BIG DATA MINING USING APACHE MAHOUT

BY

Osheen Gulati

Bhavya Pasricha

Kshitij Gurnani

Under the Guidance of

Mr. Adeel Hashmi

Assistant Professor, CSE Department

Department of Computer Science Engineering

MAHARAJA SURAJMAL INSTITUTE OF TECHNOLOGY,

JANAKPURI DELHI-58

GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY

DELHI, INDIA

DEC-2014

ACKNOWLEDGEMENT

A successful project is not a result of efforts of an individual but rather a culmination of support of many. We are indebted to many individuals and are thankful to them for rendering valuable assistance.

We would like to express our deepest appreciation for our project mentor **Mr. Adeel Hashmi** (Assistant Professor, Department of Computer Science, MSIT), who has the attitude and substance of a genius. He continuously and convincingly conveyed a spirit of adventure in regard to the research and an excitement regarding teaching. Without his guidance and persistent help this project would not have been possible.

We would also like to thank the Faculty of Department of Computer Science and Information & Technology (2nd Shift) of Maharaja Surajmal Institute Of Technology for giving us this opportunity and also for the immense support and guidance.

Osheen Gulati Bhavya Pasricha Kshitij Gurnani

CONTENTS

List of Figures

S.No **Title**

ABSTRACT

The projects aims at analyzing big and complex data using mahout and improve the search method by implementing different data mining algorithms using mahout.

Big data is a term for any collection of *data* sets so *large* and complex that it becomes difficult to process using traditional *data* processing applications.

Apache Mahout is a project of the Apache Software Foundation to produce free implementations of distributed or otherwise scalable machine learning algorithms focused primarily in the areas of collaborative filtering, clustering and classification. Many of the implementations use the Apache Hadoop platform.

So we need to study and discover of small to medium datasets collected from multiple domains that are highly distributed and are not easily discoverable and implement various algorithms like collaborative filtering, streaming k-means, spectral clustering etc. to improve search results from previous searches with minimum effort.

1 Problem statement

To analyze big and complex data using apache mahout and improve the search method by implementing different data mining algorithms like fuzzy kmeans, spectral clustering, collaborative filtering etc. using mahout. These algorithms are used for clustering large amounts of data which is the main problem nowadays in many organizations.

As data is generated in large amounts (zetabytes, exabytes etc) from multiple sources like social media, AI, geology , it has become difficult to perform various data mining operations like clustering,classification and collaborative filtering.

Supervised and Unsupervised learning on large amount of data has become a competitive task with big data.

Therefore ,we need a solution to effectively perform data mining on large data sets to get appropriate results. For this, we are using Apache Mahout.

2 INTRODUCTION

1.1 What is Big Data?

Big data is an all-encompassing term for any collection of data sets so large and complex that it becomes difficult to process them using traditional data processing applications.

The challenges include analysis, capture, curation, search, sharing, storage, transfer, visualization, and privacy violations. The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data, as compared to separate smaller sets with the same total amount of data, allowing correlations to be found to "spot business trends, prevent diseases, and combat crime and so on.

As far back as 2001, industry analyst Doug Laney (currently with Gartner) articulated the now mainstream definition of big data as the three Vs of big data: volume, velocity and variety1.

- **Volume**- Many factors contribute to the increase in data volume. Transaction-based data stored through the years. Unstructured data streaming in from social media. Increasing amounts of sensor and machine-to-machine data being collected. In the past, excessive data volume was a storage issue. But with decreasing storage costs, other issues emerge, including how to determine relevance within large data volumes and how to use analytics to create value from relevant data.
- **Velocity**-Data is streaming in at unprecedented speed and must be dealt with in a timely manner. RFID tags, sensors and smart metering are driving the need to deal with torrents of data in near-real time. Reacting quickly enough to deal with data velocity is a challenge for most organizations.
- **Variety**- Data today comes in all types of formats. Structured, numeric data in traditional databases. Information created from line-of-business applications. Unstructured text documents, email, video, audio, stock ticker data and financial transactions. Managing, merging and governing different varieties of data is something many organizations still grapple with.

At SAS, we consider two additional dimensions when thinking about big data:

- **Variability**- In addition to the increasing velocities and varieties of data, data flows can be highly inconsistent with periodic peaks. Is something trending in social media? Daily, seasonal and event-triggered peak data loads can be challenging to manage. Even more so with unstructured data involved.
- **Complexity**- Today's data comes from multiple sources. And it is still an undertaking to link, match, cleanse and transform data across systems. However, it is necessary to connect and correlate relationships, hierarchies and multiple data linkages or your data can quickly spiral out of control.

1.2 WHY BIG DATA?

Terabytes, Petabytes, Exabytes. Who can keep track? These strange terms have just begun to enter the business lexicon, but the hype surrounding them has reached a fever pitch. We have undoubtedly entered the age of big data.

The use of Big Data is becoming a crucial way for leading companies to outperform their peers. In most industries, established competitors and new entrants alike will leverage data-driven strategies to innovate, compete, and capture value. Indeed, we found early examples of such use of data in every sector we examined. In healthcare, data pioneers are analyzing the health outcomes of pharmaceuticals when they were widely prescribed, and discovering benefits and risks that were not evident during necessarily more limited clinical trials. Other early adopters of Big Data are using data from sensors embedded in products from children's toys to industrial goods to determine how these products are actually used in the real world. Such knowledge then informs the creation of new service offerings and the design of future products.

When big data is effectively and efficiently captured, processed, and analyzed, companies are able to gain a more complete understanding of their business, customers, products, competitors, etc. which can lead to efficiency improvements, increased sales, lower costs, better customer service, and/or improved products and services.

Key Big Data Challenges:

- Understanding and Utilizing Big Data – It is a daunting task in most industries and companies.

- New, Complex, and Continuously Emerging Technologies – Since much of the technology that is required in order to utilize big data is new to most organizations.
- Cloud Based Solutions – A new class of business software applications has emerged
  Whereby company data is managed and stored in data centers around the globe.
- Privacy, Security, and Regulatory Considerations - Given the volume and complexity of big data, it is challenging for most firms to obtain a reliable grasp on the content of all of their data and to capture and secure it adequately.
- Privacy, Security, and Regulatory Considerations - Given the volume and complexity of big data, it is challenging for most firms to obtain a reliable grasp on the content of all of their data and to capture and secure it adequately.
- The Need for IT, Data Analyst, and Management Resources – It is estimated that
  there is a need for approximately 140,000 to 190,000 more workers with "deep analytical" expertise and 1.5 million more data-literate managers, either retrained or hired.

1.3 BIG DATA MINING

Data Mining is an analytic process designed to explore data (usually large amounts of data - typically business or market related - also known as "big data") in search of consistent patterns and/or systematic relationships between variables, and then to validate the findings by applying the detected patterns to new subsets of data. The ultimate goal of data mining is prediction - and predictive data mining is the most common type of data mining and one that has the most direct business applications. The process of data mining consists of three stages: (1) the initial exploration, (2) model building or pattern identification with validation/verification, and (3) deployment (i.e., the application of the model to new data in order to generate predictions).

Stage 1: Exploration- This stage usually starts with data preparation which may involve cleaning data, data transformations, selecting subsets of records and - in case of data sets with large numbers of variables ("fields") - performing some preliminary feature selection operations to bring the number of variables

to a manageable range (depending on the statistical methods which are being considered).

Stage 2: Model building and validation- This stage involves considering various models and choosing the best one based on their predictive performance (i.e., explaining the variability in question and producing stable results across samples). This may sound like a simple operation, but in fact, it sometimes involves a very elaborate process. There are a variety of techniques developed to achieve that goal - many of which are based on so-called "competitive evaluation of models," that is, applying different models to the same data set and then comparing their performance to choose the best.

Stage 3: Deployment- That final stage involves using the model selected as best in the previous stage and applying it to new data in order to generate predictions or estimates of the expected outcome.

Big data analytics is the process of examining large data sets containing a variety of data types -- i.e., big data -- to uncover hidden patterns, unknown correlations, market trends, customer preferences and other useful business information. The analytical findings can lead to more effective marketing, new revenue opportunities, better customer service, improved operational efficiency, competitive advantages over rival organizations and other business benefits.

1.4 The Benefits of Big Data Analytics

Enterprises are increasingly looking to find actionable insights into their data. Many big data projects originate from the need to answer specific business questions. With the right big data analytics platforms in place, an enterprise can boost sales, increase efficiency, and improve operations, customer service and risk management.

Webopedia parent company, QuinStreet, surveyed 540 enterprise decision-makers involved in big data purchases to learn which business areas companies plan to use Big Data analytics to improve operations. About half of all respondents said they were applying big data analytics to improve customer retention, help with product development and gain a competitive advantage.

1.5 Hadoop

Hadoop is an open-source software framework for storing and processing big data in a distributed fashion on large clusters of commodity hardware. Essentially, it accomplishes two tasks: massive data storage and faster processing. It has three modules:-

- YARN
- Map Reduce
- Hadoop Distributed File System

1.6 ApacheMahout

Mahout provides scalable machine learning algorithms for Hadoop which aids with data science for clustering, classification and batch based collaborative filtering.

Mahout supports four main data science use cases:

- **Collaborative filtering** – mines user behavior and makes product recommendations (e.g. Amazon recommendations)
- **Clustering** – takes items in a particular class (such as web pages or newspaper articles) and organizes them into naturally occurring groups, such that items belonging to the same group are similar to each other
- **Classification** – learns from existing categorizations and then assigns unclassified items to the best category.
- **Frequent itemset mining** – analyzes items in a group (e.g. items in a shopping cart or terms in a query session) and then identifies which items typically appear together.

3 Literature Survey

International Journal of Data Mining & Knowledge Management Process Vol.4, No.5, September 2014

EXPERIMENTS ON HYPOTHESIS "FUZZY K-MEANS IS BETTER THAN K-MEANS FOR CLUSTERING"

Srinivas Sivarathri1 and A.Govardhan
1Department of Computer Science,
2School of Information Technology,
Jawaharlal Nehru Technological University, Hyderabad, Telangana, India

ALGORITHM	FEATURE	# of Clusters	Fuzziness	Max Iterations	Precision	CPU Time	Compactness
K-MEANS	Single Pass	3	1.7	200	0.01	51ms	9.541E
	Multi Pass	3	1.7	200	0.01	80ms	6.304E
	Best Separation	2	1.7	200	0.01	76ms	6.324E
	Fixed	3	1.7	200	0.01	69ms	6.314E
FUZZY K-MEANS	Fixed	3	1.7	200	0.01	81ms	5.641E
	Best Separation	2	1.7	200	0.01	396ms	6.641E

fig 3.1 comparative analysis between k means and fuzzy k-means

- As can be seen from figure, the accuracy or quality of clusters with Fuzzy K-Means is more when compared to that of K-Means.
- However, K-Means is faster than Fuzzy K-Means and consumes less CPU cycles.

In this paper, we studied the two widely used clustering algorithms for data mining purposes such as K-Means and Fuzzy K-Means. From the literature a hypothesis is conceived such as “Fuzzy K-Means is better than K-Means for Clustering”. We made experiments to know whether the hypothesis holds true. Experiments are made on the real data sets obtained from the UCI repository. The data sets used are related to diabetes disease. We built a prototype application to demonstrate the experiments with the two algorithms in terms of fixed clusters, best separation clusters, single pass clusters, multi-pass clusters, random fixed clusters with single pass, random fixed clusters with multi-pass, exchange best separation clusters, exchange fixed clusters, with other factors such as number of clusters, fuzziness, maximum iterations, precision, CPU time, and compactness. The empirical results revealed that the Fuzzy K – Means take more iterations and the CPU time when compared to that of K-Means. However, the accuracy and quality of the clusters made by Fuzzy K-Means is more comparatively. Thus our experiments proved the hypothesis “Fuzzy K-Means is better than K-Means for Clustering”.

((IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 4, No.4, 2013

Comparative Analysis of K-Means and Fuzzy C-Means Algorithms

Soumi Ghosh
Department of Computer Science and Engineering,
Amity University, Uttar Pradesh
Noida, India
Sanjay Kumar Dubey
Department of Computer Science and Engineering,
Amity University, Uttar Pradesh
Noida, India

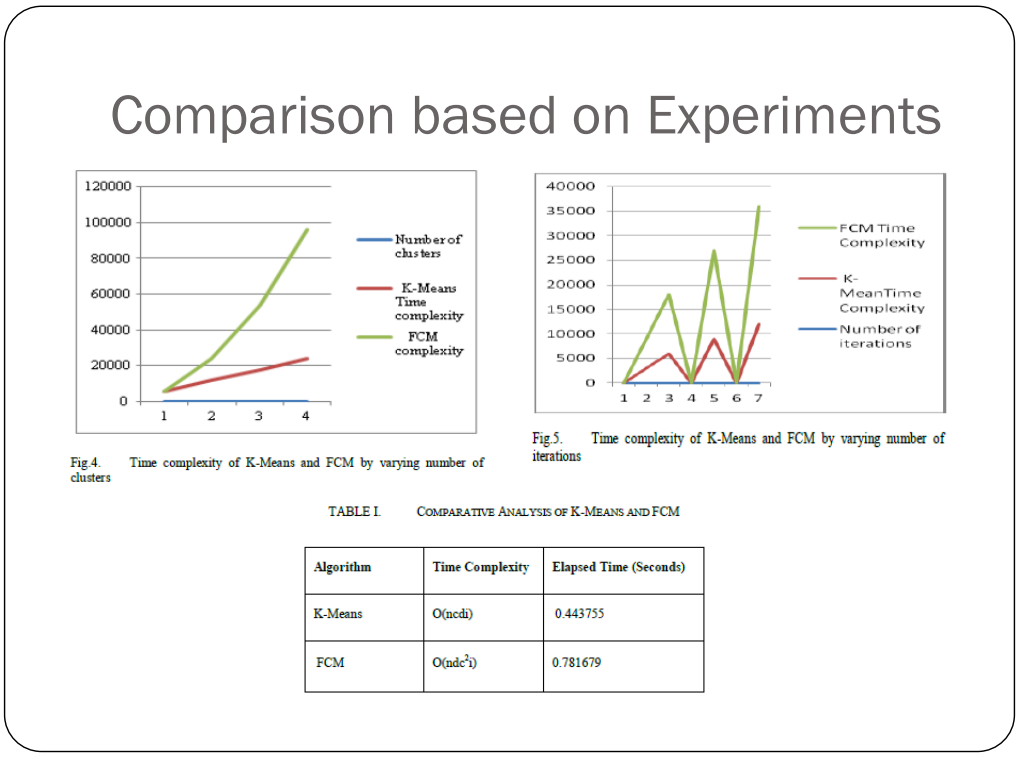

Fig.4. Time complexity of K-Means and FCM by varying number of clusters

Fig.5. Time complexity of K-Means and FCM by varying number of iterations

TABLE I. COMPARATIVE ANALYSIS OF K-MEANS AND FCM

Algorithm	Time Complexity	Elapsed Time (Seconds)
K-Means	O(ncdi)	0.443755
FCM	$O(ndc^2i)$	0.781679

fig 3.2 comparison based on experiments

K-means algorithm-

- is the algorithm which is simple to run and implement.
- Requires less computation time.
- Gives best result when data set are distinct or well separated from each other.
- Low complexity.

K-Means partitioning based clustering algorithm required to define the number of final cluster (k) beforehand. Such algorithms are also having problems like susceptibility to local optima, sensitivity to outliers, memory space and unknown number of iteration steps that are required to cluster. The time

complexity of the K-Means algorithm is O(ncdi) and the time complexity of FCM algorithm is O(ndc2i). From the obtained results we may conclude that K-Means algorithm is better than FCM algorithm. FCM produces close results to K-Means clustering but it still requires more computation time than K-Means because of the fuzzy measures calculations involvement in the algorithm. Infact, FCM clustering which constitute the oldest component of software computing, are really suitable for handling the issues related to understand ability of patterns, incomplete/noisy data, mixed media information, human interaction and it can provide approximate solutions faster. They have been mainly used for discovering association rules and functional dependencies as well as image retrieval. So, overall conclusion is that K-Means algorithm seems to be superior than Fuzzy C-Means algorithm.

International journal of Scientific and technology research volume 1,issue 11,December 2012

Evaluation of k-means and fuzzy k-means in Intrusion Detection Systems

Farhad Soleimanian Gharehchopogh,Neda Jabbari,Zeniab Ghaffari Azar

According to the growth of the Internet technology, there is a need to develop strategies in order to maintain security of system. One of the most effective techniques is Intrusion Detection System (IDS). This system is created to make a complete security in a computerized system, in order to pass the Intrusion system through the firewall, antivirus and other security devices detect and deal with it. The Intrusion detection techniques are divided into two groups which includes supervised learning and unsupervised learning. Clustering which is commonly used to detect possible attacks is one of the branches of unsupervised learning. Fuzzy sets play an important role to reduce spurious alarms and Intrusion detection, which have uncertain quality.This paper investigates k-means fuzzy and k-means algorithm in order to recognize Intrusion detection in system which both of the algorithms use clustering method.

Indeed we investigate K-Means and Fuzzy K-Means algorithms in order to detect intrusion in systems. K-Means algorithm acts better than Fuzzy K-Means in 66% of DOC attacks. In this paper k-means and fuzzy k- means methods have been used to identify the type of DoS attacks.

In this paper a comparison is done between two common algorithms in order to recognize the DoS attacks. Finally, the results derived from K-means algorithm are better to identify these kinds of attacks.

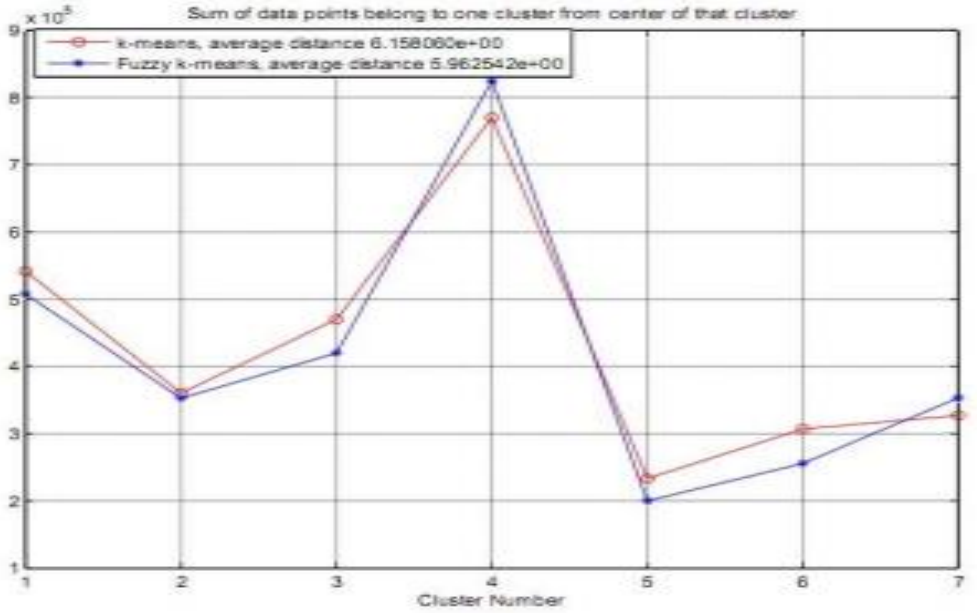

fig 3.3 Evaluation of K-means and Fuzzy K-means in IDS

International journal of emerging technology and advanced engineering,volume 4,issue1,January 2014

A Survey on Data Mining Algorithms on Apache Hadoop Platform

Dr. A.N. Nandakumar, Nandita Yambem

- K means, can process large datasets on commodity hardware effectively. One of the problems noticed when testing is, speed up is not linear with number of nodes, due to communication overhead increase if we increase the data set.
- When global data is needed for synchronization of hadoop mapreduce tasks, it was difficult with current support from hadoop platform.
- Using K-Means algorithm for remote sensing images in Hadoop. One of important lessons learnt while doing this experiment is that hadoop operates only on text and when image has to be represented as text and processed, the overhead in representation and processing is huge even for smaller images.

- While implementing Apriori algorithm on hadoop, Contrary to the believe that parallel processing will take less time to get Frequent item sets, experimental observation proved that multi node Hadoop with differential system configuration was taking more time. The reason was in way the data has been portioned to the nodes.
- Implementation of C4.5 decision tree classification algorithm on apache hadoop was performed. In this work, while constructing the final classifier many duplicates were found. These duplicates could not have avoided if proper data partitioning method have been applied.
- Naïve Bayes is a probabilistic classifier which fits properly to Map reduce architecture. Apache Mahout Implementation of Naïve Bayes has very good performance and reduced the training time. But still improvements can be made it platform is able to support block key value updation mechanism.

- Guidelines for converting serial algorithms to hadoop map reduce algorithms and when to go for approximate solutions are not available.

4 BACKGROUND STUDY

4.1 Hadoop

Hadoop is an open-source software framework for storing and processing big data in a distributed fashion on large clusters of commodity hardware. Essentially, it accomplishes two tasks: massive data storage and faster processing.

- We are using hadoop version 2.5.1.

HISTORY OF HADOOP

As the World Wide Web grew at a dizzying pace in the late 1900s and early 2000s, search engines and indexes were created to help people find relevant information amid all of that text-based content. During the early years, search results were returned by humans. It's true! But as the number of web pages grew from dozens to millions, automation was required. Web crawlers were created, many as university-led research projects, and search engine startups took off (Yahoo, AltaVista, etc.).

One such project was Nutch – an open-source web search engine – and the brainchild of Doug Cutting and Mike Cafarella. Their goal was to invent a way to return web search results faster by distributing data and calculations across different computers so multiple tasks could be accomplished simultaneously. Also during this time, another search engine project called Google was in progress. It was based on the same concept – storing and processing data in a distributed, automated way so that more relevant web search results could be returned faster.

In 2006, Cutting joined Yahoo and took with him the Nutch project as well as ideas based on Google's early work with automating distributed data storage and processing. The Nutch project was divided. The web crawler portion remained as Nutch. The distributed computing and processing portion became Hadoop (named after Cutting's son's toy elephant). In 2008, Yahoo released Hadoop as an open-source project, and, today Hadoop's framework and family of technologies are managed and maintained by the non-profit Apache Software Foundation (ASF), a global community of software developers and contributors.

How Is Hadoop Important?

- **Scalable**– New nodes can be added as needed, and added without needing to change data formats, how data is loaded, how jobs are written, or the applications on top.
- **Flexible**– Hadoop is schema-less, and can absorb any type of data, structured or not, from any number of sources. Data from multiple sources can be joined and aggregated in arbitrary ways enabling deeper analyses than any one system can provide. Unlike traditional relational databases, you don't have to preprocess data before storing it. And that includes unstructured data like text, images and videos. You can store as much data as you want and decide how to use it later.
- **Low cost.** The open-source framework is free and uses commodity hardware to store large quantities of data.
- **Computing power.** Its distributed computing model can quickly process very large volumes of data. The more computing nodes you use, the more processing power you have.
- **Inherent data protection and self-healing capabilities.** Data and application processing are protected against hardware failure. If a node goes down, jobs are automatically redirected to other nodes to make sure the distributed computing does not fail. And it automatically stores multiple copies of all data.

Hadoop Modules

Currently three core components are included:

1. **HDFS** –

   HDFS is a java-based distributed file system that can store all kinds of data without prior organization and provide high-throughput access to data. It provides a limited interface for managing the file system to allow it to scale and provide high throughput. HDFS creates multiple replicas of each data block and distributes them on computers throughout a cluster to enable reliable and rapid access.

The main components of HDFS are as described below:

- **Name Node** is the master of the system. It maintains the name system (directories and files) and manages the blocks which are present on the Data Nodes.
- **Data Nodes** are the slaves which are deployed on each machine and provide the actual storage. They are responsible for serving read and write requests for the clients.
- **Secondary Name Node** is responsible for performing periodic checkpoints. In the event of Name Node failure, you can restart the Name Node using the checkpoint.

2. **Map Reduce** –

   Map Reduce is a framework for performing distributed data processing using the Map Reduce programming paradigm. In the Map Reduce paradigm, each job has a user-defined map phase (which is a parallel, share-nothing processing of input; followed by a user-defined reduce phase where the output of the map phase is aggregated). Typically, HDFS is the storage system for both input and output of the Map Reduce jobs.

The main components of Map Reduce are as described below:

- **Job Tracker** is the master of the system which manages the jobs and resources in the cluster (Task Trackers). The Job Tracker tries to schedule each map as close to the actual data being processed i.e. on the Task Tracker which is running on the same Data Node as the underlying block.
- **Task Trackers** are the slaves which are deployed on each machine. They are responsible for running the map and reduce tasks as instructed by the Job Tracker.
- **Job History Server** is a daemon that serves historical information about completed applications. Typically, Job History server can be co-deployed with Job Tracker, but we recommend to run it as a separate daemon.

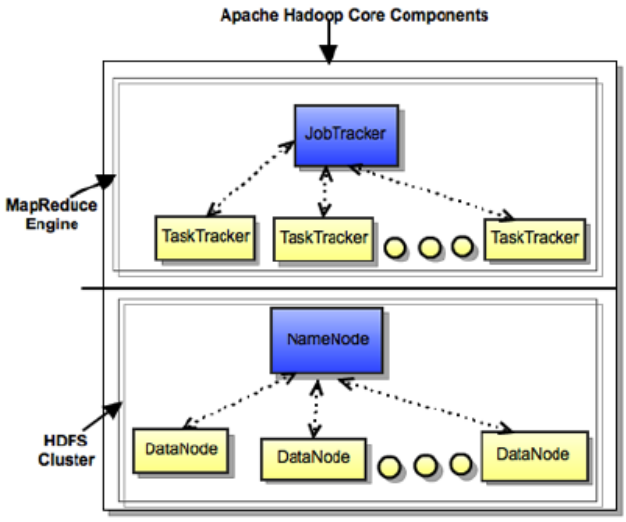

fig 4.1 hadoop architecture

3. **YARN** – stands for "Yet Another Resource Negotiator" is a resource management framework for scheduling and handling resource requests from distributed applications.

The main components of YARN are as described below:

- **Resource Manager** is the ultimate authority that arbitrates resources among all the applications in the system.
- **Application Master**: The per-application Application Master is, in effect, a framework specific entity and is tasked with negotiating resources from the Resource Manager and working with the Node Manager(s) to execute and monitor the component tasks.
- **Node Manager** is YARN's per-node agent, and takes care of the individual nodes in a Hadoop cluster. This includes keeping up-to date with the Resource Manager (RM), overseeing containers life-cycle management; monitoring resource usage (memory, CPU) of individual containers, tracking node-health, log's management and auxiliary services which may be exploited by different YARN applications. .
- **Job History Server** is a daemon that serves historical information about completed applications. Typically, Job History server can be co-deployed with Job Tracker, but we recommend running Job History server as a separate daemon.

4. **Hadoop Common** –
   It contains libraries and utilities needed by other Hadoop modules. The Hadoop Common package contains the necessary Java ARchive (JAR)

files and scripts needed to start Hadoop. The package also provides source code, documentation, and a contribution section that includes projects from the Hadoop Community.

Other components that have achieved top-level Apache project status and are available include:

1) **Apache Accumulo**
   Accumulo is a high performance data storage and retrieval system with cell-level access control. It is a scalable implementation of Google's Big Table design that works on top of Apache Hadoop and Apache ZooKeeper.

2) **Apache HBase**
   A column-oriented NoSQL data storage system that provides random real-time read/write access to big data for user applications.

HBase provides the following benefits:

- **Fault tolerant** storage for large quantities of data
- **Flexible** data model
- **Easy Java API** as well as Thrift, or REST gateway APIs
- **Near real-time** lookups
- **Atomic and strongly consistent** row-level operations
- **Automatic sharding and load balancing** of tables
- **Metrics exports** via File and Ganglia plugins
- **High availability** through automatic failover
- **In-memory caching** via block cache and bloom filters
- **Server side processing** via filters and co-processors
- **Replication** across the data center

3) **Apache HCatalog**
   A table and metadata management service that provides a centralized way for data processing systems to understand the structure and location of the data stored within Apache Hadoop.

Apache HCatalog provides the following benefits to grid administrators:

- Frees the user from having to know where the data is stored, with the table abstraction
- Enables notifications of data availability
- Provides visibility for data cleaning and archiving tools

4) **Apache Hive**
   Built on the MapReduce framework, Hive is a data warehouse that enables easy data summarization and ad-hoc queries via an SQL-like interface for large datasets stored in HDFS.
   We continue to work within the community to advance these three key facets of hive:
   - **Speed-**Deliver sub-second query response times
   - **Scale-**The only SQL interface to Hadoop designed for queries that scale from Gigabytes, to Terabytes and Petabytes
   - **SQL-**Enable transactions and SQL:2011 Analytics for Hive

5) **Apache Mahout**
   Mahout provides scalable machine learning algorithms for Hadoop which aids with data science for clustering, classification and batch based collaborative filtering.

Mahout supports four main data science use cases:

- **Collaborative filtering** – mines user behavior and makes product recommendations (e.g. Amazon recommendations)
- **Clustering** – takes items in a particular class (such as web pages or newspaper articles) and organizes them into naturally occurring groups, such that items belonging to the same group are similar to each other
- **Classification** – learns from existing categorizations and then assigns unclassified items to the best category
- **Frequent itemset mining** – analyzes items in a group (e.g. items in a shopping cart or terms in a query session) and then identifies which items typically appear together

6) **Apache Pig**
   A platform for processing and analyzing large data sets. Pig consists on a high-level language (Pig Latin) for expressing data analysis programs paired with the MapReduce framework for processing these programs.

7) **Apache Solr**
   Solr is the open source platform for searches of data stored in Hadoop. Solr enables powerful full-text search and near real-time indexing on many of the world's largest Internet sites.
8) **Apache Spark**
   Spark is ideal for in-memory data processing. It allows data scientists to

implement fast, iterative algorithms for advanced analytics such as clustering and classification of datasets.

9) **Apache Storm**
   Storm is a distributed real-time computation system for processing fast, large streams of data adding reliable real-time data processing capabilities to Apache Hadoop.

Five characteristics make Storm ideal for real-time data processing workloads. Storm is:

- **Fast** – benchmarked as processing one million 100 byte messages per second per node
- **Scalable** – with parallel calculations that run across a cluster of machines
- **Fault-tolerant** – when workers die, Storm will automatically restart them. If a node dies, the worker will be restarted on another node.
- **Reliable** – Storm guarantees that each unit of data (tuple) will be processed at least once or exactly once. Messages are only replayed when there are failures.
- **Easy to operate** – standard configurations are suitable for production on day one. Once deployed, Storm is easy to operate.

USERS OF HADOOP

- Yahoo
- Facebook
- Hadoop can be deployed in a traditional onsite datacenter as well as in the cloud. The cloud allows organizations to deploy Hadoop without hardware to acquire or specific setup expertise. Vendors who currently have an offer for the cloud include Microsoft, Amazon, and Google.

4.2 Apache Mahout

Apache Mahout is a new open source project by the Apache Software Foundation (ASF) with the primary goal of creating scalable machine-learning algorithms that are free to use under the Apache license. The project is entering its second year, with one public release under its belt. Mahout contains implementations for clustering, categorization, CF, and evolutionary programming. Furthermore, where prudent, it uses the Apache Hadoop library to enable Mahout to scale effectively in the cloud.

The Mahout project was started by several people involved in the Apache Lucene (open source search) community with an active interest in machine learning and a desire for robust, well-documented, scalable implementations of common machine-learning algorithms for clustering and categorization. The community was initially driven by Ng et al.'s paper "Map-Reduce for Machine Learning on Multicore" but has since evolved to cover much broader machine-learning approaches. Mahout also aims to:

- Build and support a community of users and contributors such that the code outlives any particular contributor's involvement or any particular company or university's funding.
- Focus on real-world, practical use cases as opposed to bleeding-edge research or unproven techniques.
- Provide quality documentation and examples.

What's in a name?

A *mahout* is a person who keeps and drives an elephant. The name Mahout comes from the project's (sometime) use of Apache Hadoop — which has a yellow elephant asits logo — for scalability and fault tolerance.

Mahout Versions

- 1 February 2014 - Apache Mahout 0.9 released
- 25 July 2013 - Apache Mahout 0.8 released
- 16 June 2012 - Apache Mahout 0.7 released
- 6 Feb 2012 - Apache Mahout 0.6 released
- 9 Oct 2011 - Mahout in Action released

Introduction to Apache Mahout

Apache Mahout is a project of the Apache Software Foundation to produce free implementations of distributed or otherwise scalable machine learning algorithms focused primarily in the areas of collaborative filtering, clustering and classification. Many of the implementations use the Apache Hadoop platform.[1][2] Mahout also provides Java libraries for common maths operations (focused on linear algebra and statistics) and primitive Java collections.

While Mahout's core algorithms for clustering, classification and batch based collaborative filtering are implemented on top of Apache Hadoop using the map/reduce paradigm, it does not restrict contributions to Hadoop based implementations. Contributions that run on a single node or on a non-Hadoop cluster are also welcomed. For example, the 'Taste' collaborative-filtering recommender component of Mahout was originally a separate project and can run stand-alone without Hadoop.

Weka vs Mahout

Weka is a collection of machine learning algorthims for data mining tasks. The algorithms can be applied directly to a dataset or called from one's java code. Weka contains tools for data pre-processing, classification, regression clustering, association rules and visualization. It is also suitable for developing new machine learning schemes but they are not scalable as in Apache Mahout.

Apache Mahout on the other hand is an open source project by the Apcahe Software Foundation (ASF) with the primary goal of creating scalable machine learning algorithms that are free to use under the Apache license. That is, it is a suite of machine learning libraries designed to be scalable and robust.

Weka allows deep analysis on smaller data sets which can fit the memory on the node on which the tool runs. They can facilitate deep analytics as they have a wide set ig ML algorithms. But they cannot work on large data sets like terabytes or petabytes of data due to scalability limitations because of non distributed nature of these ML algorithms and these are vertically scalable.

Mahout can however scale to large data sets by implementing algorithms over Hadoop, the open source MR implementation. So it is horizontally scalable. Mahout has a set of algorithms for clustering and classification and recommendation for filtering the content to users according to them.

In order to perform machine learning on data, the data has to be converted into name value pair in both the cases. So your decision will depend on the size of your data.

Similarities between Weka and Mahout

- Both tools can be used for Data Mining.
- Both packages support supervised and unsupervised algorithms.
- Both can be used for clustering, classification etc.

Differences between Mahout and Spark Mllib?

The main difference will came from underlying frameworks. In case of Mahout it is Hadoop MapReduce and in case of MLib it is Spark. To be more specific - from the difference in per job overhead
If Your ML algorithm mapped to the single MR job - main difference will be only startup overhead, which is dozens of seconds for Hadoop MR, and let say 1 second for Spark. So in case of model training it is not that important. Things will be different if Your algorithm is mapped to many jobs. In this case we will have the same difference on overhead per iteration and it can be game changer.
Lets assume that we need 100 iterations, each needed 5 seconds of cluster CPU.

- On Spark: it will take 100*5 + 100*1 seconds = 600 seconds.
- On Hadoop: MR (Mahout) it will take 100*5+100*30 = 3500 seconds.

In the same time Hadoop MR is much more mature framework then Spark and if you have a lot of data, and stability is paramount - I would consider Mahout as serious alternative.

Mahout will focus on machine learning and will have a rich set of algorithms, while MLLib will only adopt some mature and essential algorithms.

Similarities between Mahout and Spark Mllib

- Both are open-source projects focusing on large-scale machine learning.
- Mahout and MLLib both have a common exeution engine.
- Mahout provides several important building blocks for creating recommendations using Spark.

PREREQUISITES OF MAHOUT

a) **JAVA-**

It is a set of several computer software and specifications developed by Sun Microsystems, later acquired by Oracle Corporation, that provides a system for developing application software and deploying it in a cross-platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones to enterprise servers and supercomputers. While less common, Java applets run in secure, sandboxed environments to provide many features of native applications and can be embedded in HTML pages.

Knowing java is an added advantage, but java is not strictly a prerequisite for working with hadoop. Let us deep divide in two scenarios.

1)Why java is not strictly a prerequisite.

Tools like Hive and Pig are built on top of hadoop offer their own high level languages for working with data on your cluster. If you want to write your own mapreduce code, you can do so in any language(ex perl, python) that supports reading from standard input and writing to standard output with hadoop streaming.

2)Although you can use streaming to write your map and reduce functions in the language of your choice, there are some advanced features they are at present only available via JAVA API. Also sometimes it is required to get deep into hadoop code as to why something is behaving in a particular way or to know more about the functionality of the particular module. Again knowledge of java comes handy here.

Conclusion: Hadoop projects come with a lot of different roles like architect,developer. Tester etc. Administrator and some of which require explicit knowledge of java and some don’t.

Hadoop and the ecosystem can be easily extended for additional functionality like developing custom Input and OutputFormats, UDF (User Defined Functions) and others. For customizing Hadoop knowledge of Java is mandatory.

Hadoop is written in Java. Its popular Sequence File format is dependent on Java. Even if you use Hive or Pig, you'll probably need to write your own UDF someday. Some people still try to write them in other languages, Java has

more robust and primary support for them. Most Hadoop tools are not mature enough (like Sqoop, HCatalog and so on), so you'll see many Java error stack traces and probably you'll want to hack the source code someday.

Basic necessity: Java 1.6.x or greater.

b) **MAVEN:**

It is a build automation tool used primarily for Java projects. Maven addresses two aspects of building software: First, it describes how software is built, and second, it describes its dependencies. Contrary to preceding tools like Apache Ant it uses conventions for the build procedure, and only exceptions need to be written down. An XML file describes the software project being built, its dependencies on other external modules and components, the build order, directories, and required plug-ins. It comes with pre-defined targets for performing certain well-defined tasks such as compilation of code and its packaging. Maven dynamically downloads Java libraries and Maven plug-ins from one or more repositories such as the Maven 2 Central Repository, and stores them in a local cache.[3] This local cache of downloaded artifacts can also be updated with artifacts created by local projects. Public repositories can also be updated.

Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. The Maven project is hosted by the Apache Software Foundation, where it was formerly part of the Jakarta Project.

There are three things about Maven:

1. Maven will (after you declare which ones you are using) download all the libraries that you use and the libraries that they use for you automatically. This is very nice, and makes dealing with lots of libraries ridiculously easy. This lets you avoid "dependency hell". It is similar to Apache Ant's Ivy.
2. It uses "Convention over Configuration" so that by default you don't need to define the tasks you want to do. You don't need to write a "compile", "test", "package", or "clean" step like you would have to in Ant or a Make file. Just put the files in the places Maven expects them and it should work off of the bat.
3. Maven also has lots of nice plug-ins that you can install that will handle many routine tasks fromgenerating Java classes from an XSD schema using JAXB to measuring test coverage with Cobertura. Just add them to your pom.xml and they will integrate with everything else you want to do.

Basic necessity: Maven 3.x to build the source code

c) **ANT:**

It is a Java library and command-line tool whose mission is to drive processes described in build files as targets and extension points dependent upon each other. The main known usage of Ant is the build of Java applications. Ant supplies a number of built-in tasks allowing to compile, assemble, test and run Java applications. Ant can also be used effectively to build non Java applications, for instance C or C++ applications. More generally, Ant can be used to pilot any type of process which can be described in terms of targets and tasks.

Ant is written in Java. Users of Ant can develop their own "antlibs" containing Ant tasks and types, and are offered a large number of ready-made commercial or open-source "antlibs".

Ant is extremely flexible and does not impose coding conventions or directory layouts to the Java projects which adopt it as a build tool.Ant is an Apache project. It is open source software, and is released under the Apache License.Unlike make, Ant scripts are written in XML to describe the build process and its dependencies. Portability and simplicity of use are two of the main benefits of Ant.

d) **HADOOP**:

Hadoop is an open-source software framework for storing and processing big data in a distributed fashion on large clusters of commodity hardware. Essentially, it accomplishes two tasks: massive data storage and faster processing. Hadoop has two main subprojects:

a) **MapReduce:** The framework that understands and assigns work to the nodes in a cluster.

b) **Hadoop Distributed File System(HDFS):** A file system that spans all the nodes in a Hadoop cluster for data storage. It links together the file systems on many local nodes to make them into one big file system. HDFS assumes nodes will fail, so it achieves reliability by replicating data across multiple nodes

4.3 ALGORITHMS IN MAHOUT

A. CLASSIFICATION

Classification consists of predicting a certain outcome based on a given input. In order to predict the outcome, the algorithm processes a training set containing a set of attributes and the respective outcome, usually called goal or prediction attribute. The algorithm tries to discover relationships between the attributes that would make it possible to predict the outcome. Next the algorithm is given a data set not seen before, called prediction set, which contains the same set of attributes, except for the prediction attribute – not yet known. The algorithm analyses the input and produces a prediction. The prediction accuracy defines how "good" the algorithm is. For example, in amedical database the training set would have relevant patient information recorded previously, where the prediction attribute is whether or not the patient had a heart problem.

The Data Classification process includes the two steps:

- Building the Classifier or Model
- Using Classifier for Classification

BUILDING THE CLASSIFIER OR MODEL

- This step is the learning step or the learning phase.
- In this step the classification algorithms build the classifier.
- The classifier is built from the training set made up of database tuples and their associated class labels.
- Each tuple that constitutes the training set is referred to as a category or class. These tuples can also be referred to as sample, object or data points

USING CLASSIFIER FOR CLASSIFICATION

In this step the classifier is used for classification.Here the test data is used to estimate the accuracy of classification rules. The classification rules can be applied to the new data tuples if the accuracy is considered acceptable.

Classification can be thought of as two separate problems – binary classification and multiclass classification. In binary classification, a better

understood task, only two classes are involved, whereas multiclass classification involves assigning an object to one of several classes. Since many classification methods have been developed specifically for binary classification, multiclass classification often requires the combined use of multiple binary classifiers.

A common subclass of classification is probabilistic classification. Algorithms of this nature use statistical inference to find the best class for a given instance. Unlike other algorithms, which simply output a "best" class, probabilistic algorithms output a probability of the instance being a member of each of the possible classes. The best class is normally then selected as the one with the highest probability. However, such an algorithm has numerous advantages over non-probabilistic classifiers:

- It can output a confidence value associated with its choice (in general, a classifier that can do this is known as a confidence-weighted classifier).
- Correspondingly, it can abstain when its confidence of choosing any particular output is too low.
- Because of the probabilities which are generated, probabilistic classifiers can be more effectively incorporated into larger machine-learning tasks, in a way that partially or completely avoids the problem of error propagation.

1. NAÏVE BAYES CLASSIFICATION:

- The Naive Bayes algorithm is a **probabilistic classification algorithm**. It makes its decisions about which class to assign to an input document using probabilities derived from training data.
- When training, the Naive Bayes algorithm counts the number of times each word appears in a document in the
- class and divides that by the number of words appearing in that class. This is referred to as a conditional probability.

When training, the Naive Bayes algorithm counts the number of times each word appears in a document in the class and divides that by the number of words appearing in that class. This is referred to as a conditional probability, in this case, the probability that a word will appear in a particular category. This is commonly written as P(Word | Category). Imagine you have a small training set that contains three documents for the category geometry, and the word angle appears in one of the documents; there is a probability of 0.33 or 33% that any document labeled geometry would contain the word angle.

P(Category | Document) = P(Document | Category) x P(Category) / P(Document)

2. HIDDEN MARKOV MODEL

The Hidden Markov Model(HMM) is a powerful statistical tool for modeling generative sequences that can be characterised by an underlying process generating an observable sequence. HMMs have found application in many areas interested in signal processing, and in particular speech processing, but have also been applied with success to low level NLP tasks such as part-of-speech tagging, phrase chunking, and extracting target information from documents. Andrei Markov gave his name to the mathematical theory of Markov processes in the early twentieth century, but it was Baum and his colleagues that developed the theory of HMMsin the 1960s.

Three main problems for HMM model:

1. Evaluation: Given a sequence O of observations and a model M, what is the probability P(O|M) that sequence O was generated by model M. The Evaluation problem can be efficiently solved using the Forward algorithm
2. Decoding: Given a sequence O of observations and a model M, what is the most likely sequence Y*=argmax(Y) P(O|M,Y) of hidden variables to generate this sequence. The Decoding problem can be efficiently solved using the Viterbi algorithm.
3. Learning: Given a sequence O of observations, what is the most likely model M*=argmax(M)P(O|M) to generate this sequence. The Learning problem can be efficiently solved using the Baum-Welch algorithm.

B. CLUSTERING

1. K-MEANS ALGORITHM

1. It needs the number of clusters to be known priori. That is nothing but the value of K. With K value known, it defines the number of centroids required. The centroids are to be taken carefully to ensure the cluster quality.
2. Initialization- Form initial centroids based on the number of clusters (K).

3. Classification- Assign each object taken from the data set to the nearest centroid to complete the initial grouping process.
4. Centroid Recalculation-Then re-compute the positions of K centroids
5. Centriod convergence-Repeat the steps 2 and 3 until there is no need for the centroids to be adjusted. Thus, the final clusters are formed.

ADVANTAGES:

- It is the algorithm which is simple to run and implement.
- Requires less computation time.
- Gives best result when data set are distinct or well separated from each other.
- Low complexity.

2. FUZZY K-MEANS ALGORITHM

1. It accepts an input file containing vector points. User can either provide the cluster centers as input or can allow canopy algorithm to run and create initial clusters.
2. Similar to K-Means, the program doesn't modify the input directories. And for every iteration, the cluster output is stored in a directory cluster-N. The code has set number of reduce tasks equal to number of map tasks. So, those many part-0
3. Files are created in cluster N directory. The code uses driver/mapper/combiner/reducer as follows:

- Fuzzy KMeans Driver - It iterates over input points and cluster points for specified number of iterations or until it is converged. During every iteration i, a new cluster-i directory is created which contains the modified cluster centers obtained during Fuzzy K Means iteration. This will be feeded as input clusters in the next iteration. Once Fuzzy KMeans is run for specified number of iterations or until it is converged, a map task is run to output "the point and the cluster membership to each cluster" pair as final output to a directory named "points".
- Fuzzy KMeans Mapper - reads the input cluster during its configure() method, then computes cluster membership probability of a point to each cluster. Cluster membership is inversely propotional to the distance. Distance is computed using user supplied distance measure. Output key is encoded clusterId. Output values are Cluster Observations containing observation statistics.

- Fuzzy K Means Combiner - receives all key: value pairs from the mapper and produces partial sums of the cluster membership probability times input vectors for each cluster. Output key is: encoded cluster identifier. Output values are Cluster Observations containing observation statistics.
- Fuzzy K Means Reducer - Multiple reducers receives certain keys and all values associated with those keys. The reducer sums the values to produce a new centroid for the cluster which is output. Output key is: encoded cluster identifier (e.g. "C14". Output value is: formatted cluster identifier (e.g. "C14"). The reducer encodes unconverged clusters with a 'Cn' cluster Id and converged clusters with 'Vn' cluster Id.

ADVANTAGES:

- Flexible as a single object can belong to more than one cluster.
- They are really suitable for handling the issues related to understand ability of patterns, incomplete/noisy data. They have been mainly used for discovering association rules and functional dependencies as well as image retrieval.
- Allows a data point to be in multiple clusters
- More accurate results.

3. CANOPY ALGORITHM

1. Canopy Clustering is a very simple, fast and surprisingly accurate method for grouping objects into clusters.
2. All objects are represented as a point in a multidimensional feature space. The algorithm uses a fast approximate distance metric and two distance thresholds T1 > T2 for processing.
3. The basic algorithm is to begin with a set of points and remove one at random. Create a Canopy containing this point and iterate through the remainder of the point set.
4. At each point, if its distance from the first point is < T1, then add the point to the cluster. If, in addition, the distance is < T2, then remove the point from the set.
5. This way points that are very close to the original will avoid all further processing. The algorithm loops until the initial set is empty, accumulating a set of Canopies, each containing one or more points. A given point may occur in more than one Canopy.

Canopy Clustering is often used as an initial step in more rigorous clustering techniques, such as K-Means Clustering . By starting with an initial clustering the number of more expensive distance measurements can be significantly reduced by ignoring points outside of the initial canopies.

The **canopy clustering algorithm** is an unsupervised pre-clustering algorithm, often used as preprocessing step for the K-means algorithm or the Hierarchical clustering algorithm.It is intended to speed up clustering operations on large data sets, where using another algorithm directly may be impractical due to the size of the data set.

DISADVANTAGES:

- Since the algorithm uses distance functions and requires the specification of distance thresholds, its applicability for high-dimensional data is limited by the curse of dimensionality. Only when a cheap and approximative – low-dimensional – distance function is available, the produced canopies will preserve the clusters produced by K-means.

ADVANTAGES:

- The number of instances of training data that must be compared at each step is reduced.
- There is some evidence that the resulting clusters are improved

4. STREAMING K-MEANS ALGORITHM

The *Streaming K-Means* algorithm consists of two steps:

1. Streaming step
2. Ball K-Means step.

Streaming step:

Algorithm

The algorithm processes the data one-by-one and makes only one pass through the data. The first point from the data stream will form the centroid of the first cluster (this designation may change as more points are processed). Suppose there are *r* clusters at one point and a new point *p* is being processed.

The new point can either be added to one of the existing *r* clusters or become a new cluster. To decide:

- let *c* be the closest cluster to point *p*
- let *d* be the distance between *c* and *p*
- if *d > distance Cutoff*, create a new cluster from *p* (*p* is too far away from the clusters to be part of any one of them)
- else (*d <= distance Cutoff*), create a new cluster with probability *d / distance Cutoff* (the probability of creating a new cluster increases as *d* increases).

There will be either *r* or *r+1* clusters after processing a new point.
As the number of clusters increases, it will go over the *clusterOvershoot * numClusters* limit (*numClusters* represents a recommendation for the number of clusters that the streaming step should aim for and *clusterOvershoot* is the slack). To decrease the number of clusters the existing clusters are treated as data points and are re-clustered (collapsed). This tends to make the number of clusters go down. If the number of clusters is still too high, *distance Cutoff* is increased.

Ball K-Means step:

Overview

The algorithm is a Lloyd-type algorithm that takes a set of weighted vectors and returns k centroids. The algorithm has two stages:

1. Seeding
2. Ball k-means

The seeding stage is an initial guess of where the centroids should be. The initial guess is improved using the ball k-means stage.

Algorithm

- The algorithm can be instructed to take multiple independent runs (using the *numRuns* parameter) and the algorithm will select the best solution (i.e., the one with the lowest cost). In practice, one run is sufficient to find a good solution.
- Each run operates as follows: a seeding procedure is used to select k centroids, and then ball k-means is run iteratively to refine the solution.
- The seeding procedure can be set to either 'uniformly at random' or 'k-means++' . Seeding with k-means++ involves more computation but offers better results in practice.

- Each iteration of ball k-means runs as follows:

1. Clusters are formed by assigning each data point to the nearest centroid.
2. The centers of mass of the trimmed clusters (see *trim Fraction* parameter above) become the new centroids.

The data may be partitioned into a test set and a training set (see *testProbability*). The seeding procedure and ball k-means run on the training set. The cost is computed on the test set.

C. Collaborative Filtering

Collaborative filtering (**CF**) is a technique used by some recommender systems. Collaborative filtering has two senses, a narrow one and a more general one. In general, collaborative filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc. Applications of collaborative filtering typically involve very large data sets. Collaborative filtering methods have been applied to many different kinds of data including: sensing and monitoring data, such as in mineral exploration, environmental sensing over large areas or multiple sensors; financial data, such as financial service institutions that integrate many financial sources; or in electronic commerce and web applications where the focus is on user data, etc.

In the newer, narrower sense, collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). The underlying assumption of the collaborative filtering approach is that if a person *A* has the same opinion as a person *B* on an issue, A is more likely to have B's opinion on a different issue than to have the opinion on x of a person chosen randomly. For example, a collaborative filtering recommendation system for television tastes could make predictions about which television show a user should like given a partial list of that user's tastes (likes or dislikes).

User Based Collaborative Filtering

Collaborative filtering systems have many forms, but many common systems can be reduced to two steps:

1. Look for users who share the same rating patterns with the active user (the user whom the prediction is for).

2. Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user

1. Item Based Collaborative Filtering

Item-based collaborative filtering invented by Amazon.com (users who bought x also bought y), proceeds in an item-centric manner:

1. Build an item-item matrix determining relationships between pairs of items
2. Infer the tastes of the current user by examining the matrix and matching that user's data

5 Installation

1. Installation of ubuntu

2. Installation of jdk

1) Open Terminal (Applications > Accessories > Terminal.
2) Next, type the following into the terminal: "sudo apt-get install openjdk-7-jre" without quotes and press Enter.

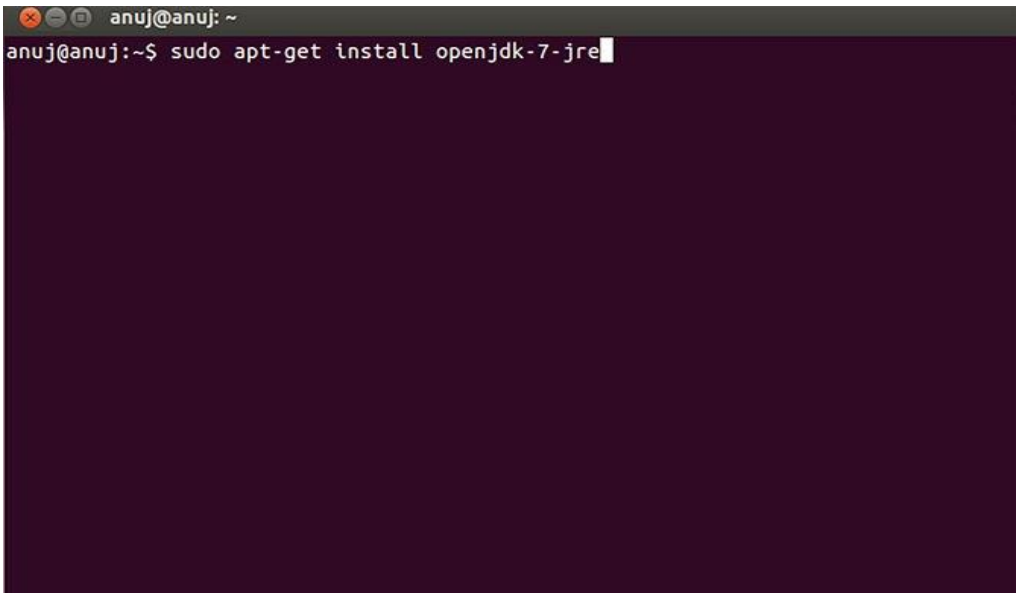

fig 5.1 installation of jdk (1)

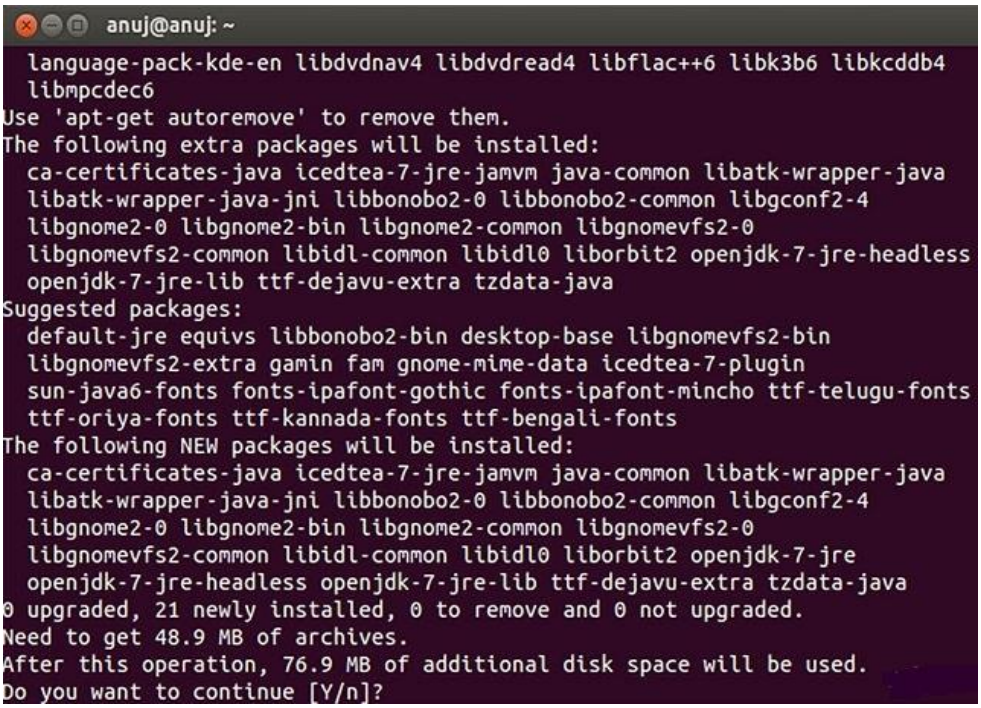

fig 5.2 installation of jdk (2)

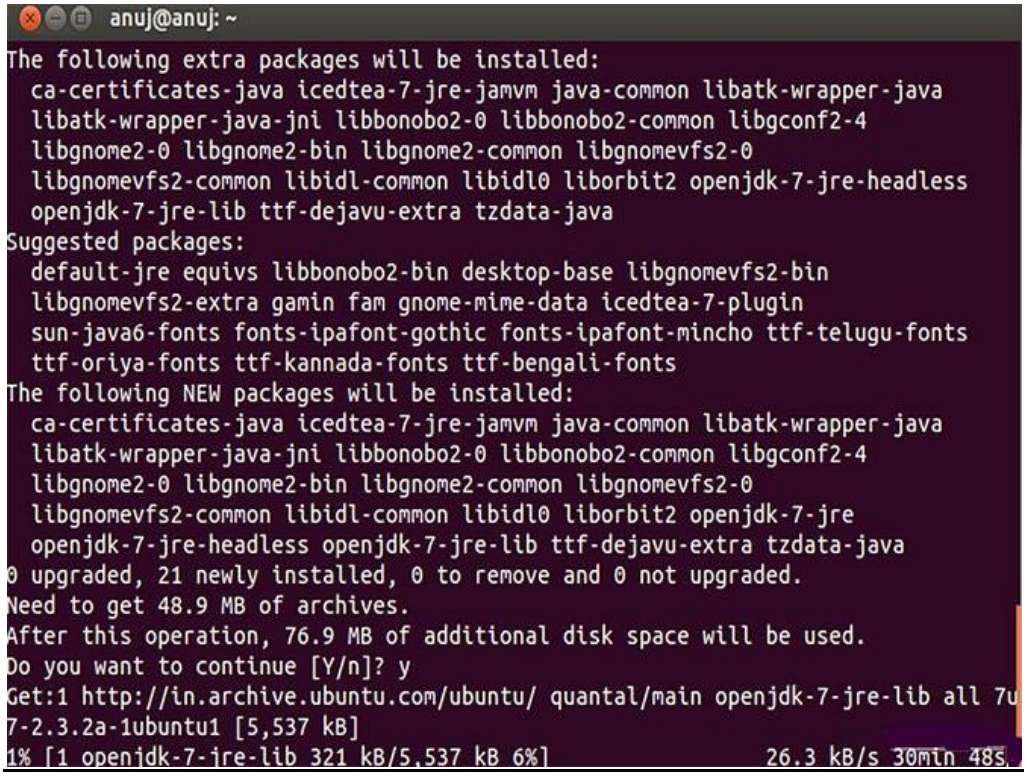

fig 5.3 installation of jdk (3)

3) After accepting the license agreement, Java will download and it will ask you several questions about configuration which you should choose based on what you want.

4) Open ~/.bashrc: $ sudo nano ~/.bashrc

Add these lines to the end of the file:

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export PATH=$JAVA_HOME/bin:$PATH
```

3. Installation of maven

1) Get Maven 3.0.4 binary distribution using the following command:

   http://www.gtlib.gatech.edu/pub/apache/maven/binaries/apache-maven-3.0.4-bin.tar.gz

2) Unpack the binary distribution using the following command:
   tar -zxf apache-maven-3.0.4-bin.tar.gz

3) Run command `sudo apt-get install maven`, to install the latest Apache Maven.

4) Run command `mvn -version` to verify your installation.

```
$ mvn -version
Apache Maven 3.0.4
Maven home: /usr/share/maven
Java version: 1.7.0_09, vendor: Oracle Corporation
Java home: /usr/lib/jvm/java-7-openjdk-amd64/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "3.5.0-17-generic", arch: "amd64", family: "unix"
```

4. Installation of hadoop

1) Install Java
2) Create and setup SSh certificates.

Hadoop uses SSH (to access its nodes) which would normally require the user to enter a password. However, this requirement can be eliminated by creating and setting up SSH certificates using the following commands:

```
ssh-keygen -t rsa -P ''
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

After executing the first of these two commands, you might be asked for a filename. Just leave it blank and press the enter key to continue. The second command adds the newly created key to the list of authorized keys so that Hadoop can use SSH without prompting for a password.

3) First let's fetch Hadoop from one of the mirrors using the following command:

```
wget http://www.motorlogy.com/apache/hadoop/common/current/hadoop-2.3.0.tar.gz
```

After downloading the Hadoop package, execute the following command to extract it:

```
tar xfz hadoop-2.3.0.tar.gz
```

This command will extract all the files in this package in a directory named hadoop-2.3.0. For this tutorial, the Hadoop installation will be moved to the /usr/local/hadoop directory using the following command:

```
mv hadoop-2.3.0 /usr/local/hadoop
```

4) To complete the setup of Hadoop, the following files will have to be modified:

- ~/.bashrc
- /usr/local/hadoop/etc/hadoop/hadoop-env.sh

Editing ~/.bashrc-

Now use nano (or your favored editor) to edit ~/.bashrc using the following command:

```
nano ~/.bashrc
```

This will open the .bashrc file in a text editor. Go to the end of the file and paste/type the following content in it:

```
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
#HADOOP VARIABLES END
```

Editing /usr/local/hadoop/etc/hadoop/hadoop-env.sh-

Open the /usr/local/hadoop/etc/hadoop/hadoop-env.sh file with nano using the following command:

```
nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh
```

In this file, locate the line that exports the JAVA_HOME variable. Change this line to the following:

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

The hadoop-env.sh file should look something like this:

```
# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
# export JAVA_HOME=${JAVA_HOME}
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

fig 5.4 installation of hadoop

Save and close this file. Adding the above statement in the hadoop-env.sh file ensures that the value of JAVA_HOME variable will be available to Hadoop whenever it is started up.

5. Installation of mahout

1) Download the Mahout from below URL using wget- http://apache.techartifact.com/mirror/mahout/0.4/mahout-distribution-0.4-src.tar.gz
2) Unzip the tar file-sudo tar xzf mahout-distribution-0.4-src.tar.gz
3) Now go to Mahout folder and execute bellow command-mvn install

```
chameera@chameera-VirtualBox: ~/Documents/mahout/trunk
ahout/mahout-math-scala/0.9-SNAPSHOT/mahout-math-scala-0.9-SNAPSHOT-tests.jar
[INFO] Installing /home/chameera/Documents/mahout/trunk/math-scala/target/mahout
-math-scala-0.9-SNAPSHOT-sources.jar to /home/chameera/.m2/repository/org/apache
/mahout/mahout-math-scala/0.9-SNAPSHOT/mahout-math-scala-0.9-SNAPSHOT-sources.ja
r
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] Mahout Build Tools ................................ SUCCESS [2:04.586s]
[INFO] Apache Mahout ..................................... SUCCESS [22.332s]
[INFO] Mahout Math ....................................... SUCCESS [5:25.762s]
[INFO] Mahout Core ....................................... SUCCESS [29:01.178s]
[INFO] Mahout Integration ................................ SUCCESS [3:17.823s]
[INFO] Mahout Examples ................................... SUCCESS [55.187s]
[INFO] Mahout Release Package ............................ SUCCESS [0.037s]
[INFO] Mahout Math/Scala wrappers ........................ SUCCESS [2:27.751s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 43:43.610s
[INFO] Finished at: Fri Jan 03 15:26:02 IST 2014
[INFO] Final Memory: 39M/158M
[INFO] ------------------------------------------------------------------------
chameera@chameera-VirtualBox:~/Documents/mahout/trunk$
```

fig 5.5 installation of mahout

6 SOFTWARE AND HARDWARE REQUIREMENTS

- RAM 2GB and above
- Hard disk space 15GB(minimum)
- Ubuntu 12.04 and above
- Jdk 1.6.x or higher
- Maven 3.0.4 or higher
- Hadoop 1.2.1 or higher

7) Implementation

7.1 K-means clustering

```
Please select a number to choose the corresponding clustering algorithm
1. kmeans clustering
2. fuzzykmeans clustering
3. lda clustering
4. streamingkmeans clustering
Enter your choice : 1
ok. You chose 1 and we'll use kmeans Clustering
creating work directory at /tmp/mahout-work-admin1
Running on hadoop, using /usr/bin/hadoop-1.2.1/bin/hadoop and HADOOP_CONF_DIR=
MAHOUT-JOB: /home/admin1/Downloads/mahout-distribution-0.9/mahout-examples-0.9-j
ob.jar
14/11/26 23:17:05 INFO vectorizer.SparseVectorsFromSequenceFiles: Maximum n-gram
```

fig 7.1

```
admin1@ubuntu: ~/Downloads/mahout-distribution-0.9/examples/bin
 size is: 1
14/11/26 23:17:05 INFO vectorizer.SparseVectorsFromSequenceFiles: Minimum LLR va
lue: 1.0
14/11/26 23:17:05 INFO vectorizer.SparseVectorsFromSequenceFiles: Number of redu
ce tasks: 1
14/11/26 23:17:05 INFO vectorizer.SparseVectorsFromSequenceFiles: Tokenizing doc
uments in /tmp/mahout-work-admin1/reuters-out-seqdir
14/11/26 23:17:05 INFO common.HadoopUtil: Deleting /tmp/mahout-work-admin1/reute
rs-out-seqdir-sparse-kmeans/tokenized-documents
14/11/26 23:17:05 INFO util.NativeCodeLoader: Loaded the native-hadoop library
14/11/26 23:17:06 INFO input.FileInputFormat: Total input paths to process : 1
14/11/26 23:17:06 INFO mapred.JobClient: Running job: job_local1663672356_0001
14/11/26 23:17:06 INFO mapred.LocalJobRunner: Waiting for map tasks
14/11/26 23:17:06 INFO mapred.LocalJobRunner: Starting task: attempt_local166367
2356_0001_m_000000_0
14/11/26 23:17:06 INFO util.ProcessTree: setsid exited with exit code 0
14/11/26 23:17:06 INFO mapred.Task:  Using ResourceCalculatorPlugin : org.apache
.hadoop.util.LinuxResourceCalculatorPlugin@1cea92
14/11/26 23:17:06 INFO mapred.MapTask: Processing split: file:/tmp/mahout-work-a
dmin1/reuters-out-seqdir/chunk-0:0+18403626
14/11/26 23:17:07 INFO mapred.JobClient:  map 0% reduce 0%
14/11/26 23:17:09 INFO mapred.Task: Task:attempt_local1663672356_0001_m_000000_0
 is done. And is in the process of commiting
14/11/26 23:17:09 INFO mapred.LocalJobRunner:
14/11/26 23:17:09 INFO mapred.Task: Task attempt_local1663672356_0001_m_000000_0
 is allowed to commit now
14/11/26 23:17:09 INFO output.FileOutputCommitter: Saved output of task 'attempt
_local1663672356_0001_m_000000_0' to /tmp/mahout-work-admin1/reuters-out-seqdir-
sparse-kmeans/tokenized-documents
14/11/26 23:17:09 INFO mapred.LocalJobRunner:
14/11/26 23:17:09 INFO mapred.Task: Task 'attempt_local1663672356_0001_m_000000_
0' done.
14/11/26 23:17:09 INFO mapred.LocalJobRunner: Finishing task: attempt_local16636
72356_0001_m_000000_0
14/11/26 23:17:09 INFO mapred.LocalJobRunner: Map task executor complete.
14/11/26 23:17:10 INFO mapred.JobClient:  map 100% reduce 0%
14/11/26 23:17:10 INFO mapred.JobClient: Job complete: job_local1663672356_0001
14/11/26 23:17:10 INFO mapred.JobClient: Counters: 12
14/11/26 23:17:10 INFO mapred.JobClient:   File Output Format Counters
```

fig 7.2

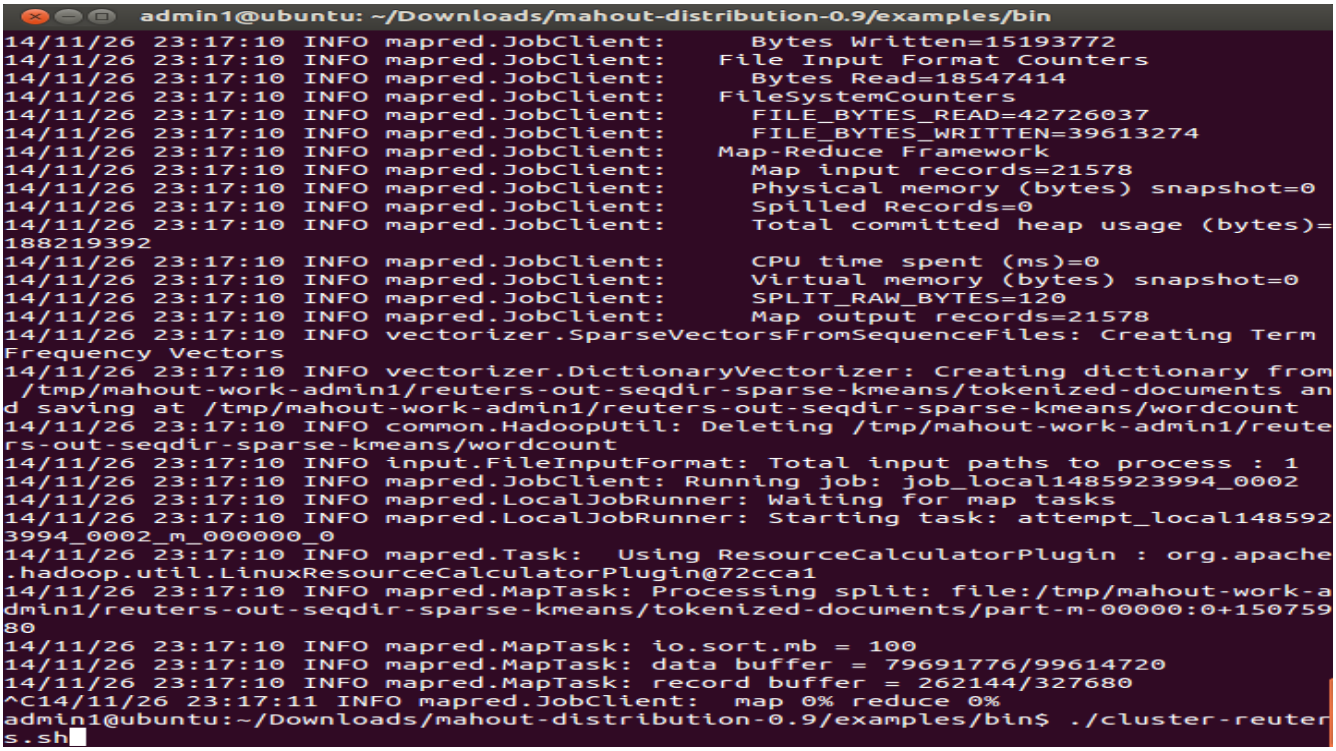
admin1@ubuntu: ~/Downloads/mahout-distribution-0.9/examples/bin
14/11/26 23:17:10 INFO mapred.JobClient: Bytes Written=15193772
14/11/26 23:17:10 INFO mapred.JobClient: File Input Format Counters
14/11/26 23:17:10 INFO mapred.JobClient: Bytes Read=18547414
14/11/26 23:17:10 INFO mapred.JobClient: FileSystemCounters
14/11/26 23:17:10 INFO mapred.JobClient: FILE_BYTES_READ=42726037
14/11/26 23:17:10 INFO mapred.JobClient: FILE_BYTES_WRITTEN=39613274
14/11/26 23:17:10 INFO mapred.JobClient: Map-Reduce Framework
14/11/26 23:17:10 INFO mapred.JobClient: Map input records=21578
14/11/26 23:17:10 INFO mapred.JobClient: Physical memory (bytes) snapshot=0
14/11/26 23:17:10 INFO mapred.JobClient: Spilled Records=0
14/11/26 23:17:10 INFO mapred.JobClient: Total committed heap usage (bytes)=
188219392
14/11/26 23:17:10 INFO mapred.JobClient: CPU time spent (ms)=0
14/11/26 23:17:10 INFO mapred.JobClient: Virtual memory (bytes) snapshot=0
14/11/26 23:17:10 INFO mapred.JobClient: SPLIT_RAW_BYTES=120
14/11/26 23:17:10 INFO mapred.JobClient: Map output records=21578
14/11/26 23:17:10 INFO vectorizer.SparseVectorsFromSequenceFiles: Creating Term
Frequency Vectors
14/11/26 23:17:10 INFO vectorizer.DictionaryVectorizer: Creating dictionary from
/tmp/mahout-work-admin1/reuters-out-seqdir-sparse-kmeans/tokenized-documents an
d saving at /tmp/mahout-work-admin1/reuters-out-seqdir-sparse-kmeans/wordcount
14/11/26 23:17:10 INFO common.HadoopUtil: Deleting /tmp/mahout-work-admin1/reute
rs-out-seqdir-sparse-kmeans/wordcount
14/11/26 23:17:10 INFO input.FileInputFormat: Total input paths to process : 1
14/11/26 23:17:10 INFO mapred.JobClient: Running job: job_local1485923994_0002
14/11/26 23:17:10 INFO mapred.LocalJobRunner: Waiting for map tasks
14/11/26 23:17:10 INFO mapred.LocalJobRunner: Starting task: attempt_local148592
3994_0002_m_000000_0
14/11/26 23:17:10 INFO mapred.Task: Using ResourceCalculatorPlugin : org.apache
.hadoop.util.LinuxResourceCalculatorPlugin@72cca1
14/11/26 23:17:10 INFO mapred.MapTask: Processing split: file:/tmp/mahout-work-a
dmin1/reuters-out-seqdir-sparse-kmeans/tokenized-documents/part-m-00000:0+150759
80
14/11/26 23:17:10 INFO mapred.MapTask: io.sort.mb = 100
14/11/26 23:17:10 INFO mapred.MapTask: data buffer = 79691776/99614720
14/11/26 23:17:10 INFO mapred.MapTask: record buffer = 262144/327680
^C14/11/26 23:17:11 INFO mapred.JobClient: map 0% reduce 0%
admin1@ubuntu:~/Downloads/mahout-distribution-0.9/examples/bin$ ./cluster-reuter
s.sh

fig 7.3

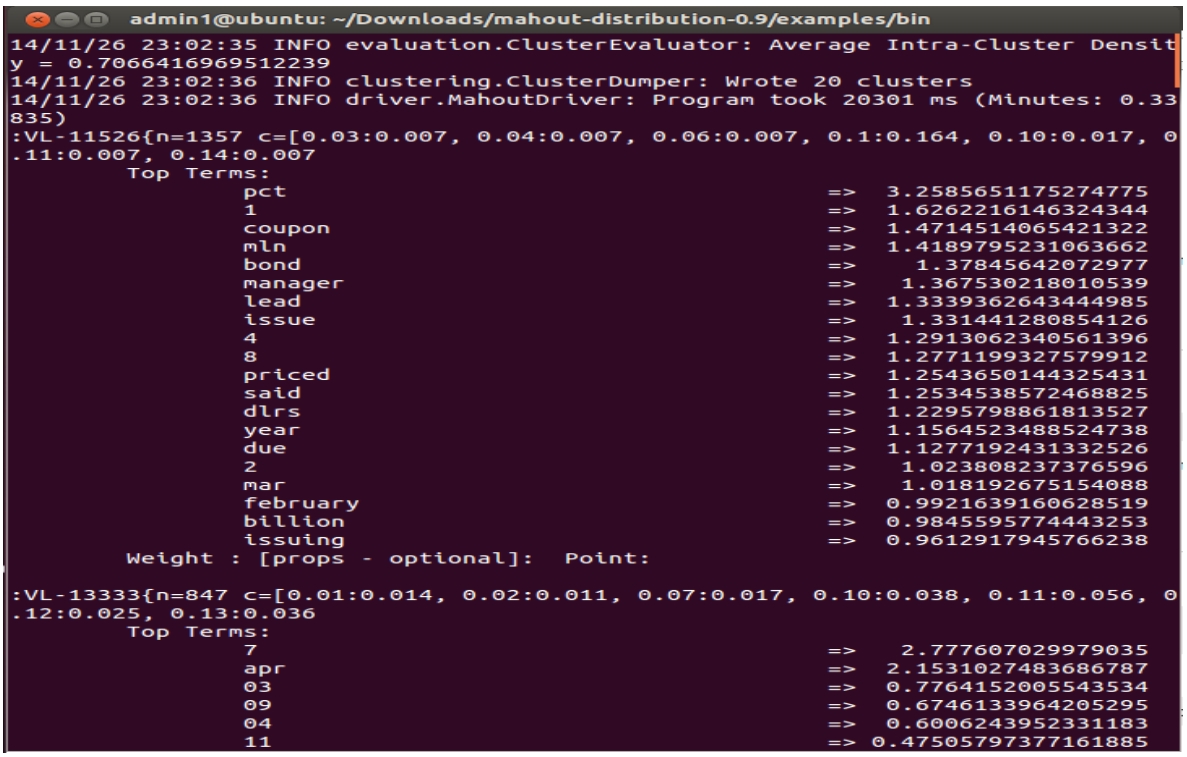
admin1@ubuntu: ~/Downloads/mahout-distribution-0.9/examples/bin
14/11/26 23:02:35 INFO evaluation.ClusterEvaluator: Average Intra-Cluster Densit
y = 0.7066416969512239
14/11/26 23:02:36 INFO clustering.ClusterDumper: Wrote 20 clusters
14/11/26 23:02:36 INFO driver.MahoutDriver: Program took 20301 ms (Minutes: 0.33
835)
:VL-11526{n=1357 c=[0.03:0.007, 0.04:0.007, 0.06:0.007, 0.1:0.164, 0.10:0.017, 0
.11:0.007, 0.14:0.007
Top Terms:
pct => 3.2585651175274775
1 => 1.6262216146324344
coupon => 1.4714514065421322
mln => 1.4189795231063662
bond => 1.37845642072977
manager => 1.367530218010539
lead => 1.3339362643444985
issue => 1.331441280854126
4 => 1.2913062340561396
8 => 1.2771199327579912
priced => 1.2543650144325431
said => 1.2534538572468825
dlrs => 1.2295798861813527
year => 1.1564523488524738
due => 1.1277192431332526
2 => 1.023808237376596
mar => 1.018192675154088
february => 0.9921639160628519
billion => 0.9845595774443253
issuing => 0.9612917945766238
Weight : [props - optional]: Point:
:VL-13333{n=847 c=[0.01:0.014, 0.02:0.011, 0.07:0.017, 0.10:0.038, 0.11:0.056, 0
.12:0.025, 0.13:0.036
Top Terms:
7 => 2.777607029979035
apr => 2.1531027483686787
03 => 0.7764152005543534
09 => 0.6746133964205295
04 => 0.6006243952331183
11 => 0.47505797377161885

fig 7.4

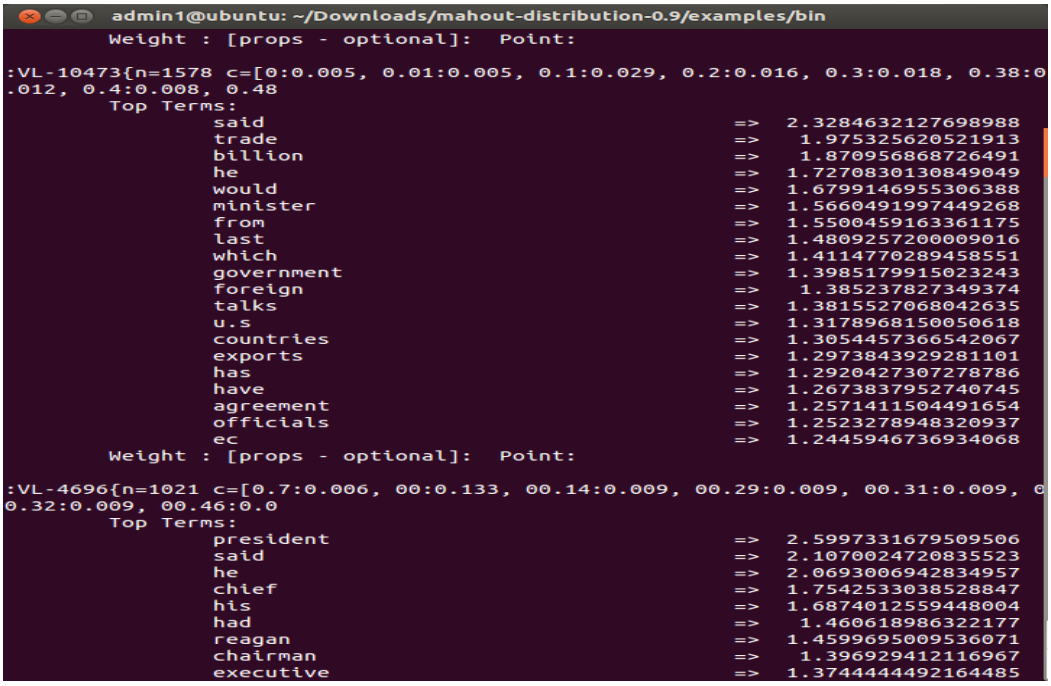
admin1@ubuntu: ~/Downloads/mahout-distribution-0.9/examples/bin
Weight : [props - optional]: Point:
:VL-10473{n=1578 c=[0:0.005, 0.01:0.005, 0.1:0.029, 0.2:0.016, 0.3:0.018, 0.38:0
.012, 0.4:0.008, 0.48
Top Terms:
said => 2.3284632127698988
trade => 1.975325620521913
billion => 1.870956868726491
he => 1.7270830130849049
would => 1.6799146955306388
minister => 1.5660491997449268
from => 1.5500459163361175
last => 1.4809257200009016
which => 1.4114770289458551
government => 1.3985179915023243
foreign => 1.385237827349374
talks => 1.3815527068042635
u.s => 1.3178968150050618
countries => 1.3054457366542067
exports => 1.2973843929281101
has => 1.2920427307278786
have => 1.2673837952740745
agreement => 1.2571411504491654
officials => 1.2523278948320937
ec => 1.2445946736934068
Weight : [props - optional]: Point:
:VL-4696{n=1021 c=[0.7:0.006, 00:0.133, 00.14:0.009, 00.29:0.009, 00.31:0.009, 0
0.32:0.009, 00.46:0.0
Top Terms:
president => 2.5997331679509506
said => 2.1070024720835523
he => 2.0693006942834957
chief => 1.7542533038528847
his => 1.6874012559448004
had => 1.460618986322177
reagan => 1.4599695009536071
chairman => 1.396929412116967
executive => 1.3744444492164485

fig 7.5

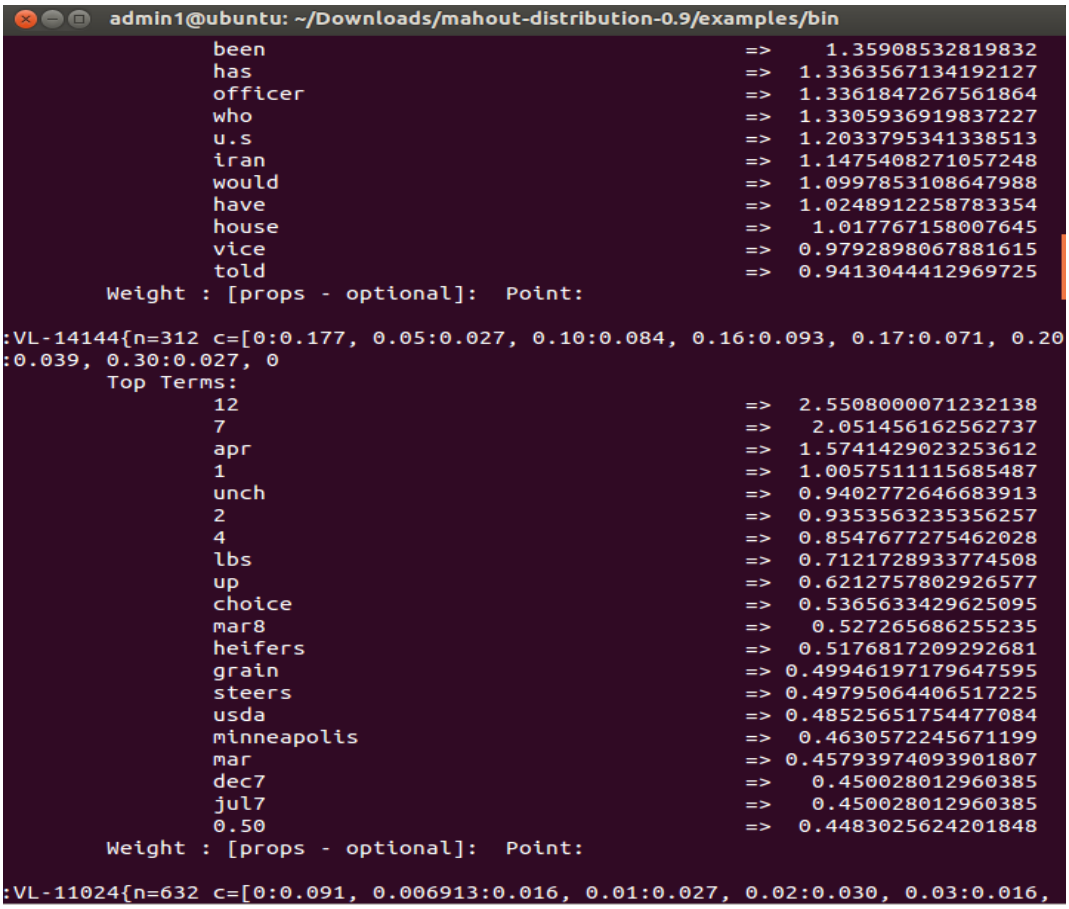
admin1@ubuntu: ~/Downloads/mahout-distribution-0.9/examples/bin
been => 1.35908532819832
has => 1.3363567134192127
officer => 1.3361847267561864
who => 1.3305936919837227
u.s => 1.2033795341338513
iran => 1.1475408271057248
would => 1.0997853108647988
have => 1.0248912258783354
house => 1.017767158007645
vice => 0.9792898067881615
told => 0.9413044412969725
Weight : [props - optional]: Point:
:VL-14144{n=312 c=[0:0.177, 0.05:0.027, 0.10:0.084, 0.16:0.093, 0.17:0.071, 0.20
:0.039, 0.30:0.027, 0
Top Terms:
12 => 2.5508000071232138
7 => 2.051456162562737
apr => 1.5741429023253612
1 => 1.0057511115685487
unch => 0.9402772646683913
2 => 0.9353563235356257
4 => 0.8547677275462028
lbs => 0.7121728933774508
up => 0.6212757802926577
choice => 0.5365633429625095
mar8 => 0.527265686255235
heifers => 0.5176817209292681
grain => 0.49946197179647595
steers => 0.49795064406517225
usda => 0.48525651754477084
minneapolis => 0.4630572245671199
mar => 0.45793974093901807
dec7 => 0.450028012960385
jul7 => 0.450028012960385
0.50 => 0.4483025624201848
Weight : [props - optional]: Point:
:VL-11024{n=632 c=[0:0.091, 0.006913:0.016, 0.01:0.027, 0.02:0.030, 0.03:0.016,

fig 7.6

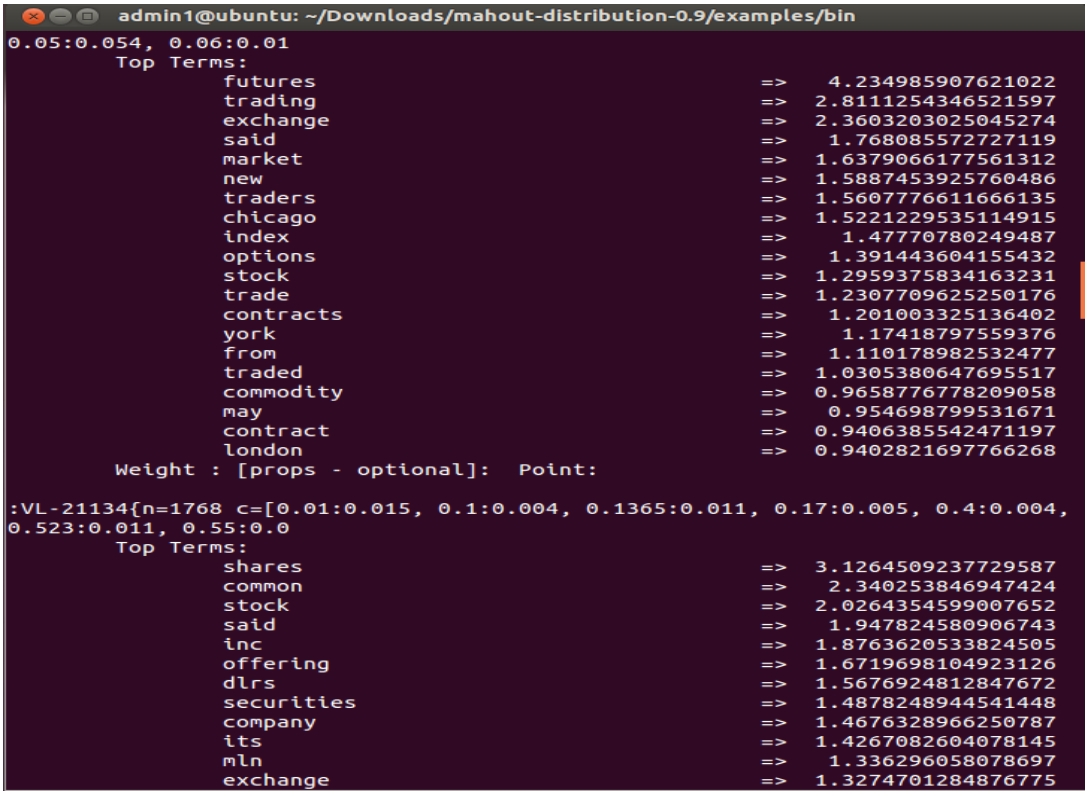
admin1@ubuntu: ~/Downloads/mahout-distribution-0.9/examples/bin
0.05:0.054, 0.06:0.01
Top Terms:
futures => 4.234985907621022
trading => 2.8111254346521597
exchange => 2.3603203025045274
said => 1.768085572727119
market => 1.6379066177561312
new => 1.5887453925760486
traders => 1.5607776611666135
chicago => 1.5221229535114915
index => 1.47770780249487
options => 1.391443604155432
stock => 1.2959375834163231
trade => 1.230770962525017 6
contracts => 1.201003325136402
york => 1.17418797559376
from => 1.110178982532477
traded => 1.0305380647695517
commodity => 0.9658776778209058
may => 0.954698799531671
contract => 0.9406385542471197
london => 0.9402821697766268
Weight : [props - optional]: Point:
:VL-21134{n=1768 c=[0.01:0.015, 0.1:0.004, 0.1365:0.011, 0.17:0.005, 0.4:0.004,
0.523:0.011, 0.55:0.0
Top Terms:
shares => 3.1264509237729587
common => 2.340253846947424
stock => 2.0264354599007652
said => 1.947824580906743
inc => 1.8763620533824505
offering => 1.6719698104923126
dlrs => 1.5676924812847672
securities => 1.4878248944541448
company => 1.4676328966250787
its => 1.4267082604078145
mln => 1.336296058078697
exchange => 1.3274701284876775

fig 7.7

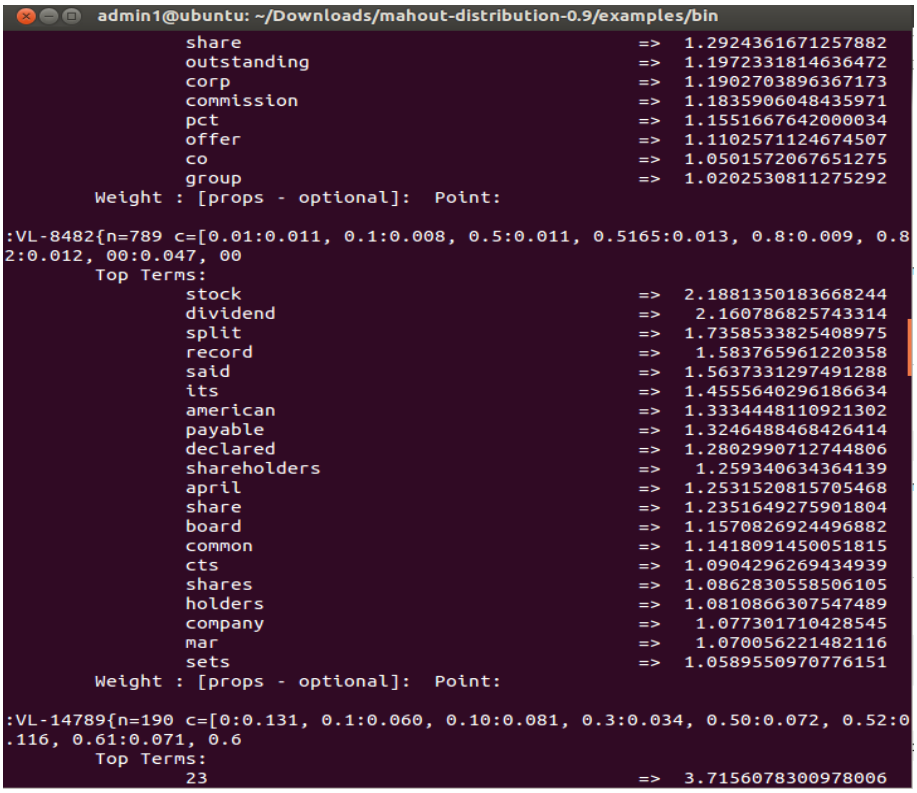
admin1@ubuntu: ~/Downloads/mahout-distribution-0.9/examples/bin
share => 1.2924361671257882
outstanding => 1.1972331814636472
corp => 1.1902703896367173
commission => 1.1835906048435971
pct => 1.1551667642000034
offer => 1.1102571124674507
co => 1.0501572067651275
group => 1.0202530811275292
Weight : [props - optional]: Point:
:VL-8482{n=789 c=[0.01:0.011, 0.1:0.008, 0.5:0.011, 0.5165:0.013, 0.8:0.009, 0.8
2:0.012, 00:0.047, 00
Top Terms:
stock => 2.1881350183668244
dividend => 2.160786825743314
split => 1.7358533825408975
record => 1.583765961220358
said => 1.5637331297491288
its => 1.4555640296186634
american => 1.3334448110921302
payable => 1.3246488468426414
declared => 1.2802990712744806
shareholders => 1.259340634364139
april => 1.2531520815705468
share => 1.2351649275901804
board => 1.1570826924496882
common => 1.1418091450051815
cts => 1.0904296269434939
shares => 1.0862830558506105
holders => 1.0810866307547489
company => 1.077301710428545
mar => 1.070056221482116
sets => 1.0589550970776151
Weight : [props - optional]: Point:
:VL-14789{n=190 c=[0:0.131, 0.1:0.060, 0.10:0.081, 0.3:0.034, 0.50:0.072, 0.52:0
.116, 0.61:0.071, 0.6
Top Terms:
23 => 3.7156078300978006

fig 7.8

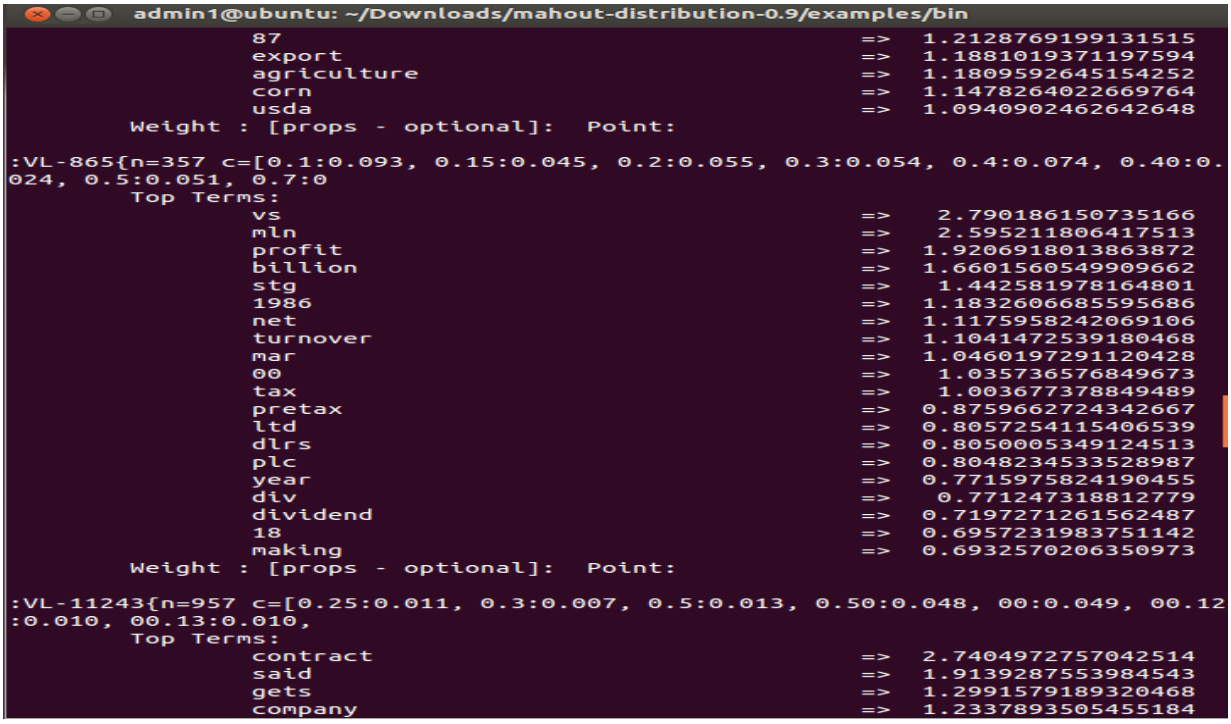

fig 7.9

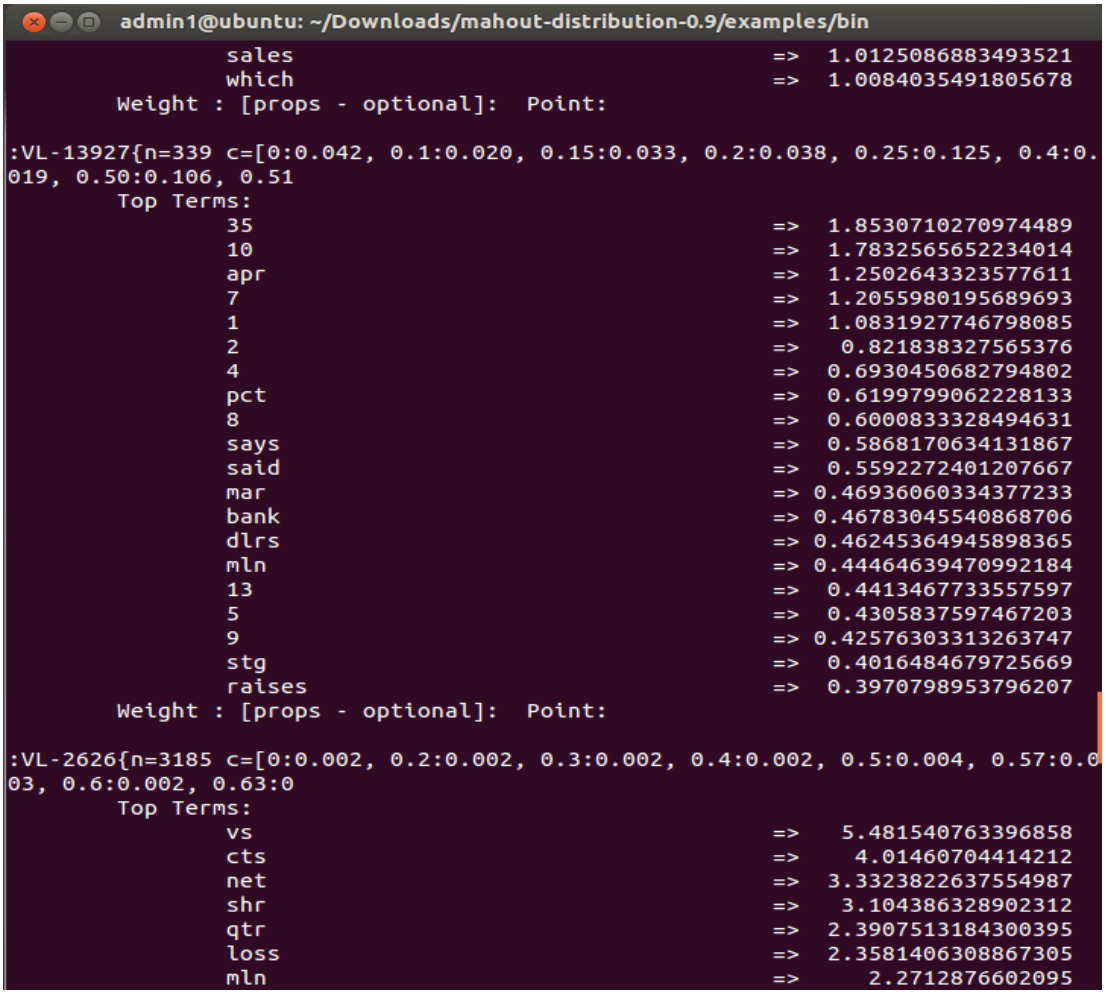

fig 7.10

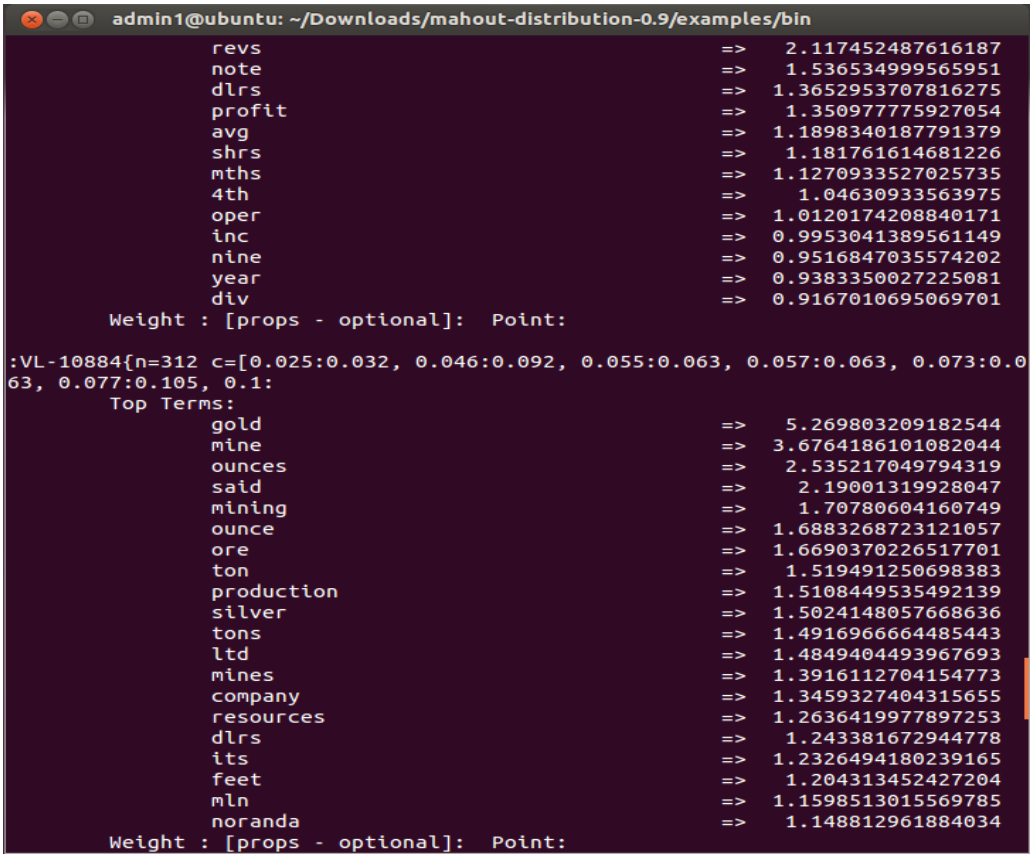

fig 7.11

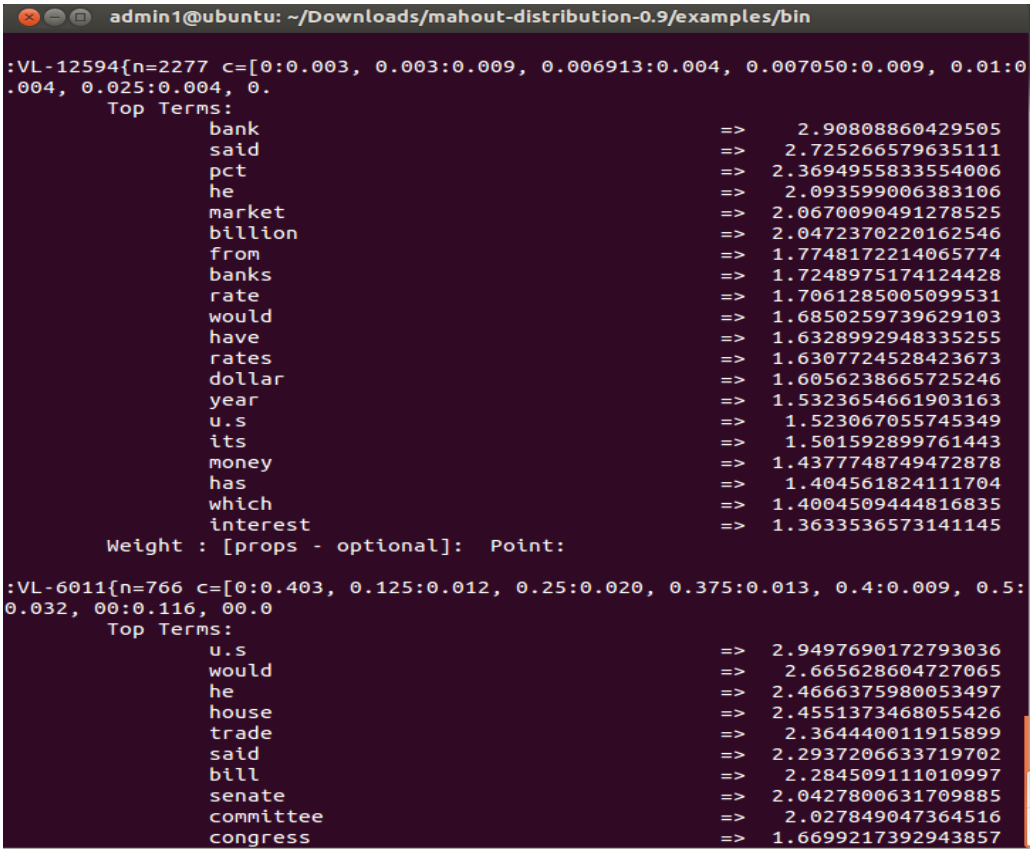

fig 7.12

7.2 FUZZY K-MEANS CLUSTERING

```
admin1@ubuntu:~/Downloads/mahout-distribution-0.9/examples/bin$ ./cluster-reuter
s.sh
Please select a number to choose the corresponding clustering algorithm
1. kmeans clustering
2. fuzzykmeans clustering
3. lda clustering
4. streamingkmeans clustering
Enter your choice : 2
ok. You chose 2 and we'll use fuzzykmeans Clustering
creating work directory at /tmp/mahout-work-admin1
Running on hadoop, using /usr/bin/hadoop-1.2.1/bin/hadoop and HADOOP_CONF_DIR=
MAHOUT-JOB: /home/admin1/Downloads/mahout-distribution-0.9/mahout-examples-0.9-j
ob.jar
14/11/26 23:21:18 INFO vectorizer.SparseVectorsFromSequenceFiles: Maximum n-gram
```

fig 7.13

```
admin1@ubuntu: ~/Downloads/mahout-distribution-0.9/examples/bin
 size is: 1
14/11/26 23:21:18 INFO vectorizer.SparseVectorsFromSequenceFiles: Minimum LLR va
lue: 1.0
14/11/26 23:21:18 INFO vectorizer.SparseVectorsFromSequenceFiles: Number of redu
ce tasks: 1
14/11/26 23:21:18 INFO vectorizer.SparseVectorsFromSequenceFiles: Tokenizing doc
uments in /tmp/mahout-work-admin1/reuters-out-seqdir
14/11/26 23:21:18 INFO util.NativeCodeLoader: Loaded the native-hadoop library
14/11/26 23:21:18 INFO input.FileInputFormat: Total input paths to process : 1
14/11/26 23:21:19 INFO mapred.JobClient: Running job: job_local265953142_0001
14/11/26 23:21:19 INFO mapred.LocalJobRunner: Waiting for map tasks
14/11/26 23:21:19 INFO mapred.LocalJobRunner: Starting task: attempt_local265953
142_0001_m_000000_0
14/11/26 23:21:19 INFO util.ProcessTree: setsid exited with exit code 0
14/11/26 23:21:19 INFO mapred.Task:  Using ResourceCalculatorPlugin : org.apache
.hadoop.util.LinuxResourceCalculatorPlugin@6bb8a3
14/11/26 23:21:19 INFO mapred.MapTask: Processing split: file:/tmp/mahout-work-a
dmin1/reuters-out-seqdir/chunk-0:0+18403626
14/11/26 23:21:20 INFO mapred.JobClient:  map 0% reduce 0%
14/11/26 23:21:22 INFO mapred.Task: Task:attempt_local265953142_0001_m_000000_0
is done. And is in the process of commiting
14/11/26 23:21:22 INFO mapred.LocalJobRunner:
14/11/26 23:21:22 INFO mapred.Task: Task attempt_local265953142_0001_m_000000_0
is allowed to commit now
14/11/26 23:21:22 INFO output.FileOutputCommitter: Saved output of task 'attempt
_local265953142_0001_m_000000_0' to /tmp/mahout-work-admin1/reuters-out-seqdir-s
parse-fkmeans/tokenized-documents
14/11/26 23:21:22 INFO mapred.LocalJobRunner:
14/11/26 23:21:22 INFO mapred.Task: Task 'attempt_local265953142_0001_m_000000_0
' done.
14/11/26 23:21:22 INFO mapred.LocalJobRunner: Finishing task: attempt_local26595
3142_0001_m_000000_0
14/11/26 23:21:22 INFO mapred.LocalJobRunner: Map task executor complete.
14/11/26 23:21:23 INFO mapred.JobClient:  map 100% reduce 0%
14/11/26 23:21:23 INFO mapred.JobClient: Job complete: job_local265953142_0001
14/11/26 23:21:23 INFO mapred.JobClient: Counters: 12
14/11/26 23:21:23 INFO mapred.JobClient:   File Output Format Counters
14/11/26 23:21:23 INFO mapred.JobClient:     Bytes Written=15193772
14/11/26 23:21:23 INFO mapred.JobClient:   File Input Format Counters
```

fig 7.14

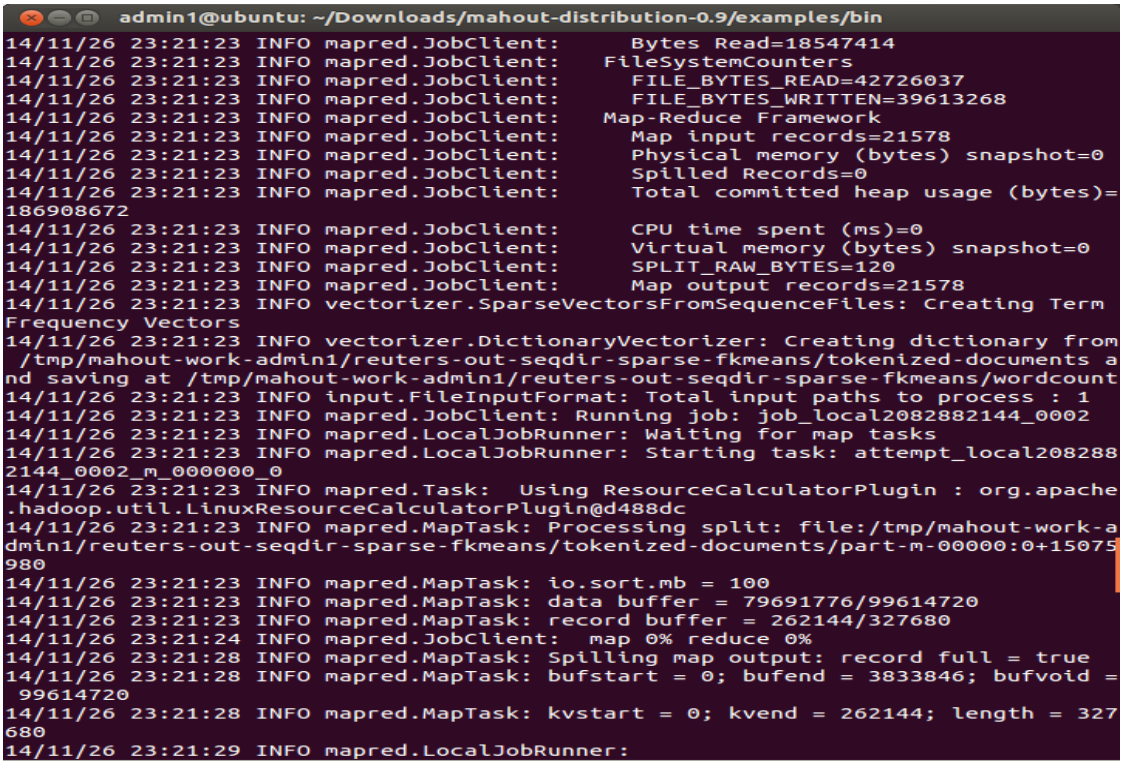

fig 7.15

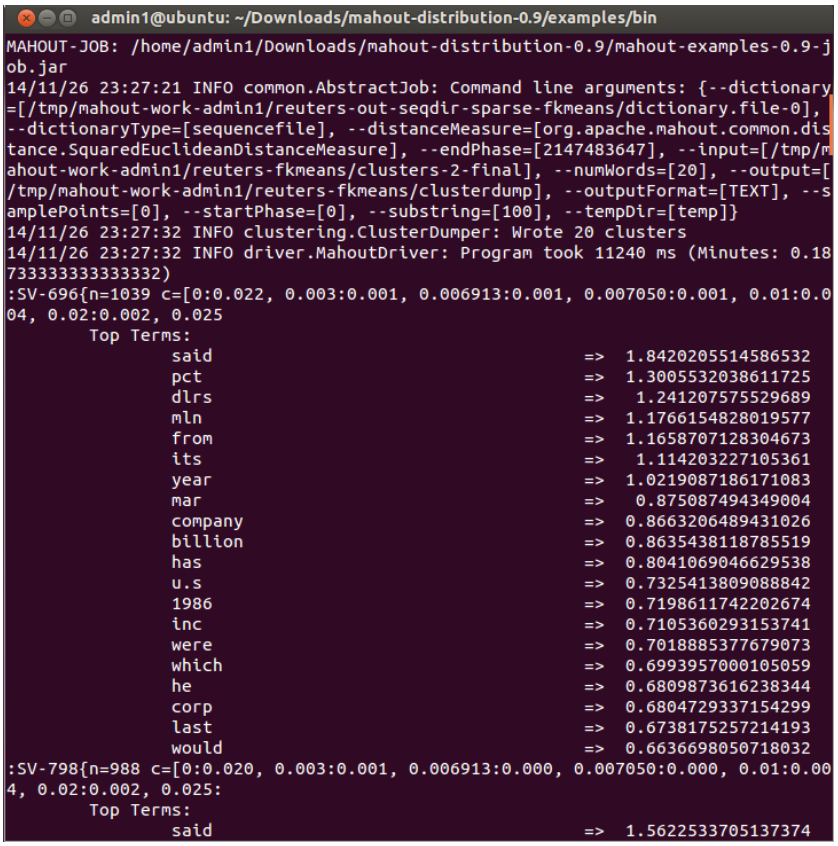

fig 7.16

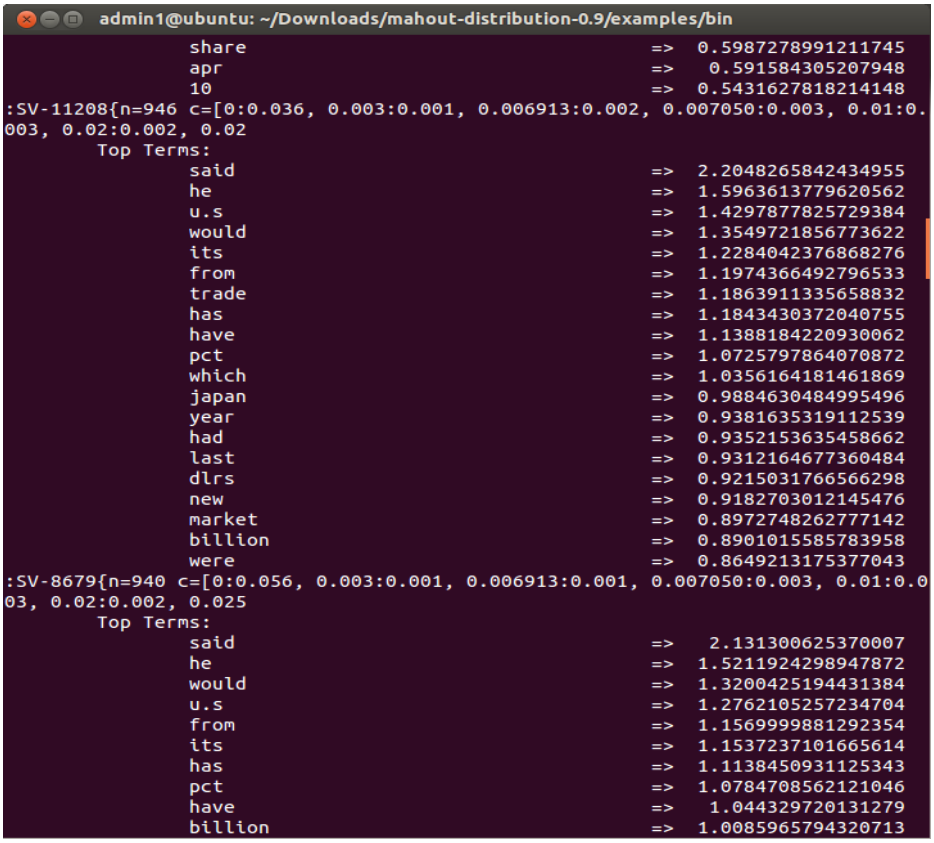

fig 7.17

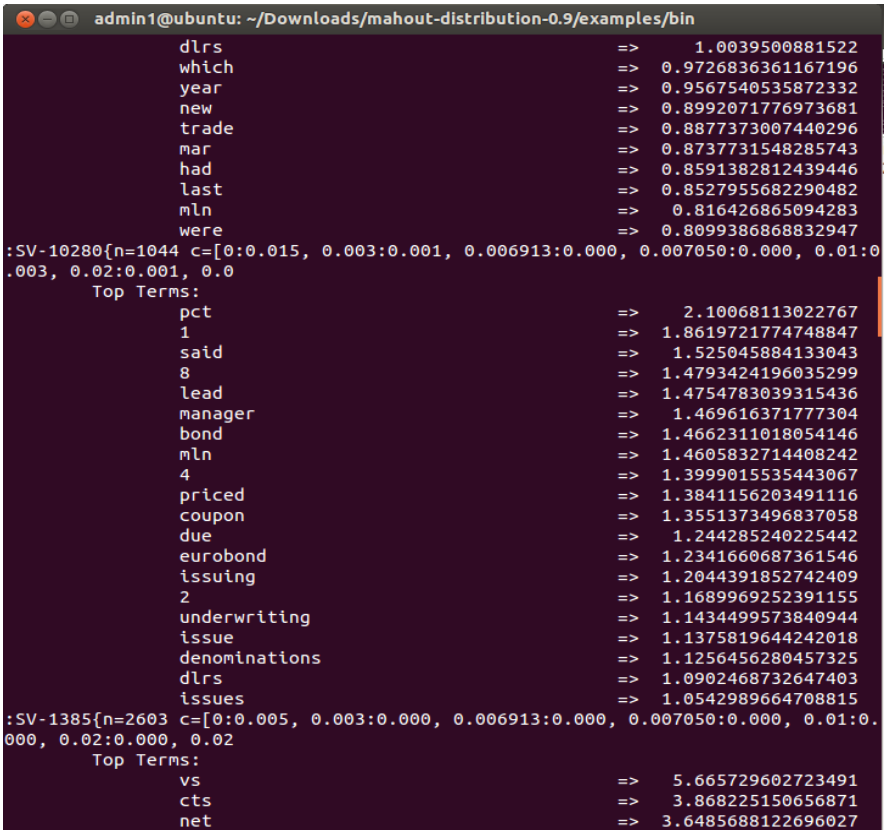

fig 7.18

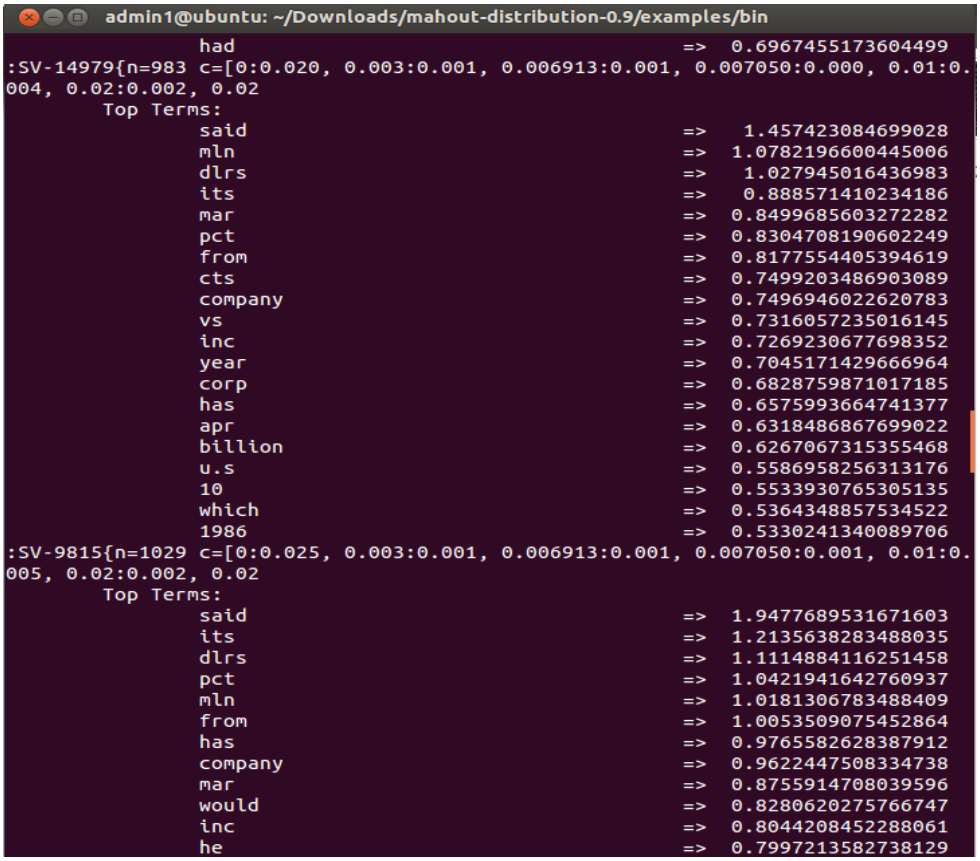

fig 7.19

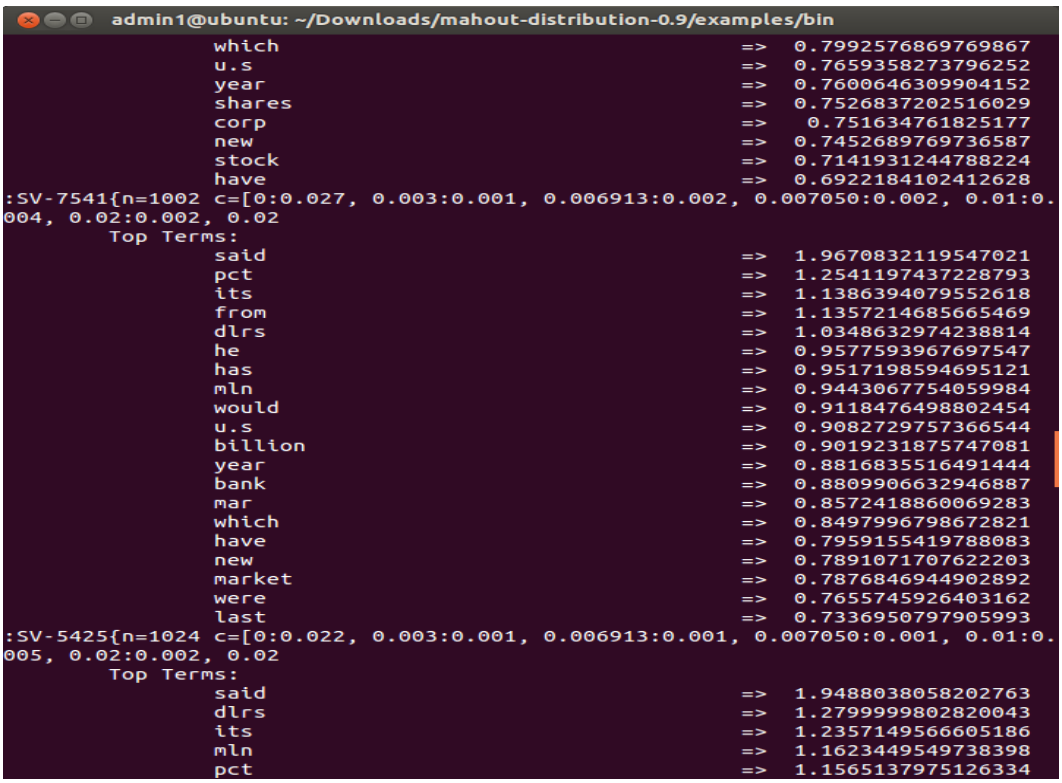

fig 7.20

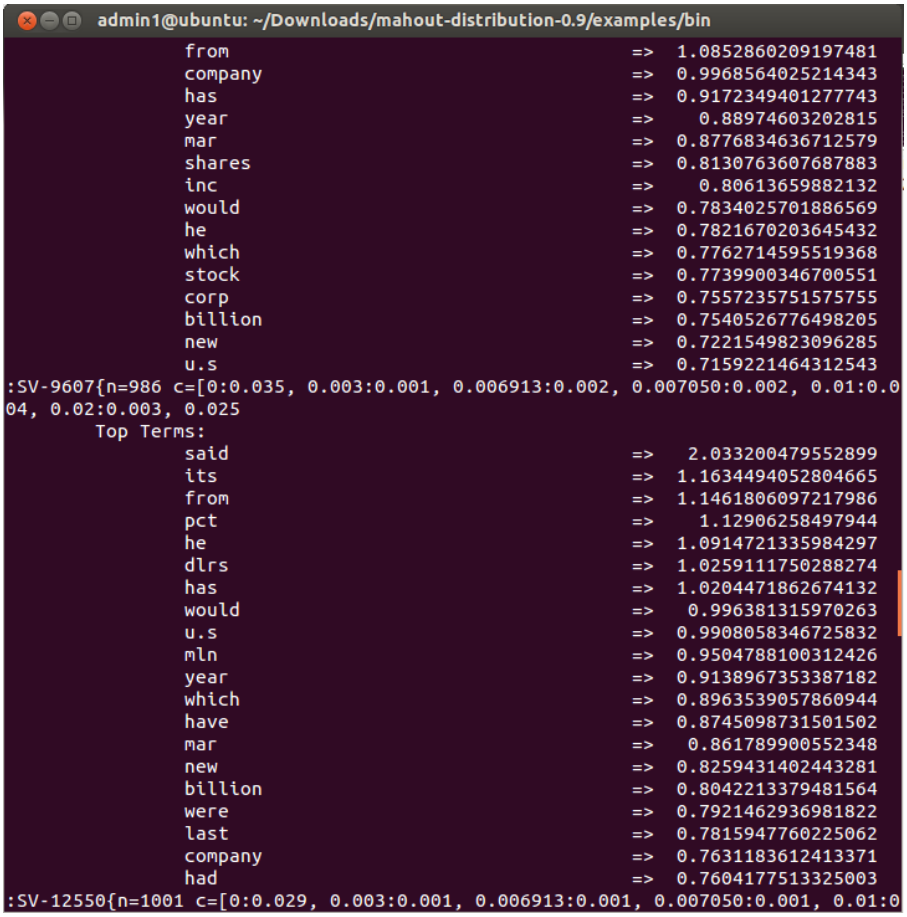

fig 7.21

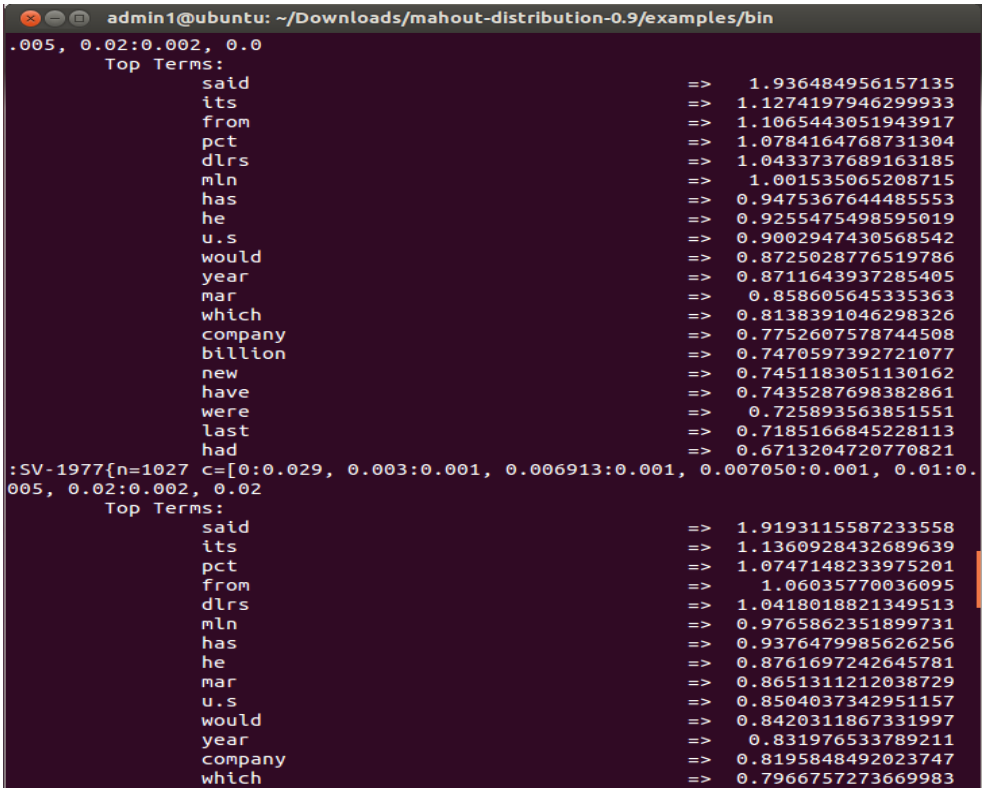

fig 7.22

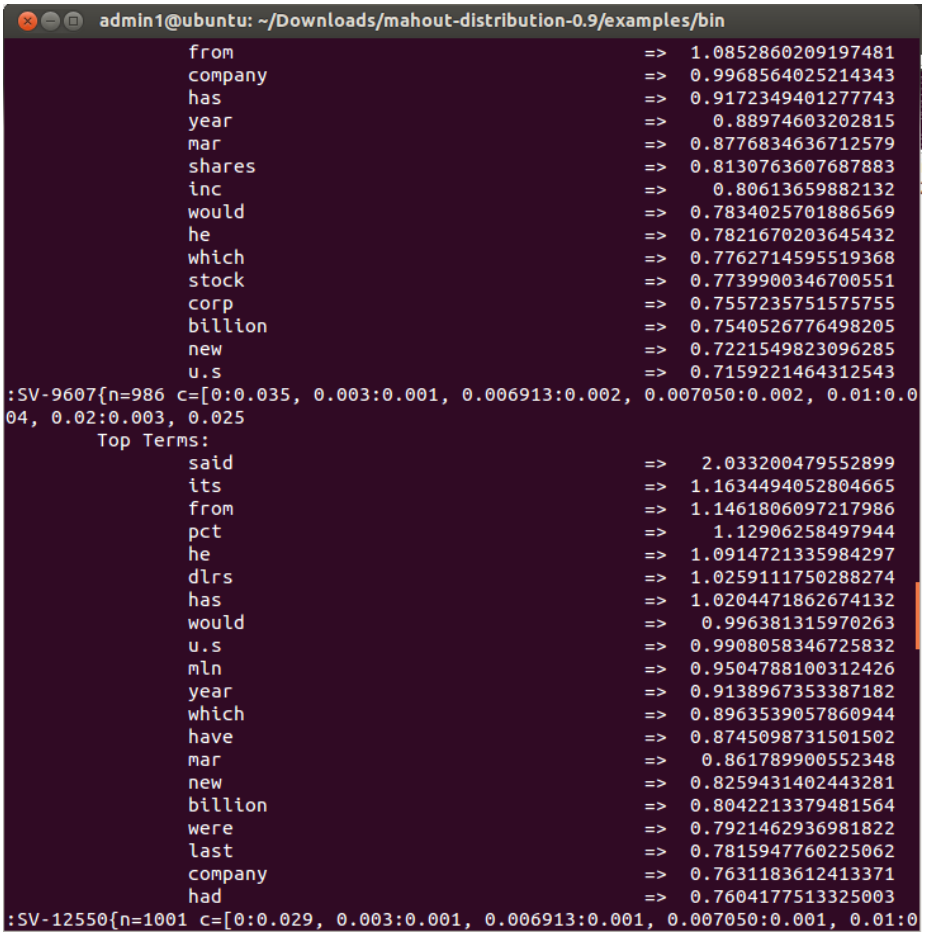

fig 7.23

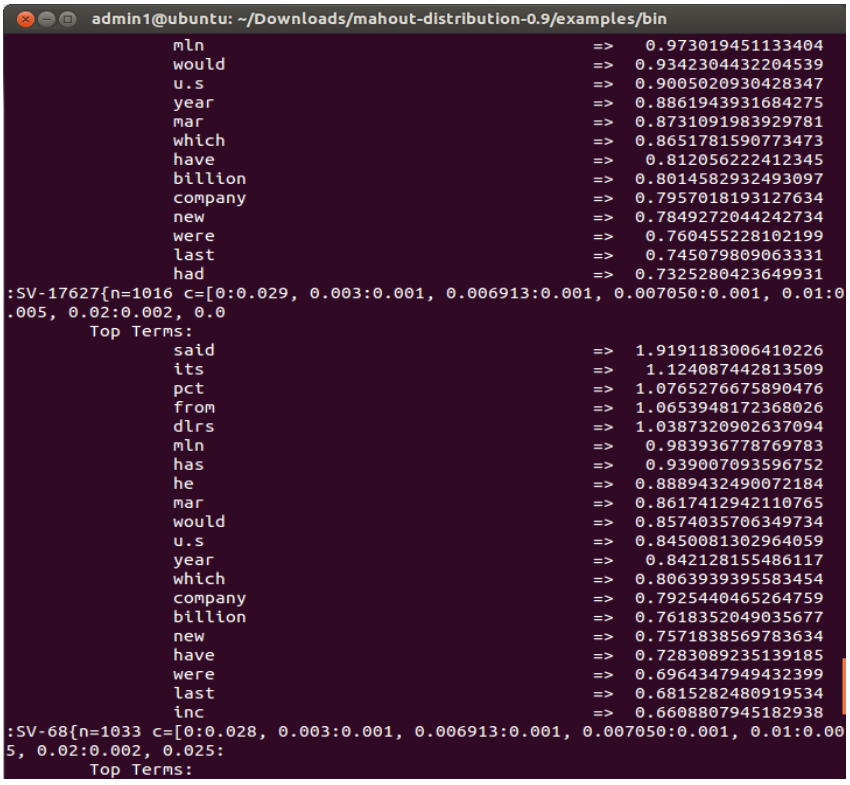

fig 7.24

7.3 STREAMING K-MEANS

```
Please select a number to choose the corresponding clustering algorithm
1. kmeans clustering
2. fuzzykmeans clustering
3. lda clustering
4. streamingkmeans clustering
Enter your choice : 4
ok. You chose 4 and we'll use streamingkmeans Clustering
creating work directory at /tmp/mahout-work-admin1
Downloading Reuters-21578
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 7959k  100 7959k    0     0  28721      0  0:04:43  0:04:43 --:--:-- 30205
Extracting...
Extracting Reuters
Running on hadoop, using /usr/bin/hadoop-1.2.1/bin/hadoop and HADOOP_CONF_DIR=
MAHOUT-JOB: /home/admin1/Downloads/mahout-distribution-0.9/mahout-examples-0.9-j
ob.jar
14/11/27 00:06:02 WARN driver.MahoutDriver: No org.apache.lucene.benchmark.utils
```

fig 7.25

```
admin1@ubuntu: ~/Downloads/mahout-distribution-0.9/examples/bin
Running on hadoop, using /usr/bin/hadoop-1.2.1/bin/hadoop and HADOOP_CONF_DIR=
MAHOUT-JOB: /home/admin1/Downloads/mahout-distribution-0.9/mahout-examples-0.9-j
ob.jar
14/11/27 00:06:02 WARN driver.MahoutDriver: No org.apache.lucene.benchmark.utils
.ExtractReuters.props found on classpath, will use command-line arguments only
Deleting all files in /tmp/mahout-work-admin1/reuters-out-tmp
14/11/27 00:06:07 INFO driver.MahoutDriver: Program took 4879 ms (Minutes: 0.081
31666666666666)
Converting to Sequence Files from Directory
Running on hadoop, using /usr/bin/hadoop-1.2.1/bin/hadoop and HADOOP_CONF_DIR=
MAHOUT-JOB: /home/admin1/Downloads/mahout-distribution-0.9/mahout-examples-0.9-j
ob.jar
14/11/27 00:06:13 INFO common.AbstractJob: Command line arguments: {--charset=[U
TF-8], --chunkSize=[64], --endPhase=[2147483647], --fileFilterClass=[org.apache.
mahout.text.PrefixAdditionFilter], --input=[/tmp/mahout-work-admin1/reuters-out]
, --keyPrefix=[], --method=[sequential], --output=[/tmp/mahout-work-admin1/reute
rs-out-seqdir], --startPhase=[0], --tempDir=[temp]}
14/11/27 00:06:13 INFO util.NativeCodeLoader: Loaded the native-hadoop library
14/11/27 00:06:19 INFO driver.MahoutDriver: Program took 7118 ms (Minutes: 0.118
63333333333333)
Running on hadoop, using /usr/bin/hadoop-1.2.1/bin/hadoop and HADOOP_CONF_DIR=
MAHOUT-JOB: /home/admin1/Downloads/mahout-distribution-0.9/mahout-examples-0.9-j
ob.jar
^Cadmin1@ubuntu:~/Downloads/mahout-distribution-0.9/examples/bin$
```

fig 7.26

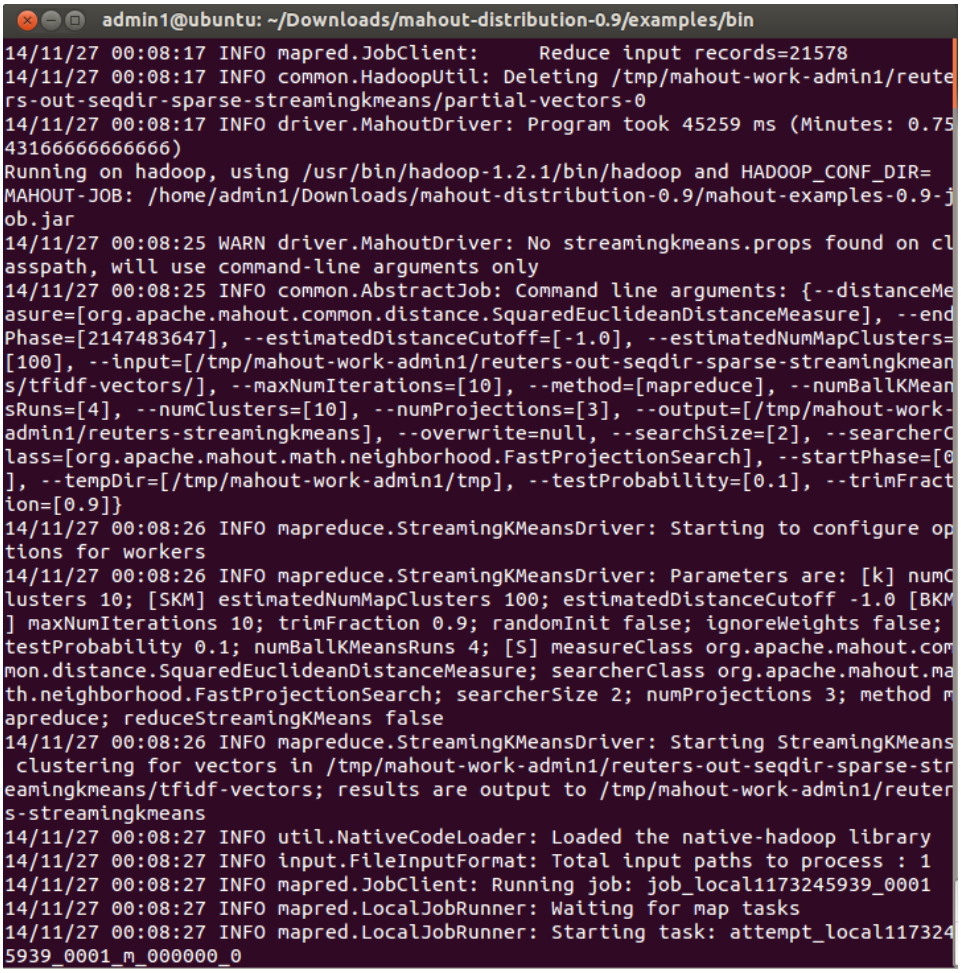

fig 7.27

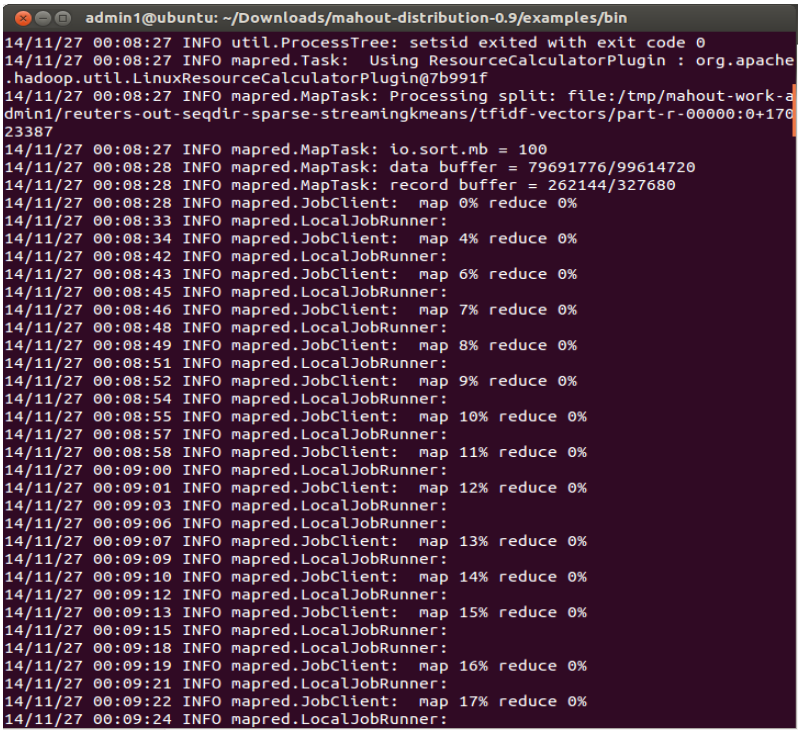

fig 7.28

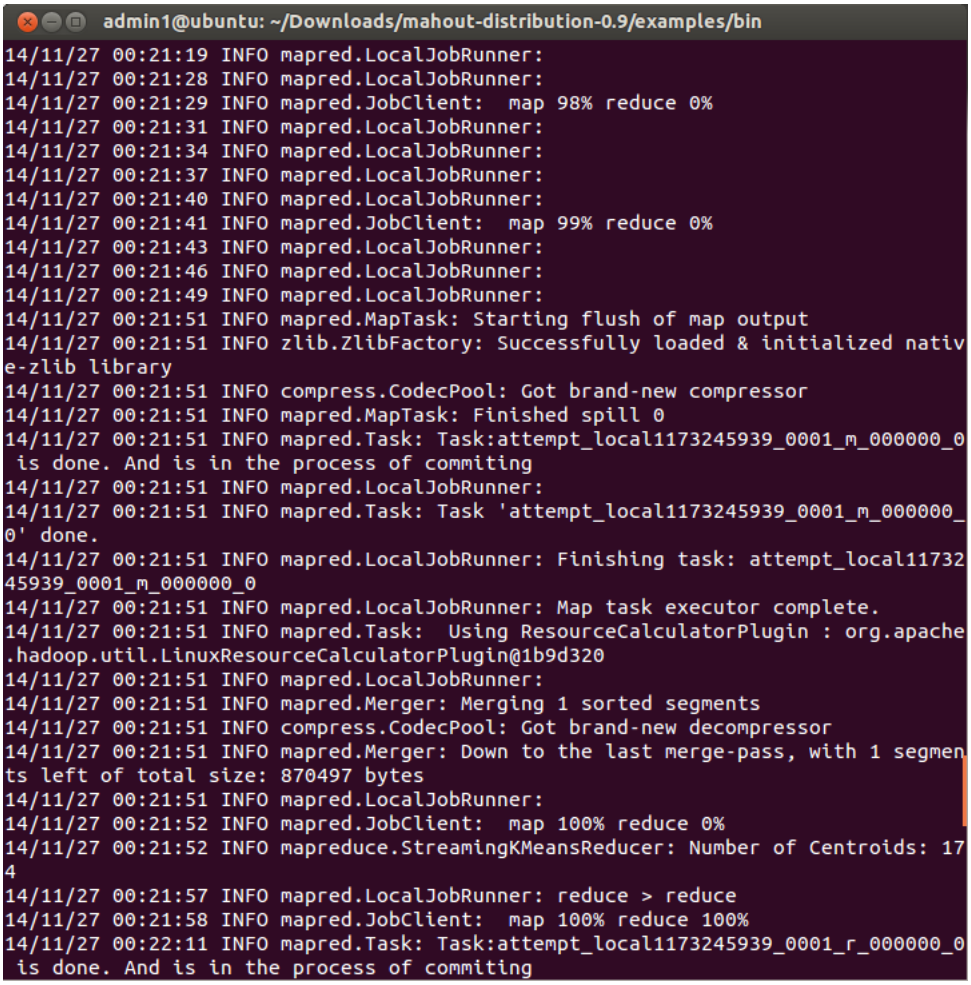

fig 7.29

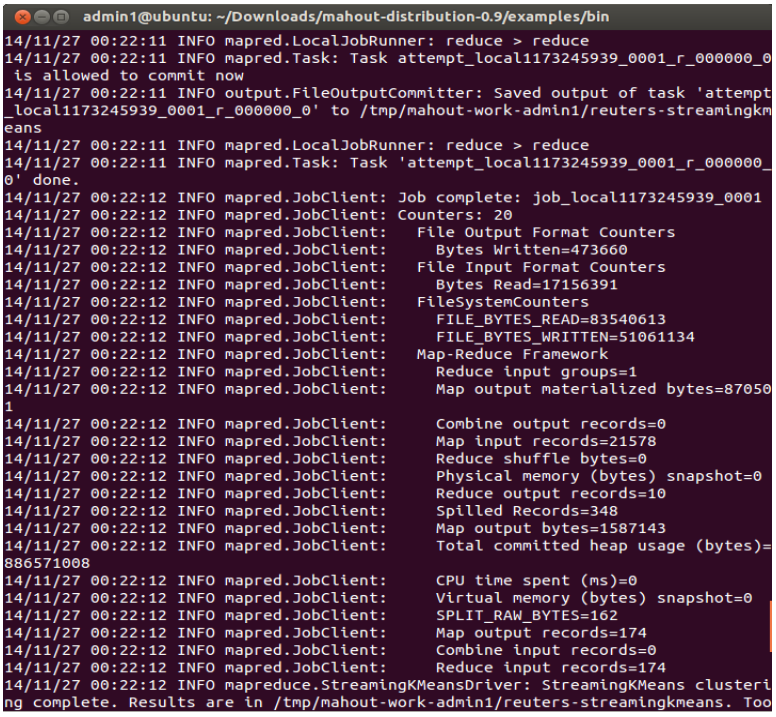

fig 7.30

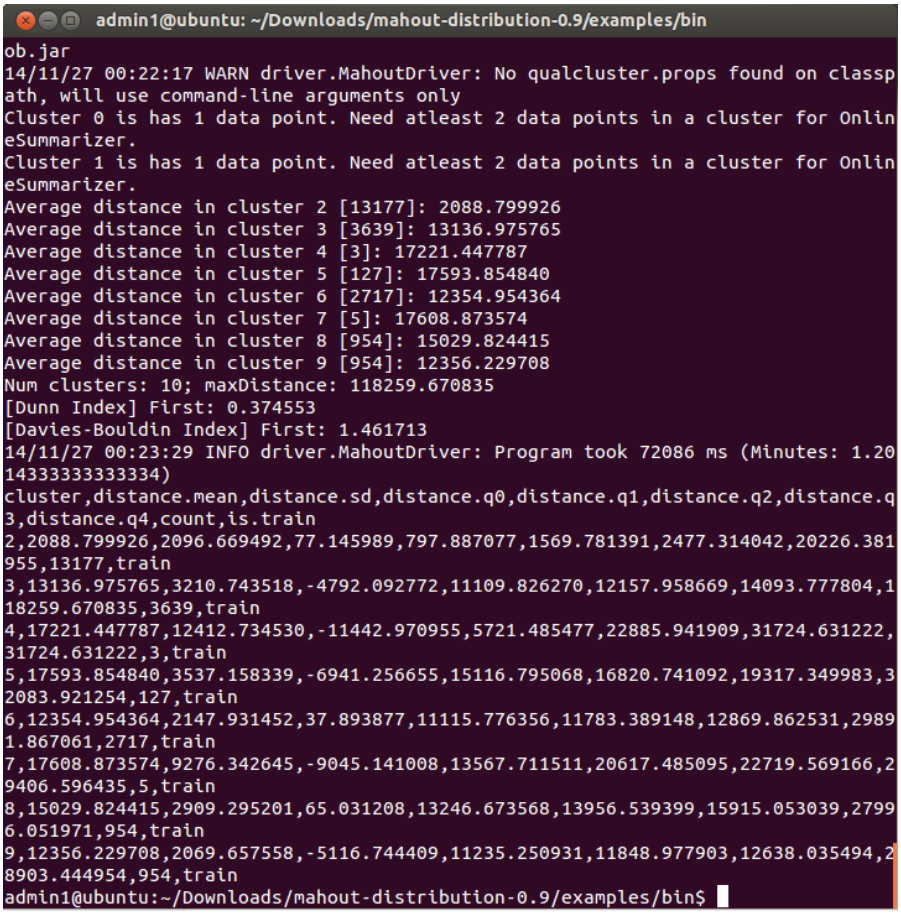
admin1@ubuntu: ~/Downloads/mahout-distribution-0.9/examples/bin
ob.jar
14/11/27 00:22:17 WARN driver.MahoutDriver: No qualcluster.props found on classp
ath, will use command-line arguments only
Cluster 0 is has 1 data point. Need atleast 2 data points in a cluster for Onlin
eSummarizer.
Cluster 1 is has 1 data point. Need atleast 2 data points in a cluster for Onlin
eSummarizer.
Average distance in cluster 2 [13177]: 2088.799926
Average distance in cluster 3 [3639]: 13136.975765
Average distance in cluster 4 [3]: 17221.447787
Average distance in cluster 5 [127]: 17593.854840
Average distance in cluster 6 [2717]: 12354.954364
Average distance in cluster 7 [5]: 17608.873574
Average distance in cluster 8 [954]: 15029.824415
Average distance in cluster 9 [954]: 12356.229708
Num clusters: 10; maxDistance: 118259.670835
[Dunn Index] First: 0.374553
[Davies-Bouldin Index] First: 1.461713
14/11/27 00:23:29 INFO driver.MahoutDriver: Program took 72086 ms (Minutes: 1.20
14333333333334)
cluster,distance.mean,distance.sd,distance.q0,distance.q1,distance.q2,distance.q
3,distance.q4,count,is.train
2,2088.799926,2096.669492,77.145989,797.887077,1569.781391,2477.314042,20226.381
955,13177,train
3,13136.975765,3210.743518,-4792.092772,11109.826270,12157.958669,14093.777804,1
18259.670835,3639,train
4,17221.447787,12412.734530,-11442.970955,5721.485477,22885.941909,31724.631222,
31724.631222,3,train
5,17593.854840,3537.158339,-6941.256655,15116.795068,16820.741092,19317.349983,3
2083.921254,127,train
6,12354.954364,2147.931452,37.893877,11115.776356,11783.389148,12869.862531,2989
1.867061,2717,train
7,17608.873574,9276.342645,-9045.141008,13567.711511,20617.485095,22719.569166,2
9406.596435,5,train
8,15029.824415,2909.295201,65.031208,13246.673568,13956.539399,15915.053039,2799
6.051971,954,train
9,12356.229708,2069.657558,-5116.744409,11235.250931,11848.977903,12638.035494,2
8903.444954,954,train
admin1@ubuntu:~/Downloads/mahout-distribution-0.9/examples/bin$

fig 7.31

8 Conclusion and Future Scope

We can extend this project on larger data sets and can perform distributed computing by creating different nodes.

In addition to, we can perform clustering on dynamic data.

9 BIBLIOGRAPHY

1) www.mahout.apache.org

2)Wikipedia

3)Books:-

a)Mahout in Action by Sean Owen

b)Apache Cookbook by Ken Coar and Rich Bowen