CHAPTER 1

INTRODUCTION

1.1 DATA WAREHOUSE

In computing, a data warehouse (DW) is a database used for reporting and analysis. The data stored in the warehouse is uploaded from the operational systems. The data may pass through an operational data store for additional operations before they are used in the DW for reporting.

A data warehouse maintains its functions in three layers: staging, integration, and access. Staging is used to store raw data for use by developers. The integration layer is used to integrate data and to have a level of abstraction from users. The access layer is for getting data out for users.

This definition of the data warehouse focuses on data storage. The main source of the data is cleaned, transformed, catalogued and made available for use by managers and other business professionals for data mining, online analytical processing, market research and decision support. However, the means to retrieve analyse, extract, transform, load and manage the data dictionary are also considered essential components of a data warehousing system. Many references to data warehousing use this broader context. Thus, an expanded definition for data warehousing includes business intelligence tools, tools to extract, transform and load data into the repository, and tools to manage and retrieve metadata.
A data warehouse maintains a copy of information from the source transaction systems. This architectural complexity provides the opportunity to:

- Maintain data history, even if the source transaction systems do not.
- Integrate data from multiple source systems, enabling a central view across the enterprise. This benefit is always valuable, but particularly so when the organization has grown by merger.
- Improve data, by providing consistent codes and descriptions, flagging or even fixing bad data.
- Present the organization's information consistently.
- Provide a single common data model for all data of interest regardless of the data's source.
- Restructure the data so that it makes sense to the business users.
- Restructure the data so that it delivers excellent query performance, even for complex analytic queries, without impacting the operational system.
- Add value to operational business applications, notably customer relationship management (CRM) systems.

Many organizations have successfully implemented data warehouses to analyse the data contained in their multiple operational systems to compare current and historical values. By doing so, they can better, and more profitably, manage their business, analyse past efforts, and plan for the future. When properly deployed, data warehouses benefit the organization by significantly
enhancing its decision-making capabilities, thus improving both its efficiency and effectiveness.

However, the quality of the decisions that are facilitated by a data warehouse is only as good as the quality of the data contained in the data warehouse – this data must be accurate, consistent, and complete. For example, in order to determine its top ten customers, an organization must be able to aggregate sales across all of its sales channels and business units and recognize when the same customer is identified by multiple names, addresses, or customer numbers. In other words, the data used to determine the top ten customers must be integrated and of high quality. Afterall, if the data is incomplete or incorrect then so will be the results of any analysis performed upon it.

The data warehouse concept originated in an effort to solve data synchronization problems and to resolve data inconsistencies that resulted when analysts acquired data from multiple operational or production systems. One of the most important functions of a data warehouse is to serve as a collection point for consolidating and further distributing data extracts from an organization’s production systems. The data warehouse also must ensure that this data is uniform, accurate, and consistent and thereby it serves as a “single version of truth” for the enterprise.

However, this is much more complicated than it might first appear, especially since each production system is developed to satisfy a particular operational need. Consequently, each application system is designed with its own data standards and thus it is poorly integrated with other systems. This integration is particularly challenging when dealing with legacy systems that
are implemented before any real effort is made to establish enterprise data standards or even common data definitions.

Even if we live in a world with enough disk space and CPU resources to allow time-stamped data values from each transaction associated with every production system to be saved forever, year-end data purges never took place, and computers could quickly read and aggregate all this data for analysis, data warehouses would still be desirable. At a minimum, the data warehouse would be needed to integrate the data in each system and establish a common format. Moreover, all the data that an organization requires for analysis purposes is not stored in its operational systems. Consequently, data warehouses are frequently augmented with data from third-party content providers. This content may, for example, include customer demographics and lifestyle data, credit information, or geographic data used to determine distances from firehouses, telephone company central offices, or even tax jurisdictions. Data warehouses are also likely to contain derived data fields and summary values resulting from the consolidation of data contained in one or more operational systems.

Even when organizations develop data standards, they are unlikely that they modify the existing operational systems to reflect these standards; rather these standards are applied only when they are developing and implementing new systems. Consequently, when the data residing in these operational systems is needed to populate a data warehouse, it is often necessary to first transform the data from each source to be consistent with the enterprise data standards prior to loading it into the data warehouse. The data warehouse is sometimes the first attempt and often the first place that the data actually conforms to corporate standards.
Data integration and data quality are the two key components of a successful data warehouse as both completeness and accuracy of information are of paramount importance. Once this data is collected it can be made available both for direct analysis and for distribution to other, smaller data warehouses.

From a conceptual perspective, data warehouses store snapshots (http://sumit989.wordpress.com/tag/data-warehouse) and aggregations of data collected from a variety of source systems. Data warehouses encompass a variety of subject areas. Each of these source systems could store the same data in different formats, with different editing rules, and different value lists. For example, gender code could be represented in three separate systems as male/female, 0/1, and M/F respectively; dates might be stored in a year/month/day, month/day/year, or day/month/year format. In the United States “03062010” could represent March 6, 2010 while in the United Kingdom it might represent June 3, 2010.

Data warehouses involve a long-term effort and they are usually built in an incremental fashion. In addition to adding new subject areas, at each iteration, the breadth of data content of existing subject areas is usually increased as users expand their analysis and their underlying data requirements. Users and applications can directly use the data warehouse to perform their analysis. Alternately, a subset of the data warehouse data, often relating to a specific line-of-business and/or a specific functional area, can be exported to another, smaller data warehouse, commonly referred to as a data mart. Besides integrating and cleansing an organization’s data for better analysis, one of the benefits of building a data warehouse is that the effort initially spent to populate it with complete and accurate data content further benefits any data marts that are sourced from the data warehouse in Table 1.1.
Table 1.1 An organization’s data warehousing architecture can consist of a variety of components that co-exist and complement each other

<table>
<thead>
<tr>
<th></th>
<th>ODS</th>
<th>Data warehousing</th>
<th>Data mart</th>
<th>EII</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Values</strong></td>
<td>Current</td>
<td>Historic, possibly current</td>
<td>Historic, possibly current</td>
<td>Current</td>
</tr>
<tr>
<td><strong>Rigorous data quality</strong></td>
<td>Usually</td>
<td>Yes</td>
<td>Dependent: Yes Independent: Unlikely</td>
<td>Unlikely</td>
</tr>
<tr>
<td><strong>Primary use</strong></td>
<td>Operational &amp; Tactical</td>
<td>Tactical and Strategic</td>
<td>Tactical and Strategic</td>
<td>Operational &amp; Tactical</td>
</tr>
<tr>
<td><strong>Organizational scope</strong></td>
<td>Functional or Line-of-Business</td>
<td>Enterprise</td>
<td>Functional or Line-of-Business</td>
<td>Depends on source</td>
</tr>
<tr>
<td><strong>Implementation timeframe</strong></td>
<td>Intermediate-term</td>
<td>Long Term</td>
<td>Dependent: Short term Independent: Intermediate term</td>
<td>Short term</td>
</tr>
<tr>
<td><strong>Level of detail</strong></td>
<td>Detailed</td>
<td>Detailed &amp; Summary</td>
<td>Detailed &amp; Summary</td>
<td>Depends on source</td>
</tr>
<tr>
<td><strong>Data volatility</strong></td>
<td>High-values added</td>
<td>Low-values added</td>
<td>Low-values added</td>
<td>Depends on source</td>
</tr>
<tr>
<td><strong>Relationship to other variants</strong></td>
<td>Complementary</td>
<td>Complementary</td>
<td>Complementary</td>
<td>Complementary</td>
</tr>
</tbody>
</table>
1.2 DATA MARTS

A data mart Michael (2013), if populated from a data warehouse, contains a subset of the data from the data warehouse. If this is the case, then it is generally considered to be a dependent data mart and can be implemented relatively quickly as the data has already been collected and integrated within data warehouse. The quality of its content is directly dependent upon the contents of the data warehouse. Independent data marts are those that are developed without regard to overall data warehouse architecture, perhaps at the departmental or line-of-business level, typically for use as a temporary solution.

As the independent data mart cannot rely on an existing data warehouse for its content, implementation will take longer than a dependent data mart. Regardless of a data mart operating independently of any other data mart or data warehouse, it is still important that the data within it be complete and accurate. If not, erroneous analysis is likely to occur and invalid conclusions are drawn.

Pragmatically, an independent data mart may be the only viable approach when the existing enterprise warehouse is being built incrementally and the data needed by the data mart users is not yet available from the warehouse.

Building a corporate data warehouse on a “subject by subject” approach is certainly a reasonable and proved strategy. Many organizations that have tried to populate their enterprise data warehouses with data for all requested subject areas prior to initial rollout have found that this is akin to attempting to trying to “boil the ocean,” the task is simply too overwhelming to be realistically accomplished in anything other than a phased approach.
It is reasonable to assume that an organization’s independent data marts will ultimately be combined. Eventually they will lose their independence as individual data needs are ultimately satisfied through an enterprise data warehouse.

Combining the content requirements of these independent data marts to determine the contents of the enterprise data warehouse will be significantly easier if each data mart contains high quality and complete data. This “bottoms up” approach of using the requirements of existing independent data marts to determine the requirements of a data warehouse from which they will be populated has been effective in organizations where several departments first needed to quickly implement their own solutions. These organizations could simply not wait for their “top down” data warehouse to first be built.

A common problem that exists in many organizations is the inability to quickly combine operational data about the same entity such as a customer or vendor that exists in multiple systems. A classic example occurred when banking institutions first started adding new service offerings such as investment accounts to their more traditional savings and checking account offerings. Many of these new services were supported by systems that existed independently. When the bank needed to see all the current financial information it had about a customer, it needed to combine and consolidate data from all these systems, assuming of course it could identify that a customer whose account information resided in several systems, was the same customer. As this need became increasingly more important, the operational data store (ODS) came into vogue.

A primary difference between data warehouses and operational data stores is that while a data warehouse frequently contains multiple time-stamped historical data snapshots, with new snapshots being added on a well-defined periodic schedule, an operational data store contains current values that are
continually in flux. A data warehouse adds new time-stamped data values and retains the old ones; an operational data store updates the existing data values. While the initial load and continual updating of the operational data store are classic examples of data integration, the ability to identify and link different accounts each captured from a different system, as belonging to the same customer is also a classic example of data quality. This underscores the importance of, and interdependence between, data quality and data integration, while solving real-world business problems.

1.3 ENTERPRISE INFORMATION INTEGRATION (EII)

While not necessarily a new concept Michael (2013), the idea of enterprise information integration, or EII, has received much publicity in the past few years. Simply stated, it involves quickly bringing together data from multiple sources for analysis purposes without necessarily storing it in a separate database. Some vendors even have gone so far as to claim that an EII approach can replace a traditional data warehouse or data mart with a “virtual data warehouse” by eliminating the need to extract and store the data into another database. However, the ramifications associated with this approach (e.g., such as the underlying data changing, or even being purged, between analysis) must be not be overlooked.

However, an EII solution complements the other data warehousing variants and can be a valuable resource for those wishing to perform quick, perhaps ad hoc, analysis on current data values residing in operational systems. It can help alleviate the backlog of requests which are a constant struggle for any IT staff.

However, organizations must recognize that the data in these operational systems may not be consistent with each other, that the data quality of each source may vary widely, and that historical values may not be available.
This is a risk and many users are willing to take for “quick and dirty” analysis when the needed data is not contained in a formal data warehouse or data mart. In fact, many organizations use an EII approach to establish processes and programming logic that enable their users to transform and pull together data from multiple sources for purposes that include desktop analysis. EII is at its best when the quality of the data in the underlying operational systems is high.

EII solutions may also be successfully used to prototype or evaluate additional subject areas for possible inclusion in a data warehouse. Some organizations have initially deployed EII solutions when the data warehouses or data marts did not contain the needed data and later added this data to their data warehouse content. In order to combine current and historical values, organizations can include an existing data warehouse or data mart as one of their sources and thus combine historical and current data values.

1.4 DATA INTEGRATION AND DATA QUALITY

While estimates vary Michael (2013), it is generally agreed that data integration and data quality represent the majority of the cost of implementing a data warehouse. Lack of data integration and poor data quality are the most common causes of post-implementation data warehouse failures. First impressions count; if decision-makers find that the data warehouse contains incomplete or incorrect data, they are likely to seek their data elsewhere.

At a very simple level, data integration and data quality are concerned with collecting accurate data, transforming it into a common format with a common set of values, providing appropriate aggregations or summary tables, and loading it into the data warehouse environment. This sounds simple enough but there are many complicating factors that must be considered.
1.5 MULTIPLE DATA SOURCE

The requisite data is likely to be stored in a variety of systems Michael (2013), in a variety of formats, and on a variety of platforms. Assuming the required data resides in a computer system, and not on paper in a file cabinet, the data sources may be relational databases, XML documents, legacy data structures (such as Adabas, IDMS, IMS, VSAM, or even sequential EBCDIC files). It may be contained in packaged enterprise application systems such as SAP or Siebel where knowledge of the business logic is necessary to understand and access the underlying data. The data may reside on a wide variety of computing platforms including mainframe, Unix, Windows, and Linux environments.

1.6 DATA TRANSFORMATION

The detailed data residing in the operational systems must frequently be consolidated in order to generate and store Michael (2013), for example, daily sales by product by retail store, rather than storing the individual line-item detail for each cash register transaction.

Complex arithmetic or statistical calculations frequently need to be applied to the source data in order to perform, for example, percent of sales calculations or “top x” rankings, especially if items that of low value are grouped into an “others” category before being loaded into the warehouse.

1.7 DATA LINKING

In many cases data records relating to the same object Michael (2013) (for example, a given customer, employee, or product), resides in multiple source systems. These records must first be linked together and consolidated prior to being loaded into the data warehouse. Integration with
data quality software is often the only realistic way of matching these records, especially when trying to deal with the nuances involved identifying customers or vendors. Each system may contain its own variation, in format or spelling, of the same customer name and address. Once again, data quality software can greatly facilitate this task.

1.8 DATA VOLUMES

It is frequently necessary to load very large data volumes into the warehouse in a short amount of time Michael (2013), thereby requiring a parallel processing and memory-based processing. While the initial data loads are usually the most voluminous, organizations have a relatively long load window in which to accomplish this task since the initial load is done prior to the data warehouse being opened for production. After the data warehouse is in use, new data content must be loaded on a periodic basis. The load volume can be reduced if change data capture techniques are employed to capture only data that has changed since the prior data load. In some cases, Enterprise Application Integration (EAI) technology, frequently involving message queues, can be used to link enterprise applications to the data warehouse data integration processes in order to capture new data on a near-real-time basis.

1.9 METADATA INTEGRATION

Most data integration tools store the metadata (or data about data) associated with its sources and targets in a metadata repository that is included with the product Michael (2013). At a minimum, this metadata includes information such as source and target data formats, transformation rules, business processes concerned with data flows from the production systems to the data warehouse (i.e., data lineage), and the formulae for computing the values of any derived data fields. While this metadata is needed by the data integration tool for use in defining and creating appropriate data transformation
processes, its value is enhanced when shared with other tools utilized in designing the data warehouse tables and business intelligence tools that access the warehouse data. If the metadata also includes information about what analysis program uses which data element, it can be a valuable source for analysing the ramifications of any change to the data element (i.e., impact analysis).

1.10 ADDITIONAL THOUGHTS ON DATA QUALITY

Data quality is involved throughout the entire data warehousing environment and is an integral part of the data integration process Michael (2013). Data quality involves ensuring the accuracy, timeliness, completeness, and consistency of the data used by an organization and also it makes sure that all parties utilizing the data have a common understanding of what the data represents. For example, does sales data include or exclude internal sales and is it measured in units or dollars, or perhaps even Euros? In most data warehousing implementations data quality is applied in at least two phases. The first phase is concerned with ensuring that the source systems themselves contain high quality data while the second phase is concerned with ensuring that the data extracted from these sources can then be combined and loaded into the data warehouse. As mentioned earlier, even if the data residing in each of the sources is already accurate and clean, it is not simply a matter of directly combining the individual sources as the data in each source could exist in a different format and use a different value list or code set. One system may use the alphabetic codes (S,M,D) to represent “single,” “married,” and “divorced” and another may represent them with the numeric codes (1, 2, 3). The data loaded into the warehouse must conform to a single set of values; data cleansing and data transformation technology must work together to ensure that they do. Of course, duplicate occurrences of the same customer or vendor across multiple systems, or even in the same system, with different variations
of the same name and/or address, is a well-known example of a data quality issue that was previously discussed.

As more decisions are distributed across the organizational hierarchy and as more information is processed and exposed to end-consumers, there is a growing need for visual representations of many aspects of business transactions, such as type, time, duration, and critically, business transaction locations etc. Enhancing the value and utility of business information for both operational and analytical applications requires a combination of sound data management practices, such as data governance, data quality assurance, data enhancement, and increasingly, location intelligence capabilities, geo-coding, mapping, routing etc. Corporate mergers and acquisitions, changes in regulatory compliance requirements, increased customer attrition, noticeable increase in call centre activity, system migrations, improvements to customer-facing business processes, introduction or CRM or ERP, or committing to MDM are all examples of business events that depend on organizational adjustments in data management practices in order to succeed.

Each of these events demonstrates a need for instituting solid business processes and the corresponding information management practices to improve the customer experience while simultaneously maintaining financial system of scale. As opposed to blindly throwing technology, perhaps multiple times, at a presumed problem and hoping that something will stick, a more thoughtful approach seeks a return on the technology investment that is achieved when the necessary information management practices are supported with core data services implemented once but deployed multiple times in a consistent way.

The techniques employed in data warehouse development have become ubiquitous, especially as the need for system inter-operability has grown. As more people want to re-engineer their environments to allow
sharing, or even consolidation of siloed data sets, there is a growing need for the underlying data integration services to provide the bridging ability to seamlessly access data from many different sources and deliver that data to numerous targets. The ability for remote servers to access data from their original sources supports the efficient implementation and deployment of data management best practices that can be implemented using the core data services as illustrated.

Integration services rely on the ability to connect to commonly-used infrastructures and data environments coupled with the parsing and standardization services for transforming extracted data into formats that are suitable for data delivery. And in the modern enterprise, data sharing, federation, and synchronization is rapidly becoming the standard. As more organizations transition to a services oriented architecture, seamless data integration becomes a fundamental building block for enterprise applications.

1.11 TRENDS IN DATA WAREHOUSING

Several trends are developing in the data warehouse market, many of which are directly concerned with data integration and data quality. These include:

- EAI and ETL, will continue to converge due to the need to update the data warehouse with the recent transactions.
- The use of “active” data warehouses that directly feed analytical results back to operational systems will grow
- Pragmatic hybrid approaches to data warehousing will continue to win-out over insistence on architectural purity
• Data quality gains additional recognition as an up-front requirement for both operational and analytical systems efforts, rather than an after-the-fact fix.

• EII is recognized as a complementary to and not a replacement for traditional data warehouses and data marts.

• Disparate data integration tools will give way to end-to-end data integration platforms with end-to-end data integration functionality

• Data integration platforms, callable both through direct application programming interfaces and as Web services will also tap into leading-edge features, such as appliances, open source, cloud computing etc.

1.12 DUPLICATE DETECTION

Data de-duplication is a term used to illustrate an algorithm, method or technique that eliminates duplicate entries of data to a storage system Ananthakrishna et al (2002), Chuanyi et al (2008). Data De - Duplication (often called "intelligent compression" or "single-instance storage") is a method of reducing storage needs by eliminating redundant data. Only one unique instance of the data is actually retained on storage media such as disk or tape. Redundant data is replaced with a pointer to the unique data copy.

De-duplication is a crucial step in the data cleaning Bohannon (2007), Chaudhuri et al (2003) storage processes. The process to identify the duplicates will be recorded in storage system. The advancement of computer technology data are stored in digitized form. The amount of data keeps on increasing every year in all types of storage data. However, there may be a necessity to store a huge amount of duplicate data in storage system. Collective
de-duplication is a generalization in which one requires to find types of real-world entities in a set of records that are already stored and they are exactly the same content and also when merging two datasets from two different sources system or two different platforms, there is often a problem with overlapping among the records. Finding these duplicate records can be challenging since the format of the data is often different between databases.

Duplicate data, in the domain of computer data management, is any data that is stored at more than one location in a computer system, including duplicate data stored on the same computer, in the same network, or even in external memory. This is true not only of data that is stored two or more times in its entirety, but also partially repeated data. A classic example is data records (stored in a database) that contain identical or partially overlapping information. The duplicate data problem follows from the extreme storage capabilities of modern computers, wherein a commodity hard disk can easily store the equivalent of thousands of encyclopaedia volumes. Among such large quantities of data, it is commonplace for unrecognized duplicates to be present. This duplicate data problem is a current, critical, and yet unsolved, problem in database systems. Duplicate data has a negative impact on data storage and retrieval, system execution performance, data quality, and the fundamental usefulness of the data. For example, in a relational database, duplicate data causes unnecessary bloating of the data set, consuming secondary storage resources and requiring search queries to take extra, unwarranted CPU cycles. We may note that duplication can improve access time, for example, if a record is stored under two different keys or in multiple tables. However, despite this speed increase, duplicate data tends to cause problems when updating database entries. Moreover, duplicate data records are often incomplete, with only the union of all overlapping records providing a complete understanding of the data element. Thus, duplicate data reduces the efficacy of a database system as a whole.
There is a need for effective and automatic de-duplication of data records, and it is a challenging and multi-faceted problem that requires: (1) automatic detection of two overlapping records, and (2) merging of partial information from each record to form a new, more complete record. Both steps are difficult due to the varied nature of data to which de-duplication could be applied. While existing approaches, as detailed in Chapter 2, present various alternatives for dealing with data de-duplication, none of them have been able to propose a solution that successfully resolves this problem in its entirety. When others perform important work on data de-duplication, as detailed in the Previous Work section, there are no sufficient solutions in existing technique for dis-similarity calculation.

This thesis presents a robust, flexible, and automated solution for duplicate record detection based on rule-based data de-duplication that utilizes learning-based information fusion, specifically in kernel-machine-based learning. Our novel method applies individual de-duplication rules to data record fields, and then it combines the resultant match scores via learning-based information fusion into one overall match score. This aggregate match score effectively determines whether two data records represent the same entity. Our learning-based fusion design (1) alleviates the need for manual tuning of the de-duplication rules, (2) provides robust detection of overlapping records beyond what individual rules can provide, and (3) is highly reliable when faced with partial or missing record data.

Unlike the existing de-duplication approaches, our proposed technique has a number of key strengths, which include: (1) automatic and rapid tuning of the de-duplication rules, (2) robust detection of overlapping records that is an order of magnitude more accurate than traditional methods, and (3) reliability when faced with partial or missing record data.
This novel solution represents a significant step forward in the state-of-the-art for automated duplicate record detection. We demonstrate this fused, learning-based de-duplication on a significant real-world schema.

Rule-based de-duplication suffers from two primary weaknesses: (1) effective identification of duplicate data requires significant manual tuning of the de-duplication rules, and (2) poor duplicate detection occurs when there are missing data values. This thesis consists of two complementary techniques that address these weaknesses. First one is grouping mechanism and second one is dis-similarity calculation.

1.13 CLUSTERING

Cluster analysis or clustering is the task of assigning a set of most closely related objects into groups (called clusters) so that the objects in the same cluster are more similar (in some sense or another) to each other than to those in other clusters.

Clustering (http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/) can be considered the most important unsupervised learning problem; and so, as every other problem of this kind, it deals with finding a structure in a collection of un-labeled data. A loose definition of clustering could be “the process of organizing objects into groups whose members are similar in some way”. A cluster is therefore a collection of objects which are “similar” among them and are “dissimilar” to the objects belonging to other clusters.
In this case (Figure 1.1: Simple graphical representation of Clustering) we easily identify the 4 clusters into which the data can be divided; the similarity criterion is distance: two or more objects belong to the same cluster if they are “close” according to a given distance (in this case geometrical distance). This is called distance-based clustering. Another kind of clustering is conceptual clustering: two or more objects belong to the same cluster if this one defines a concept common to all that objects. In other words, objects are grouped according to their fit to descriptive concepts and not according to simple similarity measures.

Goals of Clustering: Clustering is a main task of explorative data mining, and a common technique for statistical data analysis used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics.

So, the goal of clustering is to determine the intrinsic grouping in a set of un-labeled data. But how to decide what constitutes a good clustering? It can be shown that there is no absolute “best” criterion which would be independent of the final aim of the clustering. Consequently, it is the user who must supply this criterion, in such a way that the result of the clustering will suit
their needs. For instance, we may be interested in finding representatives for homogeneous groups (data reduction), in finding “natural clusters” and describe their unknown properties (“natural” data types), in finding useful and suitable groupings (“useful” data classes) or in finding unusual data objects (outlier detection).

Cluster analysis Bolshakova et al (2005) itself is not an algorithm but the general task is to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with low distances among the cluster members, dense areas of the data space and multivariate normal distributions. The appropriate clustering algorithm and parameter settings (including values such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery that involves try and failure.

Clustering algorithms can be applied in many fields, for instance:

- Marketing: finding groups of customers type with similar behavior given a large database of customer data containing their properties and past buying records and its details;
- Biology: classification of plants and animals given their features and behavior;
• Insurance: identifying groups of motor insurance policy holders type with a high average claim cost and low average claim cost; to identify frauds;

• City-planning: identifying groups of houses according to their house type, value and geographical location;

• Earthquake studies: clustering observed earthquake epicenters to identify dangerous zones in the earth;

• WWW: document classification; clustering weblog data to discover groups of similar access patterns.

The main requirements that a clustering algorithm should satisfy are:

• scalability;

• dealing with different types of attributes;

• discovering clusters with arbitrary shape;

• minimal requirements for domain knowledge to determine input parameters;

• ability to deal with noise and outliers;

• insensitivity to order of input records;

• high dimensionality;

• interpretability and usability

There are a number of problems with clustering. Among them:
• current clustering techniques do not address all the requirements adequately (and concurrently);

• dealing with large number of dimensions and large number of data items can be problematic because of time complexity;

• the effectiveness of the method depends on the definition of “distance” (for distance-based clustering);

• if an obvious distance measure does not exist we must “define” it, which is not always easy, especially in multi-dimensional spaces;

• the result of the clustering algorithm (that in many cases can be arbitrary itself) can be interpreted in different ways.

Clustering algorithms may be classified as listed below:

Exclusive Clustering: This separation is based on the characteristic that allows a data object to exist 1 or more than 1 clusters. Exclusive clustering is as the name suggests and stipulates that each data object can only exist in one cluster. Below (Figure 1.2: Exclusive Clustering) is an example as each object is only a member of one cluster. Figure 1.3 (to the right) is another example of clustering.

![Figure 1.2 Exclusive Clustering](image)

Overlapping Clustering: It (Shown in the Figure 1.3: Overlapping Clustering) allows data objects to be grouped in 2 or more clusters. A real world example would be the breakdown of personnel at a school. Overlapping
clustering would allow a student to also be grouped as an employee while exclusive clustering would demand that the person must choose the one that is more important. In Fuzzy clustering every data object belongs to every cluster, and we may guess that fuzzy clustering may be described as an extreme version of overlapping, in which the major difference is that the data objects has a membership weight that is between 0 and 1 where 0 means it does not belong to a given cluster and 1 means it absolutely belongs to the cluster. Fuzzy clustering is also known as probabilistic clustering.

![Figure 1.3 Overlapping Clustering](image)

Hierarchical Clustering: Hierarchical clustering algorithm has two versions: agglomerative clustering and divisive clustering. Agglomerative clustering is based on the union between the two nearest clusters. The beginning condition is realized by setting every datum as a cluster. After a few iterations it reaches the final clusters wanted. Basically, this is a bottom-up version. Divisive clustering starts from one cluster containing all data items. At each step, clusters are successively split into smaller clusters according to some dissimilarity. Basically this is a top-down version.

Probabilistic Clustering: Probabilistic clustering is an iterative soft clustering technique in which the cluster memberships of a data point are based on the distances (typically Euclidean) from the cluster centers. According to Israel and Iyigun (2008) in probabilistic (distance) clustering, “Given clusters, their centers and the distances of data points from these centers, the probability of cluster membership at any point is assumed inversely proportional to the
distance from (the center of) the cluster in question.” In this iterative approach, the cluster centers are updated as convex combinations of data points and this continues until the centers stop changing. This approach is similar to k-means, however, in this case the cluster assignment is “soft” and probabilities of cluster membership for each data point (i.e., consumer) are calculated. This method is considered to be robust, insensitive to outliers and works best when cluster sizes are about equal. In SAS Global Forum Dipanjan et al (2011) an SAS macro was published that discussed a way of implementing probabilistic clustering technique in SAS Enterprise Miner. This macro utilizes the distances calculated by the k-means algorithm to calculate cluster membership probabilities. We used this macro in order to segment the customers using Probabilistic clustering technique.

**Most used clustering algorithms**

K-means : K-means Macqueen (1967) is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. This procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a cunning way because of different location causes different results. So, the best choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenter of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated.
As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more.

Assume we have n sample feature vectors \( x_1, x_2, \ldots, x_n \) all from the same class, and we know that they fall into k compact clusters, \( k < n \).

Let \( m_i \) be the mean of the vectors in cluster i. If the clusters are well separated, we can use a minimum-distance classifier to separate them. That is, we can say that \( x \) is in cluster i if \( ||x - m_i|| \) is the minimum of all the \( k \) distances. This suggests the following procedure for finding the k means:

Make initial guesses for the means \( m_1, m_2, \ldots, m_k \)

Until there are no changes in any mean

Use the estimated means to classify the samples into clusters

For i from 1 to k

Replace \( m_i \) with the mean of all of the samples for cluster i

end_for

end_until

Fuzzy C-means: Fuzzy c-means (FCM) is a method of clustering which allows one piece of data to belong to two or more clusters. This method (developed by Dunn in 1973 and improved by Bezdek in 1981) is frequently used in pattern recognition. It is based on minimization of the following objective function:

\[
J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^m \left\| x_i - c_j \right\|^2, \quad 1 \leq m < \infty
\]
where m is any real number greater than 1, \( u_{ij} \) is the degree of membership of \( x_i \) in the cluster \( j \), \( x_i \) is the \( i \)th of \( d \)-dimensional measured data, \( c_j \) is the \( d \)-dimension centre of the cluster, and \( \|*\| \) is any norm expressing the similarity between any measured data and the centre. data are bound to each cluster by means of a Membership Function, which represents the fuzzy behaviour of this algorithm. To do that, we simply have to build an appropriate matrix named \( U \) whose factors are numbers between 0 and 1, and represent the degree of membership between data and centers of clusters.

Hierarchical clustering: Given a set of \( N \) items to be clustered, and an \( N*N \) distance (or similarity) matrix, the basic process of hierarchical clustering Johnson (1967) is this:

1. Start by assigning each item to a cluster, so that if you have \( N \) items, you now have \( N \) clusters, each containing just one item. Let the distances (similarities) between the clusters be the same as the distances (similarities) between the items they contain.

2. Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one cluster less.

3. Compute the distances (similarities) between the new cluster and each of the old clusters.

4. Repeat steps 2 and 3 until all items are clustered into a single cluster of size \( N \). (*)

   Step 3 can be done in different ways, and they distinguishes single-linkage from complete-linkage and average-linkage clustering. In single-linkage clustering (also called the connectedness or minimum method), we consider the distance between one cluster and another cluster to be equal to the shortest distance from any member of one cluster to any member of the
other cluster. If the data consist of similarities, we consider the similarity between one cluster and another cluster to be equal to the greatest similarity from any member of one cluster to any member of the other cluster. In complete-linkage clustering (also called the diameter or maximum method), we consider the distance between one cluster and another cluster to be equal to the greatest distance from any member of one cluster to any member of the other cluster. In average-linkage clustering, we consider the distance between one cluster and another cluster to be equal to the average distance from any member of one cluster to any member of the other cluster. A variation on average-linkage clustering is the UCLUS method of D'Andrade (1978) which uses the median distance, which is much more outlier-proof than the average distance.

The algorithm is composed of the following steps:

1. Begin with the disjoint clustering having level \( L(0) = 0 \) and sequence number \( m = 0 \).

2. Find the least dissimilar pair of clusters in the current clustering, say pair \((r), (s)\), according to

\[
d[(r),(s)] = \min d[(i),(j)]
\]

where the minimum is over all pairs of clusters in the current clustering.

3. Increment the sequence number : \( m = m + 1 \). Merge clusters \((r)\) and \((s)\) into a single cluster to form the next clustering \( m \). Set the level of this clustering to

\[
L(m) = d[(r),(s)]
\]
4. Update the proximity matrix, D, by deleting the rows and columns corresponding to clusters \((r)\) and \((s)\) and adding a row and column corresponding to the newly formed cluster. The proximity between the new cluster, denoted \((r,s)\) and old cluster \((k)\) is defined in this way:

\[
d[(k), (r,s)] = \min \ d[(k),(r)], \ d[(k),(s)]
\]

5. If all objects are in one cluster, stop. Else, go to step 2.

Mixture of Gaussians: There is another way to deal with clustering problems: a model-based approach, which consists of using certain models for clusters and attempting to optimize the fit between the data and the model. In practice, each cluster can be mathematically represented by a parametric distribution, like a Gaussian (continuous) or a Poisson (discrete). The entire data set is therefore modelled by a mixture of these distributions. An individual distribution used to model a specific cluster is often referred to as a component distribution.

A mixture model with high likelihood tends to have the following traits:

- component distributions have high “peaks” (data in one cluster are tight);
- the mixture model “covers” the data well (dominant patterns in the data are captured by component distributions).
- Main advantages of model-based clustering:
  - well-studied statistical inference techniques available;
  - flexibility in choosing the component distribution;
  - obtain a density estimation for each cluster;
• a “soft” classification is available.

1.14 CLASSIFICATION

In machine learning, statistical classification is the problem of identifying the sub-population to which new observations belong and where the identity of the sub-population is unknown, on the basis of a training set of data containing observations whose sub-population is known. Therefore these classifications will show a variable behavior which can be studied by statistics.

Thus the requirement Breiman (1993) is that new individual items are placed into groups based on quantitative information on one or more measurements, traits or characteristics and they are based on the training set in which previously decided groupings are already established.

The problem here may be contrasted with that of cluster analysis, where the problem is to analyse a single data-set and decide how and whether the observations in the data-set can be divided into groups. In certain terminology, particularly that of machine learning, the classification problem is known as supervised learning, while clustering is known as unsupervised learning.

Topological / Back-End Classification This classification takes the back-end constellation of the data warehouses into consideration, i.e., the data source side. There are two classes:

Single-source data warehouses: A data warehouse belongs to this class if it has only one source application. This class contains the simplest data warehouses of all. As there is only one operational application delivering data to the data warehouse, nothing has to be done for the issue of data integration, which is in general a challenging task when constructing data warehouses. On
the other hand, the data has to be transformed more or less to meet the representation requirements set by the analysis needs. There are several special cases encountered quite frequently in data engineering. If the system neither transforms nor collects the data, (i.e., the data it keeps is only a replica of that from the source application), this system is not a data warehouse, even if the data is used for analysis. If the system transforms but does not collect the data at all, (i.e., absolutely no history of the operational data from the source application is maintained there), the system is not a data warehouse. If the system collects but it does not transform the data, (i.e., a history of the operational data is maintained online for analysis purposes), then the system is not a data warehouse and it is an operational data store.

Today, many standard operational applications have their own data warehouse extensions for subsequent analysis of the executed operations Bin (2012). All of them can be considered as single-source data warehouses.

Multi-source data warehouses: A data warehouse belongs to this class if it is not a single-source warehouse. When talking about data warehouses, we mean mostly multi-source warehouses since one of the most important functionalities a data warehouse provides is data integration. From an evolutionary perspective, the single-source warehouse represents the primary phase of the data warehousing initiative of the organization. If the job is well done, the mature and final stadium is a multi-source data warehouse. In other words, the evolution is unidirectional, from single-source to multi-source.

Organizational / Front-End Classification : This classification is based on the front-end constellation of the data warehouses, i.e., the end user side. There are two classes:
Departmental data warehouses: If the data warehouse is utilized mainly by a part of the organization, it is a departmental data warehouse. In data warehousing literature, it is also called a data mart. Within a big organization, there can be many independent departmental data warehouses. They are used, owned and supervised independently by the respective “departments.” One of the major issues with maintaining multiple departmental data warehouses within the organization is the inconsistency of the data from different data warehouses. The other is the high total cost of ownership of all these data warehouses.

Enterprise data warehouses: If the data warehouse is employed by the whole organization for analysis purposes, it is an enterprise data warehouse. The enterprise data warehouse represents the target stadium of the data warehouse evolution within the organization. The following stages are typical along the unidirectional evolution path within an organization:

1. A lot of departmental data warehouses exist to meet the departmental analysis needs.

2. The enterprise data warehouse appears and coexists with the departmental data warehouses to solve the “single version of the truth” and “total cost of ownership” issues previously mentioned.

3. The enterprise data warehouse exists alone within the organization.

Since the enterprise data warehouse is of strategic importance for the organization, it must meet a lot of fundamental requirements.
Temporal / Freshness Classification: This classification takes the state of the data warehouse content into account, i.e., their change modes. There are two classes:

Periodical data warehouses: If updating the data warehouse content is carried out periodically – for example, daily or weekly – it is a periodical data warehouse. The updating is independent of the source data generation and the period is defined beforehand in accordance with the business needs. Data warehouses, whose updating is triggered by the system administrator, belong to this class as well. In principle, this can also be called demand-driven updating.

Real-time data warehouses: If the content of the data warehouse is updated very shortly after the source data is generated by the operational applications – for instance, if it is updated only a few minutes after the source data generation or only after the generation of a small, defined number of data rows – it is a real-time, or more accurately, a near real-time data warehouse. The content updates of these data warehouses are driven by the source data generation; in other words, generation-driven updating. The data freshness of data warehouses in this class is crucial for the near real-time business enabling the so-called tactical decisions.

More often, a data warehouse begins its life with demand-driven updating. Success stories and effectiveness attract more users, who in turn generate more demands and set higher data freshness requirements. This eventually leads to a complete real-time data warehouse. During the evolution, there may be a long time with a mixture of updating modes, partially periodically and partially real-time.

Geographical / Location Classification: This classification considers the geographical location modes of the data warehouses. There are two classes:
Distributed data warehouses. If the major data objects of the data warehouse – or parts of them – are logically based on the same model but stored and processed physically at different geographical locations, it is a distributed data warehouse. If the underlying data models are different and independent of each other, the data warehouses in consideration are departmental. This class of data warehouses is especially meaningful for big international organizations.

Centralized data warehouses: If all data objects of the data warehouse are processed and stored physically at one geographical location, it is a centralized data warehouse.

Classification and prediction methods can be compared and evaluate according to the following criteria http://cs.uiuc.edu/class/fa05/cs412/chaps/6.pdf):

- **Accuracy:** The accuracy of a classifier refers to the ability of a given classifier to correctly predict the class label of new or previously unseen data. Similarly, the accuracy of a predictor refers to how well a given predictor can guess the value of the predicted attribute for new or previously unseen data. Estimation techniques are cross-validation and bootstrapping, because the accuracy computed is only an estimate of how well the classifier or predictor will do on new data tuples, confidence limits can be computed to help gauge this estimate.

- **Speed:** This refers to the computational costs involved in generating and using the given classifier or predictor.

- **Robustness:** This is the ability of the classifier or predictor to make correct predictions given noisy data or data with missing values.
Scalability: This refers to the ability to construct the classifier or predictor efficiently given large amounts of data.

Interpretability: This refers to the level of understanding and insight that is provided by the classifier or predictor. Interpretability is subjective and therefore more difficult to assess.

A very brief description of all the methods included in this classification (Geographical / Location Classification) is presented here just providing the minimum information to make the final choice. Below may be referred for more detailed discussion about the technical assumptions on data for correct application on every technique.

- Conceptual clustering: Provides grouping of homogeneous objects. Requires hypothesis about the number of classes to be found. Results are directly understandable. Usually do not work with very big data sets.

- Statistical clustering: Provides grouping of homogeneous objects. May not require the number of classes. Can be efficient with big data sets. Sometimes difficult to understand the meaning of grouping provided.

- Clustering based on rules: Provides grouping of homogeneous objects. Do not require number of classes as input. Can introduce prior expert knowledge as semantic bias. Guarantee interpretability of results and coherence with prior expert knowledge.

- Association rules: Provides patterns of associated values of variables and frequencies of appearance. Interpretable results.
• Model-based reasoning: Provides formal model of the causal relationships among the domain variables, by providing models for the dependencies among variables.

• Qualitative reasoning: Provides qualitative model of the causal relationships among the domain variables, by representing which variables increase or decrease values as a consequence of modifications in the values of other variables.

• Principal component analysis: Provides graphical representation to see numerical variables which behave associated or not. Extra work is required to interpret results.

• Simple correspondence analysis: Provides graphical representation to see modalities of two qualitative variables which behave associated or not. Extra work is required to interpret results.

• Multiple correspondence analysis: Provides graphical representation for associations among modalities of various qualitative variables. Extra work is required for interpretation.

• Bayesian networks: Provides graphical interpretation of causal relationships between variables together with conditional probabilities.

• Instance-based learning: Uses historical data to classify a new instance of a problem in a predefined set of classes.

• Rule-based classifiers: Provide a set of classification rules that can be used later to evaluate a new case and classify in a predefined set of classes.
- Decision trees: Provide a graphical representation of a tree with conditions associated to nodes that permit to classify a new instance in a predefined set of classes. Problems are there with very big data sets. It works with qualitative variables.

- Discriminant analysis: Provides an algebraic discriminant function and a cut-off as the rule to decide between two groups for a new instance. Only for numerical variables, two predefined classes and they work only under linear separably classes.

- Support Vector Machines (SVM): They can provide discriminant functions to distinguish between two predefined classes that can be non-lineary separable.

- Boxplot-based induction rules: Provide a set of probabilistic classification rules that can be used later to classify a new instance in a predefined set of classes.

- Regression-trees: Provide decision trees for prediction of numerical values. Each leaf has a numerical value, which is the average of all the training set values that the leaf, or rule, applies to.

- Model trees: Provide regression trees combined with regression equations. The leaves of these trees contain regression equations rather than single predicted values. A model tree approximates continuous functions by several linear sub models.

- Naïve Bayes classifier: Provides an adaptative classifier that can improve initial knowledge-based predictions for the class of a new instance by refining the model on the basis of the evidences provided by the whole history of processed cases.
• Connexionist models: Include all artificial neural networks models. Permit to predict the value of one or more variables for a new instance on the basis of non-linear combination of the values of several input variables and intermediary layers.

• Evolutionary computation: Provides the optimization of a certain objective function through the evolution of a population of individuals, who are subjected to several genetic operators. Include techniques simulating the theory of evolution, like genetic algorithms and genetic programming.

• Swarm Intelligence (SI): Provides predictions of numerical variables by training the system under the metaphore of the collective behavior of decentralized, self-organized systems, natural or artificial. Local interactions among very simple agents lead to the emergence of intelligent global behavior. Natural examples of SI include ant colonies, bird flocking, animal herding, bacterial growth, and fish schooling.

• Simple linear regression: Predicts the value of a quantitative variable for a new instance as a linear equation of a single numerical variable. Requires normality, linearity and homocedasticity.

• Multiple linear regression: Predicts the value of a quantitative variable for a new instance as a linear equation of several numerical variables. Requires normality, linearity, homocedasticity and independence.

• Analysis of Variance: Predicts the value of a quantitative variable for a new instance as a linear combination of one or two
qualitative variables. Requires conditional normality, linearity, homoscedasticity and independence.

- Generalized Linear Models: Predicts the value of a quantitative variable for a new instance as a linear combination of several numerical and qualitative variables. Same hypothesis is as that of the previous methods.

- Time series: Predict the value of a quantitative variable for a future instance as a linear combination of past values of the same variable.

1.15 MOTIVATION

It is observed that duplicate detection algorithm makes use of two approaches to find a dis-similarity calculation in optimal solution, namely duplicate detection with grouping based approach on entropy and duplicate detection without grouping based on entropy approach.

1.16 BRIEF OUTLINE OF THE THESIS

This thesis is organized as follows: Chapter 2 defines many of the terms used in this thesis and discusses other related work. Chapter 3 describes the data sets used for analysis and the algorithms developed to evaluate token-based de-duplication. Chapter 4 explains the implementation of the best algorithm in a primary storage system. Section 5 discusses the experimental setup, workloads, and results from the implementation. Chapter 6 lists all the compromises that were made in my implementation, summarizes the work presented in this thesis and concludes.