes the NameNode considers the DataNode to be out of service and the block replicas hosted by that DataNode to be unavailable. The NameNode then schedules creation of new replicas of those blocks on other DataNodes. Heartbeats from a DataNode also carry information that are used for the NameNode's space allocation and load balancing decisions.
The NameNode does not directly call DataNodes. It uses replies to heartbeats to send instructions to the DataNodes. The instructions include commands to:

- replicate blocks to other nodes;
- remove local block replicas;
- re-register or to shut down the node;
- send an immediate block report.

These commands are important for maintaining the overall system integrity and therefore it is critical to keep heartbeats frequent even on big clusters. The NameNode can process thousands of heartbeats per second without affecting other NameNode operations.

2.3MapReduce: A counter traditional approach to handling Big Data

MapReduce is a programming model for expressing distributed computations on massive amounts of data and an execution framework for large-scale data processing on clusters of commodity servers. It was originally developed by Google and built on well-known principles in parallel and distributed processing. MapReduce has since enjoyed widespread adoption via an open-source implementation called Hadoop, whose development was led by Yahoo. Google uses MapReduce to continuously improve existing algorithms and to devise new algorithms for ad selection and placement.

The only feasible approach to tackling large-data problems today is to divide and conquer a fundamental concept in computer science. The basic idea is to partition a large problem

into smaller sub-problems to the extent that the sub-problems are independent so that they can be tackled in parallel by different workers, threads in a processor core, cores in a multi-core processor, multiple processors in a machine or many machines in a cluster. Intermediateresults from each individual worker are then combined to yield the final output.

The general principles behind divide-and-conquer algorithms are broadly applicable to a wide range of problems in many different application domains. However, the details of their implementations are varied and complex. The following diagram illustrates in a simple manner the basic concept of MapReduce.

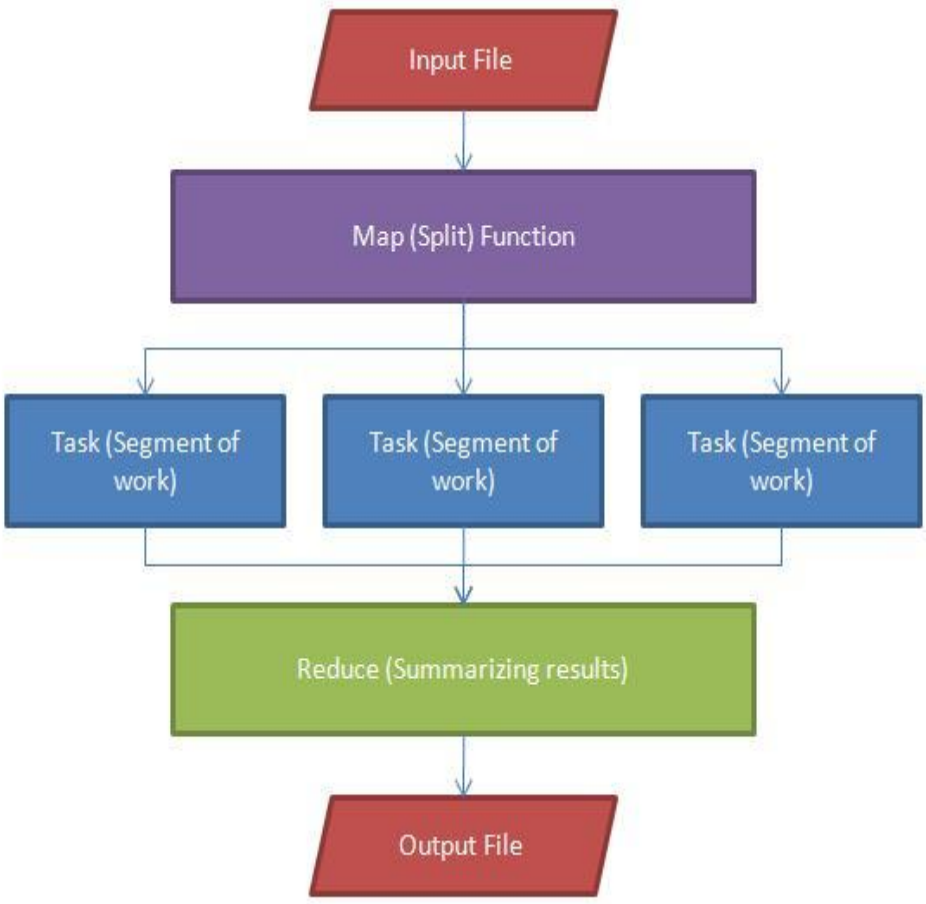


Figure 2.1The basic concept of MapReduce.

MapReduce builds on the observation that many tasks have the same structure: a computation is applied over a large number of records (e.g., documents) to generate partial results, which are then aggregated in some fashion. Naturally, the per-record computation and aggregation vary by task, but the basic structure remains fixed.

MapReduce provides an abstraction that involves the programmer defining a "mapper" and a"reducer", with the following signatures:

map: (k1, v1) →[(k2, v2)]

reduce: (k2, [v2]) →[(k3, v3)]

Key/value pairs form the basic data structure in MapReduce. The "mapper" is applied to every input key/value pair to generate an arbitrary number of intermediate key/value pairs. The "reducer" is applied to all values associated with the same intermediate key to generate output key/value pairs.

2.3.1 The Java Mapper and Reducer Class

The Java Mapper class for Wordcount:

```
public static class MyMap extends MapReduceBase implements Mapper
      <LongWritable, Text, Text, IntWritable> {

      private final static IntWritable one = new IntWritable(1);
      private Text word = new Text();

      public void map(LongWritablekey,Text value,
            OutputCollector<Text,IntWritable> output, Reporter
            reporter) throws IOException {

            String line = value.toString();
            StringTokenizertokenizer = new StringTokenizer(line);

            while(tokenizer.hasMoreTokens()) {
                  word.set(tokenizer.nextToken());
            }
            output.collect(word, one);
      }
  }
```

The Java Reducer class for Wordcount:

```
public static class MyReduce extends MapReduceBase implements
      Reducer<Text, IntWritable, Text, IntWritable> {

      public void reduce(Text key, Iterator<IntWritable> values,
            OutputCollector<Text, IntWritable> output, Reporter
            reporter) throwsIOException {

            int sum = 0;
            while (values.hasNext()) {
                  sum += values.next().get();

            }
            output.collect(key, new IntWritable(sum));
      }
 }
```

2.3.2 RDBMS vsMapReduce

Why can't we use databases with lots of disks to do large scale batch analysis?Why is MapReduce needed?

The answer to these questions comes from another trend in disk drives: seek time is improving more slowly than transfer rate. Seeking is the process of moving the disk's head to a particular place on the disk to read or write the data. It characterizes the latency of a disk operation, whereas the transfer rate corresponds to a disk's bandwidth.

If the data access pattern is dominated by seeks, it will take longer to read or write large portions of the datasets than streaming through it, which operates at the transfer rate. On the other hand, for updating a small portion of records in a database, a traditional B-Tree (the data structure used in relational databases, which is limited by the rate it can performseeks) works well. For updating the majority of a database, a B-tree is less efficient than MapReduce, which uses Sort/Merge to rebuild the database. Table 2.1 gives a comparison between RDBMS and MapReduce.

	Traditional RDBMS	**MapReduce**
Data Size	Gigabytes	Petabytes/Exabytes
Access	Interactive and batch	Batch
Updates	Read and Write many times	Write once, Read many times
Structure	Static Schema	Dynamic Schema
Integrity	High	Low
Scaling	Nonlinear	Linear

Table 2.1 RDBMS compared to MapReduce

A difference between MapReduce and an RDBMS is the amount of structure in the datasets that they operate on. Structured data is data that is organized into entities that have a defined format, such as XML documents or database tables that conform to a particular predefines schema. Semi-structured data, on the other hand, is looser, and though there may be a schema, it is often ignored, so it is often ignored, so it may be used as guide to the structure of the data: for example, a spreadsheet, in which the structure is

the grid of the cells, although the cells themselves may hold any form of data. Unstructured data does not have any particular internal structure: for example plain text or image data. MapReduce works well on unstructured or semi-structured data, since it is designed to interpret the data at processing time.

Chapter 3

Hadoop Setup

3.1 Platforms and Required Software

- Operating System : Ubuntu 12.04 (32bit)
- Virtual Box (Virtual Machine Software)
- JDK v6.0.11 (32 bit)

3.2 Getting Started

- Download and install Virtual Machine
- Download the image file (.iso) for Ubuntu 12.04 (32bit)
- Create a new virtual machine
- Specify “Name”, “OS” and “Version”
- Allocate memory and create new hard disk
- Assign file type (VDI)
- Dynamically allocate storage
- Specify location and virtual disk size (at least 30GB is preferred)
- Install OS

3.3 Network Configuration

- Attach to bridged adapter in Network Settings
- Allow VMs in Promiscuous Mode
- Set IPv4 settings to manual
- Add new connection with proper IP address, subnet mask, gateway and DNS
- Set newly made connection as the default

3.4 Enabling Internet

- Set Network Proxy method to manual
- Assign proxy and port
- Apply settings system wide
- Assign DNS servers
- Restart connection

3.5 Installing JDK

1. Enter terminal and provide the following commands-
   - chmoda+x ./jdk-6u11
   - sudo ./jdk
   - sudo mv ./jdk.6.0_11 /usr/lib/jvm/
   - cd /usr/lib/jvm
   - sudogedit ~/.bashrc
   - export JAVA_HOME=/usr/lib/jvm/jdk1.6.0_11
   - export PATH =${PATH}:${JAVA_HOME}/bin
2. Restart terminal

3.6 Installing Hadoop

1. Download a stable release of Hadoop from the Apache Download Mirrors
2. Extract the contents to a desired location-
   - $ cd /usr/local
   - $ sudo tar xzf hadoop-1.0.3.tar.gz
   - $ sudo mv hadoop-1.0.3 hadoop
   - $ sudochown -R hduser:hadoophadoop
3. Add the following lines to the end of the **$HOME/.bashrc** file of user *hduser-*
   - export HADOOP_HOME=/usr/local/hadoop
   - export JAVA_HOME=/usr/lib/jvm/java-6-sun

- export PATH=$PATH:$HADOOP_HOME/bin

4. Set the JAVA_HOME environment variable to the Sun JDK/JRE 6 directory-
   - export JAVA_HOME=/usr/lib/jvm/java-6-sun

3.7 Setting up a Standalone Node Cluster:

3.8 Setting up a Single Node Cluster:

1. Create a dedicated user account to run Hadoop in-
   - $ sudoaddgrouphadoop
   - $ sudoadduser --ingrouphadoophduser
2. Generate an SSH key for the *hduser* user-
   - user@ubuntu:~$ su - hduser
   - hduser@ubuntu:~$ ssh-keygen -t rsa -P ""
3. Enable SSH access to local machine with the newly created key-
   - hduser@ubuntu:~$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
4. Test the SSH setup by connecting to the local machine with the *hduser* user-
   - hduser@ubuntu:~$ sshlocalhost
5. Modify the file *conf/core-site.xml*-

```
<!-- In: conf/core-site.xml -->
<property>
<name>hadoop.tmp.dir</name>
<value>/app/hadoop/tmp</value>
<description>A base for other temporary directories.</description>
</property>

<property>
<name>fs.default.name</name>
<value>hdfs://localhost:54310</value>
<description>The name of the default file system.  A URI whose
scheme and authority determine the FileSystem implementation.  The
uri's scheme determines the config property (fs.SCHEME.impl) naming
```

```
the FileSystem implementation class.  The uri's authority is used to
determine the host, port, etc. for a filesystem.</description>
</property>
```

6. Modify the file *conf/mapred-site.xml*-

```
<!-- In: conf/mapred-site.xml -->
<property>
<name>mapred.job.tracker</name>
<value>localhost:54311</value>
<description>The host and port that the MapReduce job tracker runs
at.  If "local", then jobs are run in-process as a single map
and reduce task.
</description>
</property>
```

7. Modify the file *conf/hdfs-site.xml*-

```
<!-- In: conf/hdfs-site.xml -->
<property>
<name>dfs.replication</name>
<value>1</value>
<description>Default block replication.
 The actual number of replications can be specified when the file is created.
 The default is used if replication is not specified in create time.
</description>
</property>
```

8. Format the HDFS filesystem via the NameNode-
    - hduser@ubuntu:~$ /usr/local/hadoop/bin/hadoop namenode –format
9. Start the single-node cluster-
    - hduser@ubuntu:~$ /usr/local/hadoop/bin/start-all.sh

3.9 Setting up a Multi Node Cluster:

1. Define Master and Slave in /etc/hosts file-
    - <IP Address> master
    - <IP Address> slave
2. Modify conf/slaves in the master to add slaves-
    - master
    - slave
3. Modify the file *conf/core-site.xml*-

```
<!-- In: conf/core-site.xml -->
<property>
<name>fs.default.name</name>
<value>hdfs://master:54310</value>
<description>The name of the default file system.  A URI whose
scheme and authority determine the FileSystem implementation.  The
uri's scheme determines the config property (fs.SCHEME.impl) naming
the FileSystem implementation class.  The uri's authority is used to
determine the host, port, etc. for a filesystem.</description>
</property>
```

4. Modify the file *conf/mapred-site.xml*-

```
<!-- In: conf/mapred-site.xml -->
<property>
<name>mapred.job.tracker</name>
<value>master:54311</value>
<description>The host and port that the MapReduce job tracker runs
at.  If "local", then jobs are run in-process as a single map
and reduce task.
</description>
</property>
```

5. Modify the file *conf/hdfs-site.xml*-

```
<!-- In: conf/hdfs-site.xml -->
<property>
<name>dfs.replication</name>
<value>2</value>
<description>Default block replication.
 The actual number of replications can be specified when the file is created.
 The default is used if replication is not specified in create time.
</description>
</property>
```

6. Format the HDFS filesystem via the NameNode-
   - hduser@ubuntu:~$ /usr/local/hadoop/bin/hadoop namenode –format
7. Start the single-node cluster-
   - hduser@ubuntu:~$ /usr/local/hadoop/bin/start-all.sh

Chapter 4

N-Gram Analysis

4.1 What are n-grams?

An ***n*-gram** is a contiguous sequence of *n* items from a given sequence of text or speech. An *n*-gram could be any combination of letters. However, the items in question can be phonemes, syllables, letters, words or base pairs according to the application. The *n*-grams typically are collected from a text or speech corpus.

An *n*-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram" (or, less commonly, a "digram"); size 3 is a "trigram". Larger sizes are sometimes referred to by the value of *n*, e.g., "four-gram", "five-gram", and so on.

Field	Unit	Sample Sequence	1-gram	2-gram	3-gram
			unigram	bigram	trigram
Computational Linguistics	character	"for predicting the next item"	f, o, r, p,….	fo, or, rp, pr,…	for, orp, ror, pre,…
Computational Linguistics	word	"for predicting the next item"	for, predicting, the,…..	For predicting, predicting the, the next,…	For predicting the, predicting the next, the next item.

Table 4.1: N-Grams

4.2 Application of N-Grams

An ***n*-gram model** is a type of probabilistic language model for predicting the next item in a sequence. *n*-gram models are now widely used in probability, communication theory, computational linguistics (for instance, statistical natural language processing), computational biology (for instance, biological sequence analysis), and data compression.

The two core advantages of *n*-gram models (and algorithms that use them) are relative simplicity and the ability to scale up – by simply increasing *n*. *n*-gram models are widely used in statistical natural language processing. In speech recognition, phonemes and sequences of phonemes are modeled using a*n*-gram distribution.

4.3 N-Gram using MapReduce

Since our interest was to generate n-grams from large document collections, using the MapReduce algorithms, which harness the power of parallel computation, is an efficient way to generate n-gram (unigram, bigram, trigram) frequencies.The following diagram illustrates how n-grams and each of their frequency of occurrence was generated.

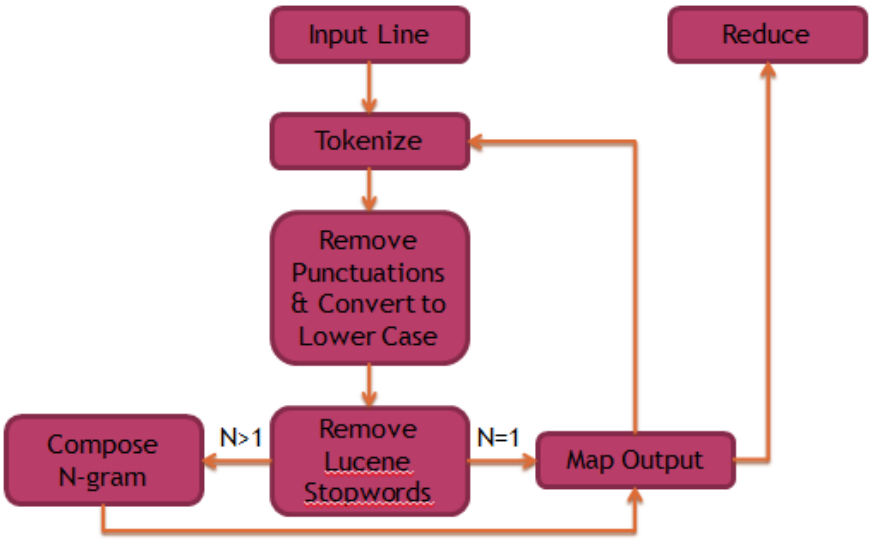


Figure 4.1 Flow Diagram of Generating N-Grams using MapReduce

4.4 Input and Output Files

Input (English):

To read English text, we only admit tokens formed fromalphanumeric characters. We remove all punctuations from the text and convert these words into lowercase. After this we remove all stopwords from the Lucene Stopword List. Stopwords are common words

that carry less important meaning than keywords. This is done because the Stopwords have high frequency of occurrence and high overlaps in Stopword frequency do not necessarily indicate similarity between two texts.

The Stopwords we have removed for our purpose are the following: “a”, “an”, “and”, “are”, “as”, “at”, “be”, “but”, “by”, “for”, “if”, “in”, “into”, “is”, “it”, “no”, “not”, “of”, “on”, “or”, “such”, “that”, “the”, “there”, “their”, “then”, “these”, “they”, “this”, “to”, “was”, “will”, “with”.

A section of Input File:

Data is growing fast. About ninety percent of the world’s existing data was created in the last year. The amount of digital content on the web is now close to five hundred billion gigabytes and the number is expected to double within a year. The explosion of mobile networks, cloud computing and new technologies has given rise to incomprehensibly large worlds of information. The necessity to manage efficiently the exponentially growing dataset is increasing everyday. Data not only needs to be processed and analyzed fast, a guarantee of a reliable backup such that there is no data loss, also needs to be provided.

A section of the Processed file:

data growing fast about ninety percent world’s existing data created last year amount digital content web now close five hundred billion gigabytes number expected double within year explosion mobile networks cloud computing new technologies has given rise incomprehensibly large worlds information necessity manage efficiently exponentially growing dataset increasing everyday data only needs processed analyzed fast guarantee reliable backup data loss also needs provided

A section of the trigram output file:

about ninety percent	1
amount digital content	1
created last year	1
data created last	1
data growing fast	1
existing data created	1
fast about ninety	1
growing fast about	1
last year amount	1
ninety percent world's	1
percent world's existing	1
world's existing data	1
year amount digital	1
…	

Input (Bangla):

As we have not found any work related to Bangla document comparison, we have created a small set of stopwords in order to observe how our algorithm works on Bangla documents. This set can be extended and improved depending on the user's needs.

We have used the following set of stopwords for our work:

"ও", "এবং", "বরং", "সুতরাং", "এ", "এই", "অধিকন্তু", "যে", "জন্য", "নিমিত্তে", "হয়", "আছে", "ছিল", "ছিলেন", "আর", "তো", "যেন", "জন্য", "যে", "কি", "যদি"

A section of Input file (Bangla):

এক চাকার সাইকেলে আঠারো কিলোমিটার ,৮৯ সাল থেকে প্রতিবছর ডেনমার্কের শিশুদের করা নানা অবিশ্বাস্য কাণ্ড নিয়ে ছাপা হয় একটি বই। শূন্য থেকে ১৮ বছরবয়সী শিশুরাই এই রেকর্ড বইয়ের নায়ক-নায়িকা।
এ বছর এক চাকার সাইকেল চালিয়ে সেই বইয়ে নিজের নামটা উঠিয়েছে ক্রিস্টিনা। তার সেই কীর্তির কথাই জানাচ্ছেনজাহীদ রেজা নূর ছবিটা দেখো। এক চাকার সাইকেলে দিব্যি চলেছে মেয়েটা। ওর নাম ক্রিস্টিনা নিলসেন।

A section of trigram outputfile (Bangla):

অবিশ্বাস্য কাণ্ড নিয়ে	1
আঠার কিলোমিটার ৮৯	1
উঠিয়েছ ক্রিস্টিন তার	1
এক চাকার সাইকেল	3
একট বই শূন্য	1
ওর নাম ক্রিস্টিন	1
কথাই জানাচ্ছেন জাহীদ	1
কর নান অবিশ্বাস্য	1
কাণ্ড নিয় ছাপ	1
কিলোমিটার ৮৯ সাল	1
কীর্তির কথাই জানাচ্ছেন	1
ক্রিস্টিন তার সেই	1
চলেছ মেয়েট ওর	1
চাকার সাইকেল আঠার	1

Chapter 5

Document Similarity Detection

5.1 TextSimilarityDetectionMethods

There are multiple text similarity methods. However, not all of them can be implemented using MapReduce algorithms. For our work, we have mainly focused on detecting syntactic similarity between pairs of documents in large document collection. The following two models focus mainly on syntactic similarity.

- Ferret model which looks for lexical matches. Each text is converted to a set of 3 word trigrams, and matching trigrams are found. The comparison is based on the Jaccard coefficient value.

- String matching model, which counts those words or word sequences that occur in bothtexts (omitting stop words) and calculates the Jaccard coefficient value to determine thesimilarity between 2 texts. A feature string may contain one or more words.

Our work mainly deals with the Ferret Model which is explained in detail below.

5.2 The Ferret Model

The Ferret Model provides very fast and fine-grained similarity detection in moderately large collections of documents by using triples of tokens or commonly known as trigrams to compute a measure of resemblance between each pair of documents in the collection. The earlier version of Ferret has been used primarily to detect collusion in students' work; and is described in from a pedagogic viewpoint.

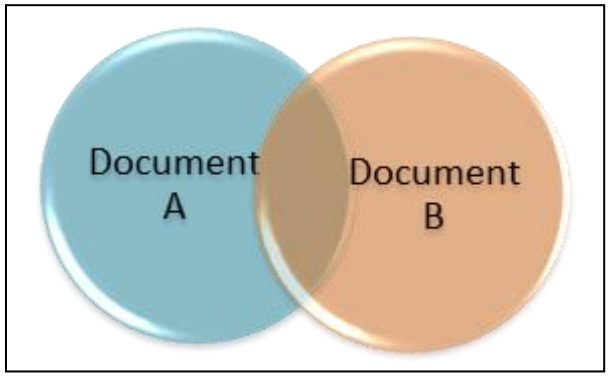
Ferret is essentially an algorithm to compute the similarity of two documents, based oncounting the number of distinct triples of tokens common between documents. This countis used to compute a resemblance measure.

5.3 The Jaccard Coefficient and the Jaccard Distance

The Jaccard index, also known as the Jaccard similarity coefficient (by Paul Jaccard), is a statistic used for comparing the similarity and diversity of sample sets.

The Jaccard coefficient, denoted by *J(A,B),* measures similarity between sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets.

The Jaccard distance, denoted by $J_\delta(A,B)$,which measures *dis*similarity between sample sets, is complementary to the Jaccard coefficient and is obtained by subtracting the Jaccard coefficient from 1, or, equivalently, by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union.



$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

$$J_\delta(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}.$$

Fig. 5.1 The Jaccard Coefficient and Distance

Chapter 6

Implementation

6.1 The N-Gram Analysis Class (for English)

```
Packageorg.myorg;
importjava.io.IOException;
importjava.util.*;
importorg.apache.hadoop.fs.Path;
importorg.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
importorg.apache.hadoop.mapred.*;
importorg.apache.hadoop.util.*;


public class MyNGram {

      public static String str = "";
      public static long wordCounter = 0;
      public static intnumWords=0;
      public static intoutputCounter=1;
      public static Text tx;


//=============================== MAP ========================================

      public static class MyMap extends MapReduceBase implements Mapper
      <LongWritable, Text, Text, IntWritable> {

            private final static IntWritable one = new IntWritable(1);
            private Text word = new Text();

            public void map(LongWritablekey,Text value,
            OutputCollector<Text,IntWritable> output, Reporterreporter)
            throws IOException {

                  String line = value.toString();
                  StringTokenizertokenizer = new StringTokenizer(line);
                  String wd="";

                  while(tokenizer.hasMoreTokens()) {
                        wd=tokenizer.nextToken();  //get a word
                        wd=removePunctuation(wd);  //remove punctuation
                        wd=removeStopWord(wd);   //return "" if stopword

                        if (wd.length()!=0){       //if not blank
                        if (numWords==1){   //for word count only
                              word.set(wd);
```

```
                              output.collect(word, one);
                        }else if (numWords>1){
                              //Compose Multiple Words
                              wd = composeWord(wd,numWords);
                              if (wd.length()!= 0) {
                              word.set(wd);
                              output.collect(word, one);
                              }
                           }

                  }
                        //output.collect(word, one);
                  }
            }
      }
//================================ REDUCE ================================

      public static class MyReduce extends MapReduceBase implements
      Reducer<Text, IntWritable, Text, IntWritable> {

            public void reduce(Text key, Iterator<IntWritable> values,
            OutputCollector<Text, IntWritable> output, Reporter
            reporter) throwsIOException {
                  int sum = 0;
                  while (values.hasNext()) {
                        sum += values.next().get();
                  }
            //convert int 'outputCounter' to Text Object and add 'key'
                  tx=new Text(outputCounter+" "+key);
                        output.collect(tx, new IntWritable(sum));
                  outputCounter++;
            }
      }
//================================ MAIN ================================

      public static void main(String[]args) throws Exception {

            Scanner sc=new Scanner (System.in);
            System.out.print ("Enter Value of N ");
            numWords=sc.nextInt();

            JobConfconf = new JobConf(MyNGram.class);
            conf.setJobName("MyNGram");
            conf.setOutputKeyClass(Text.class);
            conf.setOutputValueClass(IntWritable.class);
            conf.setMapperClass(MyMap.class);
            conf.setCombinerClass(MyReduce.class);
            conf.setReducerClass(MyReduce.class);
            conf.setInputFormat(TextInputFormat.class);
            conf.setOutputFormat(TextOutputFormat.class);
            FileInputFormat.setInputPaths(conf, new Path(args[0]));
            FileOutputFormat.setOutputPath(conf, new Path(args[1]));
            JobClient.runJob(conf);
```

```
	}

//============================ STOP WORDS ============================

	public static String removeStopWord(String str){
		String[] ENGLISH_STOP_WORDS = {
		"a", "an", "and", "are", "as", "at", "be", "but", "by",
		"for", "if", "in", "into", "is", "it", "no", "not", "of",
		"on", "or", "such", "that", "the", "their", "then", "there",
		"these", "they", "this", "to", "was", "will", "with"
		};
		for (int i=0; i<ENGLISH_STOP_WORDS.length; i++){
			if (str.equals(ENGLISH_STOP_WORDS[i])){
				str="";
			}
		}
		returnstr;
	}

//============================ PUNCTUATIONS ============================

	public static String removePunctuation(String str){

		intlen=0;
		int i=0;
		charlastChar=0;
		String Ch="ABCDEFGHIJKLMNOPQRSTUVWXY
		Zabcdefghijklmnopqrstuvwxyz0123456789";
		//remove prefix
		while(i<3){
			str=str.trim();
			len=str.length();
			if (len==0) break;
			lastChar=str.charAt(0);

			if(Ch.indexOf(Character.toString(lastChar))==-1){
			str=str.substring(1,len);
			}else{
			break;
			}
		}
		//remove suffix
		while(i<3){
			str=str.trim();
			len=str.length();
			if(len==0) break;
			lastChar=str.charAt(len-1);
			if(Ch.indexOf(Character.toString(lastChar))==-1){
			str=str.substring(0,len-1);
			}else{
			break;
			}
		}
```

```
            returnstr.toLowerCase(); //return word in lowercase
      }


//==================== WORD COMPOSE for N-GRAM=======================

      public static String composeWord(String str1, int n){
            String s1 = "";
            if (wordCounter< n){
                  str = str+" "+str1;
                  wordCounter++;
            }
            if (wordCounter == n){
                  s1 = str.trim();
                  wordCounter = n-1;
                  str = s1.substring(s1.indexOf(" ",0)+1, s1.length());
            }
            return s1;
      }
}//end of MyNGram class
```

6.2 Calculating the Jaccard Coefficient and the Jaccard Distance

DOCUMENT_B	
existing data created	1
data created last	1
created last year	1
last year amount	1
year amount digital	1
amount digital content	1
digital content web	1
content web now	1
web now close	1
now close five	1
close five hundred	1
five hundred billion	1
hundred billion gigabytes	1
billion gigabytes number	1

14

$$\text{Jaccard}(A,B) = \frac{\text{match}(A,B)}{\text{match}(A,B) + \text{nonMatch}(A,B)*w1 + \text{nonMatch}(B,A)*w2}$$

Where w1 = no. of trigrams in A but not in B as a fraction of total freq.

w2 = no. of trigrams in B but not in A as a fraction of total freq.

Jaccard Coefficient & Distance

$$J(A,B) = \frac{6}{6 + 7 * (7/13) + 8 * (8/14)} = 0.418$$

$$J_\delta(A,B) = 1 - 0.418 = 0.582$$

6.3 The N-Gram Analysis Class (for Bangla)

The part of the N-Gram Analysis Class for Bangla that differs from the one for English is given below:

```
//============================ STOP WORDS ======================

public static String removeStopWord(String str){

      String[] ENGLISH_STOP_WORDS = {

            "এ", "এই", "আর", "তো", "যেন",  "জন্য", "যে", "ও", "এবং",
             "বরং", "সুতরাং", "অধিকন্তু", "নিমিত্তে", "হয়", "আছে", "ছিল",
             "ছিলেন" , "যে", "কি", "যদি"

             };

      for (int i=0; i<ENGLISH_STOP_WORDS.length; i++){

      if (str.equals(ENGLISH_STOP_WORDS[i])){

            str="";

      }

      }

      return str;

}
```

```
//============================ PUNCTUATIONS ============================

public static String removePunctuation(String str){
	int len=0;
	int i=0;
	char lastChar=0;

	String
	Ch="অআইঈউঊঋএঐওঔািীুূৃেৈোৌকখগঘঙচছজঝঞটঠডঢণতথদধনপফবভমযরল
	শষসহড়ঢ়য়ৎংঃ১২৩৪৫৬৭৮৯০";
	//remove prefix
	while(i<3){
		str=str.trim();
		len=str.length();
		if (len==0) break;
l		astChar=str.charAt(0);
		if(Ch.indexOf(Character.toString(lastChar))==-1){
		str=str.substring(1,len);
		}else{
		break;
		}
	}
	//remove suffix
	while(i<3){
		str=str.trim();
		len=str.length();
		if(len==0) break;
		lastChar=str.charAt(len-1);
		if(Ch.indexOf(Character.toString(lastChar))==-1){
		str=str.substring(0,len-1);
		}else{
		break;
		}
	}
	return str.toLowerCase(); //return word in lowercase
  }
```

Chapter 7

System Evaluation

7.1 PerformanceComparison between Different Modes of Operation

Table 7.1 shows the number of tokens generated for N –Gram Analysis where N=1, N 2 and N=3 for various file sizes:

N-Gram Analysis	Number of Token				
	10.1MB	21.2MB	31.3MB	40MB	50MB
N=1	69,347	106,133	145,246	174,658	210,128
N=2	800,921	1,334,591	2,024,822	2,591,408	3,220,503
N=3	1,153,566	2,247,198	3,377,150	4,335,267	5,412,955

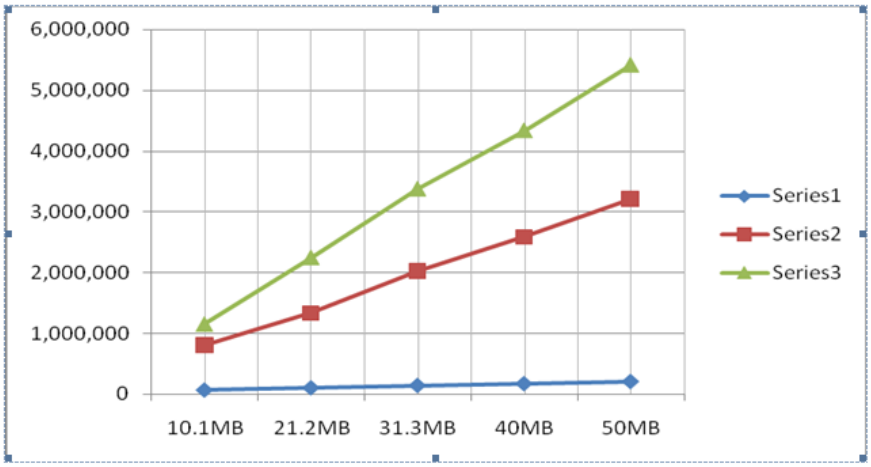


Figure 7.1

7.1.1. Standalone Mode:

The Standalone Mode performance analysis was done in a personal computer with the following specifications:

- Intel® Core i3 CPU
- 2.53 Ghz Processor Speed
- 2 GB System Memory

The analysis yields the following results described in Table 7.2:

N-Gram Analysis	Time in Second				
	10.1MB	21.2MB	31.3MB	40MB	50MB
N=1	12	21	30	42	66
N=2	15	27	39	51	78
N=3	18	33	45	60	85

Figure: 7.2

7.1.2. Cluster Mode

The Standalone Mode performance analysis was done in five personal computers with the following specifications:

- Intel® Core i3 CPU
- 2.53 Ghz Processor Speed
- 2 GB System Memory

The analysis yields the following results described in Table 7.3:

N-Gram Analysis	Time in Second				
	10.1MB	21.2MB	31.3MB	40MB	50MB
N=1	30	31	36	36	31
N=2	30	30	31	31	36
N=3	31	30	31	36	36

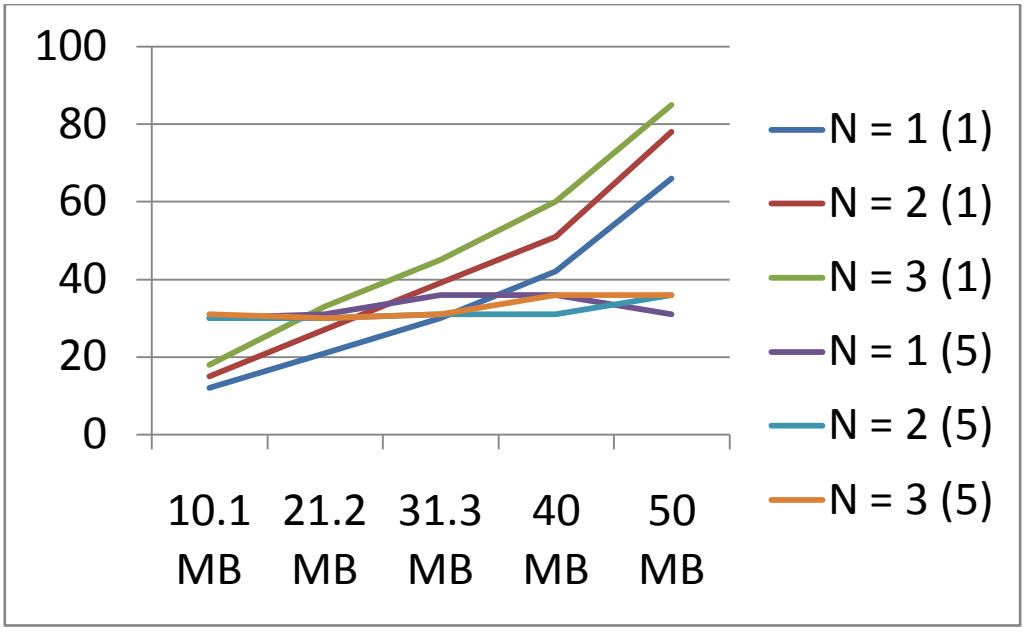


Figure 7.3

7.2 Future Works

Research to be done on Bangla stopwords:

Document similarity comparison in Bangla arouses a lot of challenges are absent when dealing with English corpuses. The Stopwords in Bangla language are much more complex than the ones in English language. Hence, we intend to enrich our set of Bangla Stopwords after thorough research on the matter.

Run algorithm on bigger data set:

Due to the limitation of Bangla resource availability, we had to run our algorithm on comparatively small data sets. We intend to run it on a much broader measure (in terabyte scale) in as we acquire more resources.

Usage of more machines:

Hardware availability was always a factor for us. Upon our productive analysis, we are hoping that more hardware will be made available to us so that when we acquire terabytes of data, we can actually run our algorithm on it with greater efficiency.

Usage of high end machines:

Another hardware limitation we faced was the machines limited capabilities. We are hopeful that once we are given better dedicated machines, our results would reflect much more efficient outputs.

Chapter 8

Conclusion

We initially selected the Hadoop platform in order to analyze Big Data. However, in order to successfully analyze it, we needed an objective based on which the analysis would have to be done.

Our objective was to develop a document similarity algorithm, through which multiple documents could be compared amongst each other to identify how much similar the documents are.

Initially the algorithm was developed for the English words and was tested on an English corpus. Through carrying out various tests by varying both the nodes and the corpus size, we were able to determine how much more efficient is a multi node cluster compared to a single node one. Meaning, how fast would a job get done when the work is being split amongst multiple machines.

So far, we were successfully able to implement our document similarity detection algorithm for English language. We achieved this by having the mapper class process unprocessed corpus to produce trigrams. We were also successfully able to modify the algorithm to detect document similarities for Bangla language. However, due to lack of resources and researches done on Bangla language in this particular area, our Bangla stopword set has a lot of room to be perfected.

There are also some slight redundancies in our algorithm which we believe we can mitigate in time; resulting in even more efficient results. This means there is still some room for further coding.

In future, we hope that we will be able to run our document similarity detection algorithm on a much broader scale. Once we are granted more high-end machines to function as nodes, hopefully we will be able to process gigabytes, terabytes or even exabytes of data and produce fantastic results.

APPENDIX

Bangla Document File (Part of large corpus)

DOCUMENT 1

ড.রেজাখান: পাখিরাযারবন্ধু
ফরিদুররহমানপান্থ
সঠিকপৃষ্ঠপোষকতাপ্রদানেব্যর্থহওয়ায়বাংলাদেশযেমনহারিয়েছেমেঘনাদসাহাকেতেমনিদেশেরাখাযায়নিআবেদ চৌধুরীরমতোনামীবিজ্ঞানীদেরও। সেইতালিকাদিনেদিনেদীর্ঘহচ্ছে। আমাদেরসেবিষয়েতেমনভাবান্তরনেই। ড. আলীরেজাখানওতেমনইএকজন। যাকেসবাইচেনেনড.
রেজাখানহিসেবে। যাররয়েছেপাখিওবন্যপ্রাণীনিয়েবিস্তরজানাশোনা,
সঙ্গেপ্রাতিষ্ঠানিকশিক্ষাও। ছিলেনঢাকাবিশ্ববিদ্যালয়েরসহযোগীঅধ্যাপক। কিন্তুঅভিমানেতিনিওআরসবারমতো দেশত্যাগীহয়েছেন। প্রকৃতিপ্রেমিকওপাখিবিশেষজ্ঞএমানুষটিবর্তমানেকাজকরছেন দুবাইচিড়িয়াখানায়, পরিচালকহিসেবে। এইদেশেরএকজনগবেষকওপাখিবিশেষজ্ঞদুবাইয়েদাপটেরসঙ্গেকাজকরছেন। অথচঅনেকে রমতোতাকেওধরেরাখতেপারেনিএদেশ! ড.রেজাখানবিষয়টিকেনিয়তিবলেমেনেনিয়েছেন,
সবআমারভাগ্য। আমিআমারদেশকেভালোবাসি। দেশেএলেইখবরনিতেযাইযশোরেরহনুমানগুলোকেমনআছে, দেশেরপাখিগুলোকেমনরয়েছে। কিন্তুঢাকাবিশ্ববিদ্যালয়েশিক্ষকতারচাকরিটাকরতেপারলামনা। বলেরাখা ভালো, তিনিপড়াতেনঢাকাবিশ্ববিদ্যলয়েরপ্রাণিবিদ্যাবিভাগে। অনিয়মআরবিশৃঙ্খলামেনে
নিতেনাপেরেনিজেকেসরিয়েনিয়েছেনসেখানথেকে। এখনদুবাইয়েথাকেন, স্থায়ীভাবে। একজন প্রকৃতিপ্রেমীড.রেজাখানএকজনসাধক। পাখিনিয়েগবেষণাকরাতারশখ। বিষয়টিকেতিনিসাধনারপর্যায়ে নিয়েগেছেন। বনে-জঙ্গলেঘুরেঘুরেতিনিপাখিখোঁজেন। দেশেএলেইদেখতেযানসুন্দরবনের
পাখিগুলোকে। নিজেকেতিনিপ্রকৃতিপ্রেমিকবলেপরিচয়দেন। প্রকৃতিরসঙ্গেতারবড়ভাব।
যশোরেরহনুমানগুলোঠিকমতোখেতেপায়কিনাতানিয়েচিন্তিতথাকেনড. রেজাখান। এ
কারণেইসম্ভবতপ্রকৃতিতারআত্মারবিষয়হয়েদাঁড়িয়েছে। আরযেকারণেতিনিদেশেরসবার কাছেমিনতিকরেন, সুন্দরবনপৃথিবীতেআরকারোকাছেনেই, ওটাশুধুআমাদেরসম্পদ। দয়া করেওইসৌন্দর্যনষ্টকরবেননা।
পাখিরাযারবন্ধুরাস্তাঘাটেহাঁটারসময়পাখিদেরকিচিরমিচিরকরাশুনেইগড়গড়করেবলেদেনওইপাখিটিরপরিচয়। পাখিটিকীখায়, কোথায়থাকতেপছন্দকরে, কীভাবেবাসাকরেসবকিছুইতারনখদর্পণে!
পাখিসম্পর্কেপড়তেএবংবলতেপছন্দকরেনড. রেজাখান।

DOCUMENT 2

বলেরাখা ভালো, তিনিপড়াতেনঢাকাবিশ্ববিদ্যলয়েরপ্রাণিবিদ্যাবিভাগে। অনিয়মআরবিশৃঙ্খলামেনে
নিতেনাপেরেনিজেকেসরিয়েনিয়েছেনসেখানথেকে। এখনদুবাইয়েথাকেন, স্থায়ীভাবে। একজন প্রকৃতিপ্রেমীড.রেজাখানএকজনসাধক। পাখিনিয়েগবেষণাকরাতারশখ। বিষয়টিকেতিনিসাধনারপর্যায়ে নিয়েগেছেন। বনে-জঙ্গলেঘুরেঘুরেতিনিপাখিখোঁজেন। দেশেএলেইদেখতেযানসুন্দরবনের
পাখিগুলোকে। নিজেকেতিনিপ্রকৃতিপ্রেমিকবলেপরিচয়দেন। প্রকৃতিরসঙ্গেতারবড়ভাব।
যশোরেরহনুমানগুলোঠিকমতোখেতেপায়কিনাতানিয়েচিন্তিতথাকেনড. রেজাখান। এ
কারণেইসম্ভবতপ্রকৃতিতারআত্মারবিষয়হয়েদাঁড়িয়েছে। তারসংগ্রহেরয়েছেঅসংখ্যপাখিরছবিওতাদেরসম্পর্কেবি স্তারিততথ্য। দেশেএলেচলেযানবনে-
বাদাড়ে। হাতেক্যামেরানিয়েতুলতেশুরুকরেনছবি। পাখিরাওযেনতাকেসুযোগকরেদেয়। ছবিতোলারসময়বড়বড় চোখকরেতাকিয়েথাকেতারদিকে। ড. খানবলেন, ওরাআমারবন্ধু। আমারআত্মীয়!
ওদেরছাড়াআমিএকদিনওচলতেপারিনা। দুবাইচিড়িয়াখানায়তিনিপ্রতিনিয়তদেখভালকরেনপাখিদের। পাখিদের

খাওয়ানোথেকেশুরুকরেসেবা-
শুশ্রূষারসময়ওওদেরপাশেইথাকেনতিনি। পাখিদেরসঙ্গেতারভাবহয়েগেছে। রেজাখানওএকটিফসিলেরকথাএবছ রতিনিঢাকায়এসেছিলেন। সেসময়একটিহাতিরফসিলনিয়েগবেষণাচলছিলআমাদের। ওইগবেষণাকর্মেসহযো গিতাকরাইছিলতারসেবারেরসফরেরউদ্দেশ্য। হাতিরহাড়েরওইগবেষণাকাজেরজন্যতিনিগিয়েছিলেনভারতেও। তারবন্ধুদেরসহযোগিতানিয়েতিনিফলাফলজানিয়েছিলেনদেশবাসীকে। ড.রেজাখানসম্পর্কিতবাবা-
মাআরকাছেরমানুষরাতাকেচিনতআজাদনামে, পুরোনামড.
আলীরেজাখান। বড়হয়েছেনমানিকগঞ্জে। বাবাখানমোঃজোয়াহারআলীআরমাআয়েশাখাতুন। আরযেকারণেতি নিদেশেরসবার কাছেমিনতিকরেন, সুন্দরবনপৃথিবীতেআরকারোকাছেনেই, ওটাশুধুআমাদেরসম্পদ। দয়া করেওইসৌন্দর্যনষ্টকরবেননা।
পাখিরাযারবন্ধুরাস্তাঘাটেহাঁটারসময়পাখিদেরকিচিরমিচিরকরাশুনেইগড়গড়করেবলেদে নওইপাখিটিরপরিচয়। পাখিটিকীখায়, কোথায়থাকতেপছন্দকরে, কীভাবেবাসাকরেসবকিছুইতারনখদর্পণে!
পাখিসম্পর্কেপড়তেএবংবলতেপছন্দকরেনড. রেজাখান। সাতভাই-
বোনেরমধ্যেতিনিছিলেনছয়নম্বর। প্রথমযেস্কুলেতিনিপড়েছেনসেটিমানিকগঞ্জে,
নামবালিয়াপাঠশালা। এরপরবালিয়াফ্রিপ্রাইমারিস্কুলহয়েপড়েছেনমানিকগঞ্জসরকারিমাল্টিপারপাসহাইস্কুলে। এ রপরদেবেন্দ্রকলেজথেকেইন্টারমিডিয়েটশেষকরেনটরডেমকলেজথেকেনিয়েছেনবিএসসিডিগ্রি।

Trigram of Document 1

[1] অথচঅনেকেরমতো	1	[48] কেমনদেশেরপাখিগুলো	1
[2] অধ্যাপককিন্তুঅভিমানে	1	[49] কেমনরয়েছেকিন্তু	1
[3] অনিয়মবিশৃঙ্খলামেনে	1	[50] কোথায়থাকতেপছন্দ	1
[4] অনেকেরমতোতাকেও	1	[51] খবরনিতেযাই	1
[5] অভিমানেতিনিওসবার	1	[52] খানএকজনসাধক	1
[6] আত্মারবিষয়হয়ে	1	[53] খানকারণেইসম্ভবত	1
[7] আবেদচৌধুরীরমতো	1	[54] খানপাখিরাযার	1
[8] আমাদেরসম্পদদয়া	1	[55] খানবিষয়টিকেনিয়তি	1
[9] আমাদেরসেবিষয়ে	1	[56] খানহিসেবেযার	1
[10] আমারদেশকেভালোবাসি	1	[57] খানওতেমনইএকজন	1
[11] আমারভাগ্যআমি	1	[58] খায়কোথায়থাকতে	1
[12] আমিআমারদেশকে	1	[59] খেতেপায়না	1
[13] আলীরেজাখানও	1	[60] খোঁজেনদেশেএলেই	1
[14] একজনগবেষকপাখি	1	[61] গবেষকপাখিবিশেষজ্ঞ	1
[15] একজনপ্রকৃতিপ্রেমীড.রেজা	1	[62] গবেষণাকরাতার	1
[16] একজনযাকেসবাই	1	[63] গেছেনবনে-জঙ্গলেঘুরে	1
[17] একজনসাধকপাখি	1	[64] গড়গড়করেবলে	1
[18] এখনদুবাইয়েথাকেন	1	[65] ঘুরেঘুরেতিনি	1
[19] এলেইখবরনিতে	1	[66] ঘুরেতিনিপাখি	1
[20] এলেইদেখতেযান	1	[67] চাকরিটাকরতেপারলাম	1
[21] ওইপাখিটিরপরিচয়	1	[68] চিন্তিতথাকেনড	1
[22] ওইসৌন্দর্যনষ্ট	1	[69] চিড়িয়াখানায়পরিচালকহিসেবে	1
[23] ওটাশুধুআমাদের	1	[70] চেনেনডরেজা	1
[24] করছেনঅথচঅনেকের	1	[71] চৌধুরীরমতোনামী	1
[25] করছেনদুবাইচিড়িয়াখানায়	1	[72] জানাশোনাসঙ্গেপ্রাতিষ্ঠানিক	1
[26] করতেপারলামনা	1	[73] ঠিকমতোখেতেপায়	1
[27] করবেননাপাখিরা	1	[74] ডআলীরেজা	1
[28] করাতারশখ	1	[75] ডরেজাখান	4
[29] করাশুনেইগড়গড়	1	[76] ড.রেজাখানএকজন	1
[30] করেওইসৌন্দর্য	1	[77] ড.রেজাখানপাখিরা	1
[31] করেকীভাবেবাসা	1	[78] ঢাকাবিশ্ববিদ্যলয়েরপ্রাণিবিদ্যা	1
[32] করেবলেদেন	1	[79] ঢাকাবিশ্ববিদ্যালয়েশিক্ষকতার	1
[33] করেসবকিছুইতার	1	[80] ঢাকাবিশ্ববিদ্যালয়েরসহযোগী	1
[34] করেনডরেজা	1	[81] তানিয়েচিন্তিত	1
[35] করেনসুন্দরবনপৃথিবীতে	1	[82] তাকেওধরেরাখতে	1
[36] কাছেনেইওটা	1	[83] তারআত্মারবিষয়	1
[37] কাছেমিনতিকরেন	1	[84] তারনখদর্পণেপাখি	1
[38] কাজকরছেনঅথচ	1	[85] তারবড়ভাব	1
[39] কাজকরছেনদুবাই	1	[86] তারশখবিষয়টিকে	1
[40] কারণেতিনিদেশের	1	[87] তালিকাদিনেদিনে	1
[41] কারণেইসম্ভবতপ্রকৃতি	1	[88] তিনিদেশেরসবার	1
[42] কারোকাছেনেই	1	[89] তিনিপাখিখোঁজেন	1
[43] কিচিরমিচিরকরাশুনেই	1	[90] তিনিপ্রকৃতিপ্রেমিকবলে	1
[44] কিন্তুঅভিমানেতিনিও	1	[91] তিনিপড়াতেনঢাকা	1
[45] কিন্তুঢাকাবিশ্ববিদ্যালয়ে	1	[92] তিনিসাধনারপর্যায়ে	1
[46] কীখায়কোথায়	1	[93] তিনিওসবারমতো	1
[47] কীভাবেবাসাকরে	1	[94] তেমনভাবান্তরনেই	1

[95] তেমনইএকজনযাকে	1	[145] পরিচালকহিসেবেদেশের	1
[96] তেমনিদেশেরাখা	1	[146] পরিচয়দেনপ্রকৃতির	1
[97] থাকতেপছন্দকরে	1	[147] পরিচয়পাখিটিকী	1
[98] থাকেনডরেজা	1	[148] পর্যায়েনিয়েগেছেন	1
[99] থাকেনস্থায়ীভাবেএকজন	1	[149] পাখিখোঁজেনদেশে	1
[100] থেকেএখনদুবাইয়ে	1	[150] পাখিনিয়েগবেষণা	1
[101] দাঁড়িয়েছেকারণেতিনি	1	[151] পাখিবন্যপ্রাণীনিয়ে	1
[102] দাপটেরসঙ্গেকাজ	1	[152] পাখিবিশেষজ্ঞদুবাইয়ে	1
[103] দিনেদিনেদীর্ঘ	1	[153] পাখিবিশেষজ্ঞমানুষটি	1
[104] দিনেদীর্ঘহচ্ছে	1	[154] পাখিসম্পর্কেপড়তে	1
[105] দীর্ঘহচ্ছেআমাদের	1	[155] পাখিগুলোকেমনরয়েছে	1
[106] দুবাইচিড়িয়াখানায়পরিচালক	1	[156] পাখিগুলোকেনিজেকেতিনি	1
[107] দুবাইয়েথাকেনস্থায়ীভাবে	1	[157] পাখিটিকীখায়	1
[108] দুবাইয়েদাপটেরসঙ্গে	1	[158] পাখিটিরপরিচয়পাখিটি	1
[109] দেখতেযানসুন্দরবনের	1	[159] পাখিদেরকিচিরমিচিরকরা	1
[110] দেনওইপাখিটির	1	[160] পাখিরাযারবন্ধু	2
[111] দেনপ্রকৃতিরসঙ্গে	1	[161] পাহুসঠিকপৃষ্ঠপোষকতা	1
[112] দেশডরেজা	1	[162] পারলামনাবলে	1
[113] দেশকেভালোবাসিদেশে	1	[163] পারেনিদেশড	1
[114] দেশত্যাগীহয়েছেনপ্রকৃতি	1	[164] পায়নাতা	1
[115] দেশেএলেইখবর	1	[165] পৃথিবীতেকারোকাছে	1
[116] দেশেএলেইদেখতে	1	[166] পৃষ্ঠপোষকতাপ্রদানেব্যর্থ	1
[117] দেশেরাখাযায়নি	1	[167] পেরেনিজেকেসরিয়ে	1
[118] দেশেরএকজনগবেষক	1	[168] প্রকৃতিতারআত্মার	1
[119] দেশেরপাখিগুলোকেমন	1	[169] প্রকৃতিপ্রেমিকপাখি	1
[120] দেশেরসবারকাছে	1	[170] প্রকৃতিপ্রেমিকবলেপরিচয়	1
[121] দয়াকরেওই	1	[171] প্রকৃতিপ্রেমীড.রেজাখান	1
[122] ধরেরাখতেপারেনি	1	[172] প্রকৃতিরসঙ্গেতার	1
[123] নখদর্পণেপাখিসম্পর্কে	1	[173] প্রদানেব্যর্থহওয়ায়	1
[124] নষ্টকরবেননা	1	[174] প্রাণিবিদ্যাবিভাগেঅনিয়ম	1
[125] নাতানিয়ে	1	[175] প্রাতিষ্ঠানিকশিক্ষাওঢাকা	1
[126] নাপাখিরাযার	1	[176] প্রেমিকপাখিবিশেষজ্ঞ	1
[127] নাপেরেনিজেকে	1	[177] পড়তেবলতেপছন্দ	1
[128] নাবলেরাখা	1	[178] পড়াতেনঢাকাবিশ্ববিদ্যলয়ের	1
[129] নামীবিজ্ঞানীদেরওসেই	1	[179] ফরিদুররহমানপাহু	1
[130] নিজেকেতিনিপ্রকৃতিপ্রেমিক	1	[180] বনে-জঙ্গলেঘুরেঘুরে	1
[131] নিজেকেসরিয়েনিয়েছেন	1	[181] বন্ধুফরিদুররহমান	1
[132] নিতেনাপেরে	1	[182] বন্ধুরাস্তাঘাটেহাঁটার	1
[133] নিতেযাইযশোরের	1	[183] বন্যপ্রাণীনিয়েবিস্তর	1
[134] নিয়তিবলেমেনে	1	[184] বর্তমানেকাজকরছেন	1
[135] নিয়েগবেষণাকরা	1	[185] বলতেপছন্দকরেন	1
[136] নিয়েগেছেনবনে-জঙ্গলে	1	[186] বলেদেনওই	1
[137] নিয়েচিন্তিতথাকেন	1	[187] বলেপরিচয়দেন	1
[138] নিয়েবিস্তরজানাশোনা	1	[188] বলেমেনেনিয়েছেন	1
[139] নিয়েছেনসবআমার	1	[189] বলেরাখাভালো	1
[140] নিয়েছেনসেখানথেকে	1	[190] বাংলাদেশযেমনহারিয়েছে	1
[141] নেইওটাশুধু	1	[191] বাসাকরেসবকিছুই	1
[142] নেইডআলী	1	[192] বিজ্ঞানীদেরওসেইতালিকা	1
[143] পছন্দকরেকীভাবে	1	[193] বিভাগেঅনিয়মবিশৃঙ্খলা	1
[144] পছন্দকরেনড	1	[194] বিশৃঙ্খলামেনেনিতে	1

[195] বিশেষজ্ঞদুবাইয়েদাপটের	1	[245] শুনেইগড়গড়করে	1
[196] বিশেষজ্ঞমানুষটিবর্তমানে	1	[246] সঙ্গেকাজকরছেন	1
[197] বিশ্ববিদ্যলয়েরপ্রাণিবিদ্যাবিভাগে	1	[247] সঙ্গেতারবড়	1
[198] বিশ্ববিদ্যালয়েশিক্ষকতারচাকরিটা	1	[248] সঙ্গেপ্রাতিষ্ঠানিকশিক্ষাও	1
[199] বিশ্ববিদ্যালয়েরসহযোগীঅধ্যাপক	1	[249] সঠিকপৃষ্ঠপোষকতাপ্রদানে	1
[200] বিষয়হয়েদাঁড়িয়েছে	1	[250] সবআমারভাগ্য	1
[201] বিষয়টিকেতিনিসাধনার	1	[251] সবকিছুইতারনখদর্পণে	1
[202] বিষয়টিকেনিয়তিবলে	1	[252] সবাইচেনেনড	1
[203] বিষয়েতেমনভাবান্তর	1	[253] সবারকাছেমিনতি	1
[204] বিস্তরজানাশোনাসঙ্গে	1	[254] সবারমতোদেশত্যাগী	1
[205] ব্যর্থহওয়ায়বাংলাদেশ	1	[255] সম্পদদয়াকরে	1
[206] বড়ভাবযশোরের	1	[256] সম্পর্কেপড়তেবলতে	1
[207] ভাগ্যআমিআমার	1	[257] সম্ভবতপ্রকৃতিতার	1
[208] ভাবযশোরেরহনুমানগুলো	1	[258] সময়পাখিদেরকিচিরমিচির	1
[209] ভাবান্তরনেইড	1	[259] সরিয়েনিয়েছেনসেখান	1
[210] ভালোতিনিপড়াতেন	1	[260] সহযোগীঅধ্যাপককিন্তু	1
[211] ভালোবাসিদেশেএলেই	1	[261] সাধকপাখিনিয়ে	1
[212] মতোতাকেওধরে	1	[262] সাধনারপর্যায়েনিয়ে	1
[213] মতোদেশত্যাগীহয়েছেন	1	[263] সাহাকেতেমনিদেশে	1
[214] মতোনামীবিজ্ঞানীদেরও	1	[264] সুন্দরবনপৃথিবীতেকারো	1
[215] মানুষটিবর্তমানেকাজ	1	[265] সুন্দরবনেরপাখিগুলোকেনিজেকে	1
[216] মিনতিকরেনসুন্দরবন	1	[266] সেবিষয়েতেমন	1
[217] মেঘনাদসাহাকেতেমনি	1	[267] সেইতালিকাদিনে	1
[218] মেনেনিতেনা	1	[268] সেখানথেকেএখন	1
[219] মেনেনিয়েছেনসব	1	[269] সৌন্দর্যনষ্টকরবেন	1
[220] যশোরেরহনুমানগুলোকেমন	1	[270] স্থায়ীভাবেএকজনপ্রকৃতিপ্রেমী	1
[221] যশোরেরহনুমানগুলোঠিকমতো	1	[271] হওয়ায়বাংলাদেশযেমন	1
[222] যাইযশোরেরহনুমানগুলো	1	[272] হচ্ছেআমাদেরসে	1
[223] যাকেসবাইচেনেন	1	[273] হনুমানগুলোকেমনদেশের	1
[224] যানসুন্দরবনেরপাখিগুলোকে	1	[274] হনুমানগুলোঠিকমতোখেতে	1
[225] যারবন্ধুফরিদুর	1	[275] হাঁটারসময়পাখিদের	1
[226] যারবন্ধুরাস্তাঘাটে	1	[276] হারিয়েছেমেঘনাদসাহাকে	1
[227] যাররয়েছেপাখি	1	[277] হিসেবেদেশেরএকজন	1
[228] যায়নিআবেদচৌধুরীর	1	[278] হিসেবেযাররয়েছে	1
[229] যেমনহারিয়েছেমেঘনাদ	1	[279] হয়েদাঁড়িয়েছেকারণে	1
[230] রহমানপাহুসঠিক	1	[280] হয়েছেনপ্রকৃতিপ্রেমিক	1
[231] রাখতেপারেনিদেশ	1		
[232] রাখাভালোতিনি	1		
[233] রাখাযায়নিআবেদ	1		
[234] রাস্তাঘাটেহাঁটারসময়	1		
[235] রেজাখানকারণেই	1		
[236] রেজাখানবিষয়টিকে	1		
[237] রেজাখানহিসেবে	1		
[238] রেজাখানওতেমনই	1		
[239] রয়েছেকিন্তুঢাকা	1		
[240] রয়েছেপাখিবন্যপ্রাণী	1		
[241] শখবিষয়টিকেতিনি	1		
[242] শিক্ষকতারচাকরিটাকরতে	1		
[243] শিক্ষাওঢাকাবিশ্ববিদ্যালয়ের	1		
[244] শুধুআমাদেরসম্পদ	1		

Trigram of Document 2

[1] অনিয়মবিশৃঙ্খলামেনে	1	[48] করেনছবিপাখিরাও	1
[2] অসংখ্যপাখিরছবি	1	[49] করেনডরেজা	1
[3] আজাদনামেপুরো	1	[50] করেনপাখিদেরপাখিদের	1
[4] আত্মারবিষয়হয়ে	1	[51] করেনসুন্দরবনপৃথিবীতে	1
[5] আত্মীয়ওদেরছাড়া	1	[52] কর্মেসহযোগিতাকরাই	1
[6] আমাদেরওইগবেষণা	1	[53] কলেজথেকেইন্টারমিডিয়েট	1
[7] আমাদেরসম্পদদয়া	1	[54] কলেজথেকেনিয়েছেন	1
[8] আমারআত্মীয়ওদের	1	[55] কাছেনেইওটা	1
[9] আমারবন্ধুআমার	1	[56] কাছেমিনতিকরেন	1
[10] আমিএকদিনওচলতে	1	[57] কাছেরমানুষরাতাকে	1
[11] আলীমাআয়েশা	1	[58] কাজেরজন্যতিনিগিয়েছিলেন	1
[12] আলীরেজাখান	1	[59] কারণেতিনিদেশের	1
[13] আয়েশাখাতুনকারণে	1	[60] কারণেইসম্ভবতপ্রকৃতি	1
[14] ইন্টারমিডিয়েটশেষকরে	1	[61] কারোকাছেনেই	1
[15] উদ্দেশ্যহাতিরহাড়ের	1	[62] কিচিরমিচিরকরাশুনেই	1
[16] একজনপ্রকৃতিপ্রেমীড.রেজা	1	[63] কীখায়কোথায়	1
[17] একজনসাধকপাখি	1	[64] কীভাবেবাসাকরে	1
[18] একটিফসিলেরকথা	1	[65] কোথায়থাকতেপছন্দ	1
[19] একটিহাতিরফসিল	1	[66] ক্যামেরানিয়েতুলতে	1
[20] একদিনওচলতেপারি	1	[67] খাওয়ানোথেকেশুরু	1
[21] এখনদুবাইয়েথাকেন	1	[68] খাতুনকারণেতিনি	1
[22] এরপরদেবেন্দ্রকলেজ	1	[69] খানএকজনসাধক	1
[23] এরপরবালিয়াফ্রি	1	[70] খানএকটিফসিলের	1
[24] এলেচলেযান	1	[71] খানকারণেইসম্ভবত	1
[25] এলেইদেখতেযান	1	[72] খানবলেনওরা	1
[26] এসেছিলেনসেসময়	1	[73] খানবড়হয়েছেন	1
[27] ওইগবেষণাকর্মে	1	[74] খানমোঃজোয়াহার	1
[28] ওইগবেষণাকাজের	1	[75] খানসম্পর্কিতবাবা-মা	1
[29] ওইপাখিটিরপরিচয়	1	[76] খানসাতভাই-বোনের	1
[30] ওইসৌন্দর্যনষ্ট	1	[77] খায়কোথায়থাকতে	1
[31] ওটাশুধুআমাদের	1	[78] খেতেপায়না	1
[32] ওদেরছাড়াআমি	1	[79] খোঁজেনদেশেএলেই	1
[33] ওদেরপাশেইথাকেন	1	[80] গবেষণাকরাতার	1
[34] ওরাআমারবন্ধু	1	[81] গবেষণাকর্মেসহযোগিতা	1
[35] কথাবছরতিনি	1	[82] গবেষণাকাজেরজন্যতিনি	1
[36] করবেননাপাখিরা	1	[83] গবেষণাচলছিলআমাদের	1
[37] করাতারশখ	1	[84] গিয়েছিলেনভারতেওতার	1
[38] করাশুনেইগড়গড়	1	[85] গেছেরেজাখান	1
[39] করাইতারসেবারের	1	[86] গেছেনবনে-জঙ্গলেঘুরে	1
[40] করেওইসৌন্দর্য	1	[87] গড়গড়করেবলে	1
[41] করেকীভাবেবাসা	1	[88] ঘুরেঘুরেতিনি	1
[42] করেতাকিয়েথাকে	1	[89] ঘুরেতিনিপাখি	1
[43] করেদেয়ছবি	1	[90] চলছিলআমাদেরওই	1
[44] করেনটরডেম	1	[91] চলতেপারিনা	1
[45] করেবলেদেন	1	[92] চলেযানবনে-বাদাড়ে	1
[46] করেসবকিছুইতার	1	[93] চিনতআজাদনামে	1
[47] করেসেবা-শুশ্রূষারসময়ও	1	[94] চিন্তিতথাকেনড	1

[95] চিড়িয়াখানায়তিনিপ্রতিনিয়ত	1	[143] থাকেনডরেজা	1
[96] চোখকরেতাকিয়ে	1	[144] থাকেনতিনিপাখিদের	1
[97] ছবিতাদেরসম্পর্কে	1	[145] থাকেনস্থায়ীভাবেএকজন	1
[98] ছবিতোলারসময়	1	[146] থেকেইন্টারমিডিয়েটশেষ	1
[99] ছবিপাখিরাওতাকে	1	[147] থেকেএখনদুবাইয়ে	1
[100] ছাড়াআমিএকদিনও	1	[148] থেকেনিয়েছেনবিএসসি	1
[101] ছয়নম্বরপ্রথম	1	[149] থেকেশুরুকরে	1
[102] জন্যতিনিগিয়েছিলেনভারতেও	1	[150] দাঁড়িয়েছেতারসংগ্রহে	1
[103] জানিয়েছিলেনদেশবাসীকেড	1	[151] দিকেডখান	1
[104] জোয়াহারআলীমা	1	[152] দুবাইচিড়িয়াখানায়তিনি	1
[105] ঠিকমতোখেতেপায়	1	[153] দুবাইয়েথাকেনস্থায়ীভাবে	1
[106] ডআলীরেজা	1	[154] দেখতেযানসুন্দরবনের	1
[107] ডখানবলেন	1	[155] দেখভালকরেনপাখিদের	1
[108] ডরেজাখান	3	[156] দেনওইপাখিটির	1
[109] ড.রেজাখানএকজন	1	[157] দেনপ্রকৃতিরসঙ্গে	1
[110] ডেমকলেজথেকে	1	[158] দেবেন্দ্রকলেজথেকে	1
[111] ঢাকাবিশ্ববিদ্যলয়েরপ্রাণিবিদ্যা	1	[159] দেশবাসীকেডরেজা	1
[112] ঢাকায়এসেছিলেনসে	1	[160] দেশেএলেচলে	1
[113] তথ্যদেশেএলে	1	[161] দেশেএলেইদেখতে	1
[114] তানিয়েচিন্তিত	1	[162] দেশেরসবারকাছে	1
[115] তাকিয়েথাকেতার	1	[163] দেয়ছবিতোলার	1
[116] তাকেচিনতআজাদ	1	[164] দয়াকরেওই	1
[117] তাকেসুযোগকরে	1	[165] নখদর্পণেপাখিসম্পর্কে	1
[118] তাদেরসম্পর্কেবিস্তারিত	1	[166] নটরডেমকলেজ	1
[119] তারআত্মারবিষয়	1	[167] নম্বরপ্রথমস্কুলে	1
[120] তারদিকেড	1	[168] নষ্টকরবেননা	1
[121] তারনখদর্পণেপাখি	1	[169] নাতানিয়ে	1
[122] তারবন্ধুদেরসহযোগিতা	1	[170] নাদুবাইচিড়িয়াখানায়	1
[123] তারবড়ভাব	1	[171] নাপাখিরাযার	1
[124] তারভাবহয়ে	1	[172] নাপেরেনিজেকে	1
[125] তারশখবিষয়টিকে	1	[173] নামডআলী	1
[126] তারসংগ্রহেরয়েছে	1	[174] নামবালিয়াপাঠশালা	1
[127] তারসেবারেরসফরের	1	[175] নামেপুরোনাম	1
[128] তিনিছয়নম্বর	1	[176] নিজেকেতিনিপ্রকৃতিপ্রেমিক	1
[129] তিনিঢাকায়এসেছিলেন	1	[177] নিজেকেসরিয়েনিয়েছেন	1
[130] তিনিদেশেরসবার	1	[178] নিতেনাপেরে	1
[131] তিনিপাখিখোঁজেন	1	[179] নিয়েগবেষণাকরা	1
[132] তিনিপাখিদেরসঙ্গে	1	[180] নিয়েগবেষণাচলছিল	1
[133] তিনিপ্রকৃতিপ্রেমিকবলে	1	[181] নিয়েগেছেনবনে-জঙ্গলে	1
[134] তিনিপ্রতিনিয়তদেখভাল	1	[182] নিয়েচিন্তিতথাকেন	1
[135] তিনিপড়াতেনঢাকা	1	[183] নিয়েতিনিফলাফল	1
[136] তিনিপড়েছেনসেটি	1	[184] নিয়েতুলতেশুরু	1
[137] তিনিফলাফলজানিয়েছিলেন	1	[185] নিয়েছেনবিএসসিডিগ্রি	1
[138] তিনিসাধনারপর্যায়ে	1	[186] নিয়েছেনসেখানথেকে	1
[139] তুলতেশুরুকরেন	1	[187] নেইওটাশুধু	1
[140] তোলারসময়বড়	1	[188] পছন্দকরেকীভাবে	1
[141] থাকতেপছন্দকরে	1	[189] পছন্দকরেনড	1
[142] থাকেতারদিকে	1	[190] পরিচয়দেনপ্রকৃতির	1

[191] পরিচয়পাখিটিকী	1	[239] বলেনওরাআমার	1
[192] পর্যায়েনিয়েগেছেন	1	[240] বাবাখানমোঃ	1
[193] পাখিখোঁজেনদেশে	1	[241] বাবা-মাকাছেরমানুষরা	1
[194] পাখিনিয়েগবেষণা	1	[242] বালিয়াপাঠশালাএরপর	1
[195] পাখিসম্পর্কেপড়তে	1	[243] বালিয়াফ্রিপ্রাইমারি	1
[196] পাখিগুলোকেনিজেকেতিনি	1	[244] বাসাকরেসবকিছুই	1
[197] পাখিটিকীখায়	1	[245] বিভাগেঅনিয়মবিশৃঙ্খলা	1
[198] পাখিটিরপরিচয়পাখিটি	1	[246] বিশৃঙ্খলামেনেনিতে	1
[199] পাখিদেরকিচিরমিচিরকরা	1	[247] বিশ্ববিদ্যলয়েরপ্রাণিবিদ্যাবিভাগে	1
[200] পাখিদেরখাওয়ানোথেকে	1	[248] বিষয়হয়েদাঁড়িয়েছে	1
[201] পাখিদেরপাখিদেরখাওয়ানো	1	[249] বিষয়টিকেতিনিসাধনার	1
[202] পাখিদেরসঙ্গেতার	1	[250] বিস্তারিততথ্যদেশে	1
[203] পাখিরছবিতাদের	1	[251] বড়চোখকরে	1
[204] পাখিরাযারবন্ধু	1	[252] বড়বড়চোখ	1
[205] পাখিরাওতাকেসুযোগ	1	[253] বড়ভাবযশোরের	1
[206] পাঠশালাএরপরবালিয়া	1	[254] বড়হয়েছেনমানিকগঞ্জে	1
[207] পারিনাদুবাই	1	[255] ভাই-বোনেরমধ্যেতিনি	1
[208] পাশেইথাকেনতিনি	1	[256] ভাবযশোরেরহনুমানগুলো	1
[209] পায়নাতা	1	[257] ভাবহয়েগেছে	1
[210] পুরোনামড	1	[258] ভারতেওতারবন্ধুদের	1
[211] পৃথিবীতেকারোকাছে	1	[259] ভালোতিনিপড়াতেন	1
[212] পেরেনিজেকেসরিয়ে	1	[260] মধ্যেতিনিছয়	1
[213] প্রকৃতিতারআত্মার	1	[261] মাআয়েশাখাতুন	1
[214] প্রকৃতিপ্রেমিকবলেপরিচয়	1	[262] মানিকগঞ্জসরকারিমাল্টিপারপাস	1
[215] প্রকৃতিপ্রেমীড.রেজাখান	1	[263] মানিকগঞ্জেনামবালিয়া	1
[216] প্রকৃতিরসঙ্গেতার	1	[264] মানিকগঞ্জেবাবাখান	1
[217] প্রতিনিয়তদেখভালকরেন	1	[265] মানুষরাতাকেচিনত	1
[218] প্রথমস্কুলেতিনি	1	[266] মাল্টিপারপাসহাইস্কুলেএরপর	1
[219] প্রাইমারিস্কুলহয়ে	1	[267] মিনতিকরেনসুন্দরবন	1
[220] প্রাণিবিদ্যাবিভাগেঅনিয়ম	1	[268] মেনেনিতেনা	1
[221] পড়তেবলতেপছন্দ	1	[269] মোঃজোয়াহারআলী	1
[222] পড়াতেনঢাকাবিশ্ববিদ্যলয়ের	1	[270] যশোরেরহনুমানগুলোঠিকমতো	1
[223] পড়েছেনমানিকগঞ্জসরকারি	1	[271] যানবনে-বাদাড়েহাতে	1
[224] পড়েছেনসেটিমানিকগঞ্জে	1	[272] যানসুন্দরবনেরপাখিগুলোকে	1
[225] ফলাফলজানিয়েছিলেনদেশবাসীকে	1	[273] যারবন্ধুরাস্তাঘাটে	1
[226] ফসিলনিয়েগবেষণা	1	[274] রাখাভালোতিনি	1
[227] ফসিলেরকথাবছর	1	[275] রাস্তাঘাটেহাঁটারসময়	1
[228] ফ্রিপ্রাইমারিস্কুল	1	[276] রেজাখানএকটি	1
[229] বছরতিনিঢাকায়	1	[277] রেজাখানকারণেই	1
[230] বনে-জঙ্গলেঘুরেঘুরে	1	[278] রেজাখানবড়	1
[231] বনে-বাদাড়েহাতেক্যামেরা	1	[279] রেজাখানসম্পর্কিত	1
[232] বন্ধুআমারআত্মীয়	1	[280] রেজাখানসাত	1
[233] বন্ধুরাস্তাঘাটেহাঁটার	1	[281] রয়েছেঅসংখ্যপাখির	1
[234] বন্ধুদেরসহযোগিতানিয়ে	1	[282] শখবিষয়টিকেতিনি	1
[235] বলতেপছন্দকরেন	1	[283] শুধুআমাদেরসম্পদ	1
[236] বলেদেনওই	1	[284] শুনেইগড়গড়করে	1
[237] বলেপরিচয়দেন	1	[285] শুরুকরেসেবা-শুশ্রূষার	1
[238] বলেরাখাভালো	1	[286] শুরুকরেনছবি	1

[287] শেষকরেনটর	1
[288] সংগ্রহেরয়েছেঅসংখ্য	1
[289] সঙ্গেতারবড়	1
[290] সঙ্গেতারভাব	1
[291] সফরেরউদ্দেশ্যহাতির	1
[292] সবকিছুইতারনখদর্পণে	1
[293] সবারকাছেমিনতি	1
[294] সম্পদদয়াকরে	1
[295] সম্পর্কিতবাবা-মাকাছের	1
[296] সম্পর্কেপড়তেবলতে	1
[297] সম্পর্কেবিস্তারিততথ্য	1
[298] সম্ভবতপ্রকৃতিতার	1
[299] সময়একটিহাতির	1
[300] সময়পাখিদেরকিচিরমিচির	1
[301] সময়বড়বড়	1
[302] সময়ওওদেরপাশেই	1
[303] সরকারিমাল্টিপারপাসহাইস্কুলে	1
[304] সরিয়েনিয়েছেনসেখান	1
[305] সহযোগিতাকরাইতার	1
[306] সহযোগিতানিয়েতিনি	1
[307] সাতভাই-বোনেরমধ্যে	1
[308] সাধকপাখিনিয়ে	1
[309] সাধনারপর্যায়েনিয়ে	1
[310] সুন্দরবনপৃথিবীতেকারো	1
[311] সুন্দরবনেরপাখিগুলোকেনিজেকে	1
[312] সুযোগকরেদেয়	1
[313] সেসময়একটি	1
[314] সেখানথেকেএখন	1
[315] সেটিমানিকগঞ্জেনাম	1
[316] সেবা-শুশ্রূষারসময়ওওদের	1
[317] সেবারেরসফরেরউদ্দেশ্য	1
[318] সৌন্দর্যনষ্টকরবেন	1
[319] স্কুলহয়েপড়েছেন	1
[320] স্কুলেতিনিপড়েছেন	1
[321] স্থায়ীভাবেএকজনপ্রকৃতিপ্রেমী	1
[322] হনুমানগুলোঠিকমতোখেতে	1
[323] হাঁটারসময়পাখিদের	1
[324] হাইস্কুলেএরপরদেবেন্দ্র	1
[325] হাতিরফসিলনিয়ে	1
[326] হাতিরহাড়েরওই	1
[327] হাতেক্যামেরানিয়ে	1
[328] হাড়েরওইগবেষণা	1
[329] হয়েগেছেরেজা	1
[330] হয়েদাঁড়িয়েছেতার	1
[331] হয়েপড়েছেনমানিকগঞ্জ	1
[332] হয়েছেনমানিকগঞ্জেবাবা	1

REFERENCES

1. http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/
2. http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/
3. http://en.wikipedia.org/wiki/N-gram
4. http://en.wikipedia.org/wiki/Apache_Hadoop
5. http://en.wikipedia.org/wiki/Jaccard_index
6. Comparing Different Text Similarity Methods
   JunPengBao et al.

7.The Hadoop Distributed File System
   Konstantin Shvachko, HairongKuang, Sanjay Radia, Robert Chansler

8. Pairwise Document Similarity in Large Collections with MapReduce
   Tamer Elsayed, Jimmy Lin and Douglas W. Oard
9. Demonstration of the Ferret Plagiarism Detector
   Peter C. R. Lane, Caroline M. Lyon & James A. Malcolm
10. Hadoop the Definitive Guide ---- O'Reilly
11. Data Intensive Text Processing ---- Jimmy Lin and Chris Dyer