Improvements in Visual Analytics

Ansh Bahri
University of Southern California
abahri@usc.edu

Salman Aleid
University of Southern California
aleid@usc.edu

Gaurav Mathur
University of Southern California
gmathur@usc.edu

Abhishek Pimpale
University of Southern California
apimpale@usc.edu

ABSTRACT
Visual Analytics as the name suggests is the representation of analytical reasoning using various visual interactive interfaces making it easier for the users to perceive silent aspects from the data quickly. In this paper we try to tackle some of the concerns regarding data enrichment, integration and cleaning.

1. INTRODUCTION
An integration of new computation and theory based tools leading to visual representations to further improve the user's ability to better read data and make decisions regarding it is what data analytics does. The fusion of various interdisciplinary fields such as visual and data representations, analytical reasoning techniques and theories of visualization leads to the field of Visual Analytics.

There are some data analysis tool which already exists for such purposes like Tableau Public [1]. Although the system enable users to visually explore the data without actually needing to learn any programming language, these systems are observed to have some limitations which still need to be improved upon.

These systems need a data set to work upon. Not all data sets are clean and in a well-structured format. The cleaning of the data sets needs to be integrated with its visualization. The need to identify and enrich relevant data sets should be handled by the Visualization System, without concerning the users. Integration of various data sets from different sources could lead to missing or redundant data. The capable system should be able to detect such abnormalities and act in a rational way and even modify the data set if required.

All the above mentioned capabilities- cleaning, detecting, integrating and updating the data; should take place in a single framework in an integrated manner.

2. PROBLEM DEFINITION
In this paper we basically focus on improving the following four aspects in the context of data visualization

2.1 Combined Data Visualization and cleaning
Data exploration and cleaning can help reveal outliers that correspond to errors which effect the data visualization. Changes are required to be made over such data sets by the system. Some of the options for fixing such a problem could be finding additional data sets to integrate, perform entity resolution over the data or even manually fix the problem. Deciding what needs to be done is an issue. A lot of resources are wasted upon cleaning fragments of the data set which are not relevant to the required visualization. Such fragments need to be identified keeping in mind that the managing and cleaning actions are shared among various users.

2.2 Data Enrichment
The user's job is to find useful information in the form of data sets and feed it to the system to perform its visualizations. Then he/she needs to check if the data can be integrated successfully with the other data sets over many users. How to do the above efficiently is a problem of concern keeping in mind the large size of information generated and available for the user.[1]

2.3 Seamless Data Integration
Often referred to as data blending, data integration is the ability of the system to integrate data without causing any disturbance to the data representation. The system should be capable of using prior visualizations by the same or other users and be able to recommend data and alterations within the data set during other visualizations interaction. Inconsistent representation of the same entity across data sets can compromise the visual integrity and its decision making ability.

2.4 Common Formalism
There should be such a system where the interaction by the users on the data set and the visualizations should result in deciding semantics without the user needing to authorize. There exist no such software to support the complete analysis cycle consisting of cleaning, integration and collaboration from GUI. The need for a common centralized system with a formal language is required to integrate the system over the vast data available and different interfaces used by the user.

3. CHALLENGE
Finding ways to optimize existing visual analytical systems to increase the overall efficiency and make it more accurate for a better user experience.

4. POSSIBLE SOLUTION
Techniques to generate annotations to help detect and explain outliers and trends in point based visualization. Prioritizing the cleaning of the data records that impact the visualization more can help in increasing the efficiency. Systems should be developed to be robust to manage conflicting updates with global sharing by various users. Recommendations to identify useful data set and alterations

over it that are more relevant and can be successfully integrated. To improve the above suggestions, clues should be picked up from the current data. New data integration tasks should be determined with the help of previous relevant visualization actions from the user.

5. SURVEY

The survey of the four aspects has been done below. These aspects are individual research topics in their own right. Here we make an attempt to provide an overview of the existing techniques in these areas and how they relate to our problem.

5.1 Data Cleaning

Data cleaning involves getting rid of different kinds of errors in the data set. Some of the common errors include duplicate entries, incorrect format of a particular field, incorrect addition of an entry and incomplete entries. As a result, a generalized and efficient method for data cleaning is very difficult to come up with. Still researchers have come up with many great approaches to solve this problem.

One of the widely used approaches for removing duplicate entries are similarity join and similarity search [8]. Similarity join takes two sets of objects and an acceptable error range to output a pair of points, each belonging to one of the input sets, such that the error, defined by a distance function, between them is less than the acceptable value [9]. Thus similarity join is frequently used to remove duplicate–looking entries. Similarity search is another popular approach for removing duplicate entries. It returns all the objects in a collection that are similar to a query object [8]. These methods use a filter and search approach. In the first step they efficiently filter out objects which are not similar and then verify the remaining objects for actual similarity. The primary filtering technique used in the first step is prefix filtering. Even though the method works, the length of prefixes have an effect on the performance. Longer prefix lengths result in better filtering but long filtering time and shorter ones result in less filtering time but longer verification time. Thus, similarity join and similarity search give good results but take a long time for huge data sets. The primary objective of the application is improving data visualization for users and long intervals of data cleaning would be pretty irritating. Also, these approaches only take care of removing duplicates and don't particularly help with other problems.

Another approach for data cleaning is maintaining data consistency. In this approach we define a set of integrity constraints like functional dependencies, inclusion dependencies and denial constraints for the schema [10]. Then we check whether all our data tuples satisfy all these constraints. Any tuples which don't satisfy these constraints are removed. To improve the results from this approach a variant of this approach which uses conditional functional dependencies instead of the traditional functional dependencies has also been implemented [11]. Though this is a great method for cleaning the data set and coming up a consistent data set, it is impractical for our application. The greatest problem with this approach is that we don't have a set schema for our data. This happens because we take data from various sources and all the sources don't have the same schema. Also, we can't expect the user to come up with a unified schema which fits the data from all the sources. Thus to apply this approach we would have to analyze the data and come up with a unified schema which fits the data from all the sources. This is a time consuming endeavor which would again result in user dissatisfaction.

There are other data cleaning solutions available but none of them is a tailor made fit for our application as our data sets are huge and using any of these solution would be very time consuming. To solve this problem we will have to come up with a hybrid approach which doesn't take a lot of time resulting in an ergonomic application. As a possible solution we are thinking of cleaning data based on the user's priority. For example if a user would like to focus on a particular visualization which uses only a part of the data set or only a part of the attributes then we could concentrate our efforts only on cleaning that part on high priority. In the mean time we could use a technique which uses lower processing power behind the scenes to clean the data. Thus the user receives accurate visualizations and also doesn't see any visible drop in the performance of the application.

5.2 Data Visualization and Data Enrichment

Visual analytics gives a way to combine the enormous storage and processing capabilities of the current technology with the flexibility, creativity and background knowledge of the decision makers to gain insight into complex problems. The transformation from confirmatory data analysis to exploratory data analysis is the most important step. The need for involving the user in KDD (Knowledge Discovery in Databases) process providing better interaction capabilities and knowledge transfer can be integrated using visualization techniques.

Presenting the data efficiently and using this to effectively communicate the results of an analysis. The analysis should be from goal oriented examination of hypothesis. The data analysis need to be explored as an interactive and undirected search for structures and trends. A major concern for doing this is overlooking a large information space which in many cases is conflicting with the information at hand and needs to be integrated from heterogeneous data sources. After applying analytical reasoning, such data sets can be either affirmed or discarded and eventually lead to better understanding of data.

The first process in Visual Analytic would be to transform the data in order to extract meaningful units of data for further processing using techniques like cleaning, normalization, grouping, or integration of heterogeneous data into a common schema. Then using the interaction with the decision maker, insightful information need to be revealed from the visualization. These findings can be reused to build a model for automatic analysis. After the successful creation of a model, the new analysts would have the ability to interact with the automatic methods by modifying parameters or selecting other types of analysis algorithms. Knowledge should be gained from visualization, automatic analysis, as well as the preceding interactions between visualization, models and decision maker or analysts.

Visual analytic is responsible to create a higher level view onto the data set, while maximizing the amount of details at the same time. The existing technologies to handle the above drawbacks have not been used widely primarily because of the users refusal to change their daily working routines. User acceptance is another challenge to be tackled since the advantages of developed tools need to be properly communicated to the audience of further users to overcome usage barriers and to tap the full potential of the visual analytic approach.

5.3 Seamless Data Integration

A lot of research was done regarding Data Integration and its techniques. Mainly it involves integrating the sources in an efficient way. One approach that was implemented [5] was an integration learner or as it labeled "Smart Copy and Paste" where the used system watches as the user copies data from applications (including the Web browser) then the system proposes an auto completion and the user provides feedback with each proposal. Another technique [14] introduced was "Data Curation" which is the act of discovering data sources that helps the local data sources based on attribute identification. In the same fashion, a paper talked about a web application "Mash up" [15] that integrates data from multiple web sources to provide a unique service that involves solving multiple problems, such as extracting data from multiple web sources, cleaning it, and combining it together. Existing work relies on a widget paradigm where users address problems during a Mashup building process by selecting, customizing, and connecting widgets together. In this way the user can create Mashups intuitively and the user won't need any programming background. However, none introduced the idea of suggesting prior action that may become useful for Integrating Data seamlessly. In our research, we are planning to suggest previous alteration that may result in improvements in results or at least shorten the amount of time taken in gathering information since it should minimally interrupt the user's analysis task. As a result, we are trying to implement a new improved data integration that helps the user that has no programming background and save resources such as time.

5.4 Common Formalism

There exist no such software to support the complete analysis cycle consisting of cleaning, integration and collaboration from GUI efficiently. A software called SAS was developed that integrated data cleaning, integration and GUI. It was considering number of benefits such as reuse work by others, eliminate overlapping redundant tools and systems, extract, cleanse, transform, conform, aggregate, load and manage data but none considered the approach we are trying to achieve which is taking advantage of one framework working concurrently instead of doing each part of the date enrichment integration, and cleaning sequentially.

6. STATUS REPORT

We have taken the approach of comparing the quantitative performance of all the techniques discussed above in each of the categories. For example, we analyze the efficiency and accuracy of similarity join technique by some existing algorithms and conditional functional dependency based data cleaning to compare the effect they might have on the overall system.

6.1 Data Cleaning

We discussed about similarity join in the previous section and now we will talk about the performance of certain existing algorithms which implement this join. So for the comparison a synthetic dataset was created by randomly drawing points from a uniform distribution of 4 million points and 1024 dimensions. The three algorithms that are discussed in various papers are LSS [9], EGO [16], [17] and GESS [18]. The execution times shown in the graphs [9] below include the time spend on index construction, I/O and join processing.

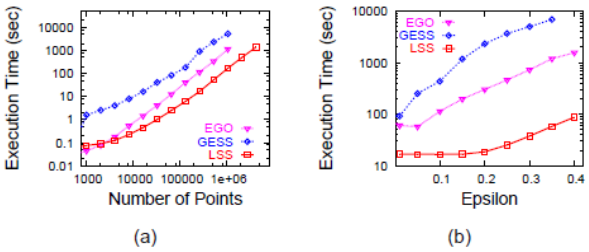


Figure 3. Execution times on uniformly-distributed data, for (a) varying n up to 4M points, ϵ=0.1, d=16, and (b) n=256k, varying ϵ, d=16

This figure is taken from [9] where they have proposed the LSS algorithm which improves on GESS and EGO. The figure clearly shows the LSS outperforms the other two for all values of n. Since these results are based on synthetic datasets the authors of [9] have taken a real word dataset of 60,000 photo images and extracted image features like Color Histogram, Color Moments, Layout Histogram and Cooc Texture from them. A similar plot has been shown in [9] which again proves that the LSS algorithm for similarity join is better than its competitors.

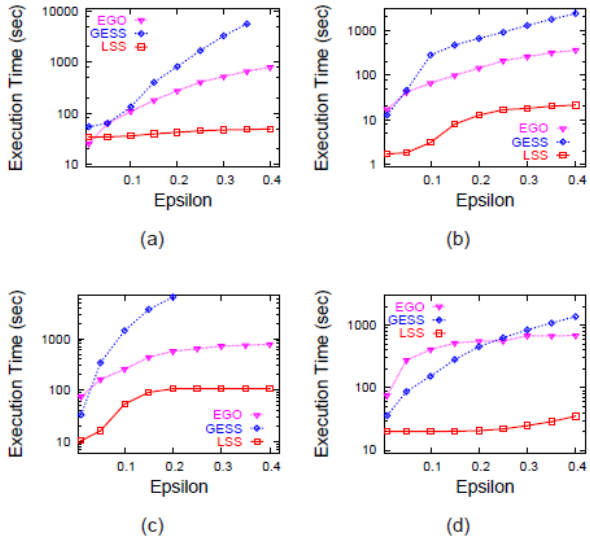
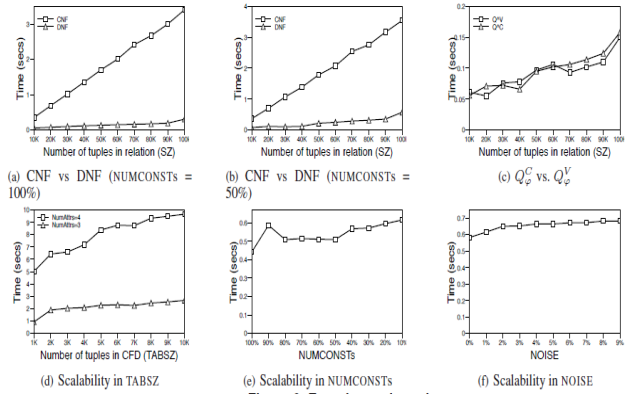


Figure 4. Execution times for varying ϵ on (a) ColorHistogram, (b) ColorMoments, (c) CoocTexture, and (d) LayoutHistogram

Now that we have a fast algorithm for similarity join we will have a look at the other data cleaning techniques and compare the execution times of the prevalent algorithms. Next we look at conditional functional dependency based data cleaning.

To test this method [11] describes a relation which model's tax records with 8 additional attributes like state of residency, salary, tax rate, marital status, existence of dependents and others related to tax exemptions. They populate this data from real life examples and also vary the parameters SZ (tuple number in the relation) and NOISE (percentage of dirty tuples). While generating the data they change the RHS of a CFD to an incorrect value with the probability NOISE. The CFDs include zipcode implies states, states and salaries determine tax bracket and so on. They vary the following parameters: NUMCFDs, number of CFDs, NUMATTRs, the maximum number of attributes in CFDs, TABSZ, the maximum number of tuples in CFDs and NUMCONSTs, the percentage of tuples with constants against ones with variables in each CFD. They have presented a



comparison between two strategies of query evaluation namely conjunctive normal form (CNF) and disjunctive normal form (DNF). The figures below are from [9] and they show the execution times for each of these startegies if some of the parameters is varied. As the graphs show for varying values of the number of tuples from 10,000 to 100,000 in 10,000 increments and constant values for other parameters DNF clearly outperforms CNF. Even when we change the percentage of constants in the CFD from 100% to 50% DNF is still better than CNF. The other figures show the scalability of the detection queries while a particular parameter is varied.

Thus we have gone through the algorithms to implement the two data cleaning techniques we discussed before. We have also pointed out which algorithms provide faster performance and in which factors. This will help us in choosing the best possible startegy for data cleaning in our application.

6.2 Data Visualization

To support interactive analysis, many of the databases are augmented with hierarchical structures that provide meaningful levels of abstraction that can be leveraged by both the computer and analyst. This hierarchical structure generates many challenges and opportunities in the design of systems for the query, analysis, and visualization of these databases. We present an interactive visual exploration tool that facilitates exploratory analysis of data warehouses with rich hierarchical structure, such as might be stored in data cubes. We base this tool on Polaris, a system for rapidly constructing table based graphical displays of multidimensional databases. Polaris builds visualizations using an algebraic formalism that is derived from the interface and interpreted as a set of queries to a database.

Relational Databases vs Data Cubes

Relational databases organize data into relations, where each row in a relation corresponds to a basic entity or fact and each column represents a property of that entity. The fields within a relation can be partitioned into two types: dimensions and measures. Dimensions and measures are similar to independent and dependent variables in traditional analysis.

In many data warehouses, these multidimensional databases are structured as n-dimensional data cubes. Each dimension in the data cube corresponds to one dimension in the relational schema. Each cell in the data cube contains all the measures in the relational schema corresponding to a unique combination of values for each dimension.

The dimensions within a data cube are often augmented with a hierarchical structure. This hierarchical structure may be derived from the semantic levels of detail within the dimension or generated from classification algorithms. Using these hierarchies, the analyst can explore and analyze the data cube at multiple meaningful levels of aggregation. Each cell in the data cube corresponds to the measures of the base fact table aggregated to the proper level of detail.

The aggregation levels are determined from the hierarchical dimension, which is structured as a tree with multiple levels. Each level corresponds to a different level of detail for that dimension. Within each level of the tree there are many nodes, with each node corresponding to a value within the domain of that level of detail of that dimension. These relationships are the basis for aggregation, drill down, and roll up operations within the dimension hierarchy. The following figure illustrates the dimension hierarchy for a Time dimension. Simple hierarchies, like the one shown in figure below, are commonly modeled using a star schema. The entire dimensional hierarchy is represented by a single dimension table (also stored as a relation) joined to the base fact table. In this type of hierarchy, there is only one path of aggregation. However, there are more complex dimension hierarchies where the aggregation path can branch.

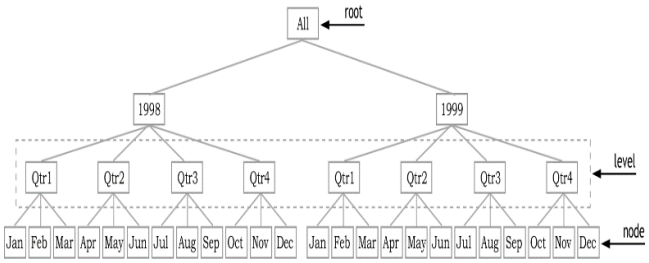


Figure 1: A hierarchical Time dimension with four levels: All, Year, Quarter, and Month.

Polaris Overview

The goal of Polaris was to provide an interface for rapidly and incrementally generating table-based. Users construct these table-based visualizations via a drag-and-drop interface, dragging field names from the Schema box to various blue shelves throughout the interface, as shown in the figure below.

Each visual specification is an expression of the Polaris formalism that can be interpreted to determine the exact analysis, query, and drawing operations to be performed by the system. The specification consists of two main portions. The first portion, built on top of an algebra, describes the structure of the table-based visualization.

While the first portion of the specification determines the "outer table layout," the within a pane and the computation of statistical properties remaining portion determines the layout within a pane, such as how the data within a pane is transformed for analysis and how it is encoded visually. Specifically, it describes:

1. The sorting and filtering of fields.
2. The mapping of data sources to layers.
3. The grouping of data, aggregates, and other derived fields.

4. The type of graphic displayed in each pane of the table. Each graphic consists of a set of marks (e.g., circles, bars, glyphs, etc.), with one mark per record in that pane.

5. The mapping of data fields to retinal properties of the marks in the graphics (for example, mapping Profit to the size of a mark or Quarter to the color).

Redefining the User Interface

Having redefined the formalism underlying the Polaris interface, we must now alter the interface to support hierarchically structured data. Five major changes need to be made:

1. Schema list must display dimension hierarchies and measures, not simply database fields;

2. Analyst must be able to distinguish between Month and Year .Month when including Month in a specification;

3. Analyst must be able to filter a dimension level using qualified values;

4. Analyst must be able to quickly drill down and roll up a dimension hierarchy using the interface;

5. Analyst needs to be able to change the number of marks within each pane to reflect different levels of detail.

1. The Schema

In the original interface, the analyst was presented with a list box containing the ordinal and quantitative fields in the database. The analysts included these fields in a specification simply by dragging and dropping the field's name onto the appropriate shelf in the interface. To support hierarchical data cubes, we have extended this list box to display the dimensions of the data cube with an ordered list of the dimension's levels beneath each dimension. The analysts can drag and drop any dimension level to the interface as they did with the ordinal fields of the database. The dimension's name, however, cannot be dragged to the interface; the analyst can only manipulate the individual levels within a dimension.

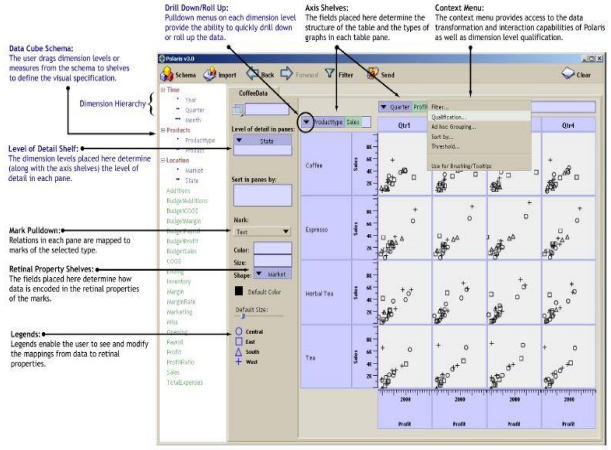


Figure 2: The Polaris user interface.

2. Qualifying Dimension Levels

When an analyst drops a dimension level, such as Month, on a shelf, there are several potential intentions. He may intend to include the operand Month in an expression, but he may also mean Year.Month or Year.Quarter .Month; the analyst needs to be able to specify the exact qualification desired. Our solution is the make full qualification (e.g., Year.Quarter.Month) the default. To generate a different qualification, the user can right-click the dimension level in the shelf and select the "Qualification. . . " menu item. He is then presented with a dialog box that allows him to explicitly specify which of the intermediate levels to include in the qualification of the operand, thus generating the applicable expression.

3. Qualifying Dimension Level Filters

When applying filters to a dimension level, an analyst may want to specify the filter using either qualified or unqualified values. We have extended the Polaris interface to allow both options. For ex- ample, if the user wishes to exclude 1998.Jan but not 1999.Jan, he can choose to filter using qualified values. Similarly, it is possible to specify a filtering using unqualified values: each qualified value that matches the unqualified value will be included in the filter. Currently, Polaris requires the filter be specified using either qualified or unqualified values, but not both. As a future extension, we intend to support heterogeneous filtering.

4. Drilling down and Rolling up

When analyzing and exploring large data cubes, a common operation is to drill down or roll up within a dimension hierarchy. There- fore, it is important to include a simple mechanism for performing these operations. One option is for the analyst to remove the cur- rent level from the appropriate shelf (by dragging it off the shelf) and then drag the new level to that same shelf. Although the desired effect is achieved, it is more complicated than we would like.

We provide an alternate mechanism for drilling down and rolling up a dimension. Within the box representing each dimension level on a shelf, there is a "∇" icon, as can be seen in Figure 2. When the user clicks on the "∇" icon, he is presented with a listing of all the levels of the dimension (including diverging levels in complex dimensional hierarchies). When a new level is selected, this is interpreted as a drill down (or roll up) operation along that dimension and the current level is automatically replaced with the selected level (with the same qualification). Thus, the user can rapidly move between different levels of detail along a dimension, refining the visualization as he navigates.

5. Grouping within Panes

In the original version of Polaris, the analyst specified the grouping of tuples within each pane by placing fields on the shelf titled "Group By". Each field in this shelf was included in the GROUP BY clause in the SQL query that aggregated the data in each pane into tuples to be mapped to marks.

The situation when visualizing data cubes is slightly different. The query for each pane does not produce a relational data set that is then grouped and aggregated. Instead, each pane corresponds to a projection of the data cube, with the projection determined by the dimension levels included in the table expressions. To produce additional marks within a pane, the analyst must specify additional dimensions to be included in these projections, done by including the desired dimension levels in the "Level of Detail" shelf (shown in Figure 2), the hierarchical analog of the "Group by" shelf.

As was the case with the original version of Polaris, this "Level of Detail" shelf gives the analyst the ability to rapidly drill down into their data without changing the table configuration. Changing the level of detail without changing the table configuration only changes the data density within each pane.

6.3 Seamless Data Integration

We approach each technique and analyze it from systems that uses that techniques. First, we analyze the "Smart Copy and Paste" technique. We take a look to a Prototype called "CopyCat" [5] that implements the "Smart Copy and Paste" (a screen shot showing the auto completion function figure 3). As you can see in the figure you have the option to accept the suggestion from the system or reject it. The system takes the feedback and learn if the suggestion is relevant or not. This can be very useful in helping both Regular users and expert users. In CopyCat, three kind of source learners are used:

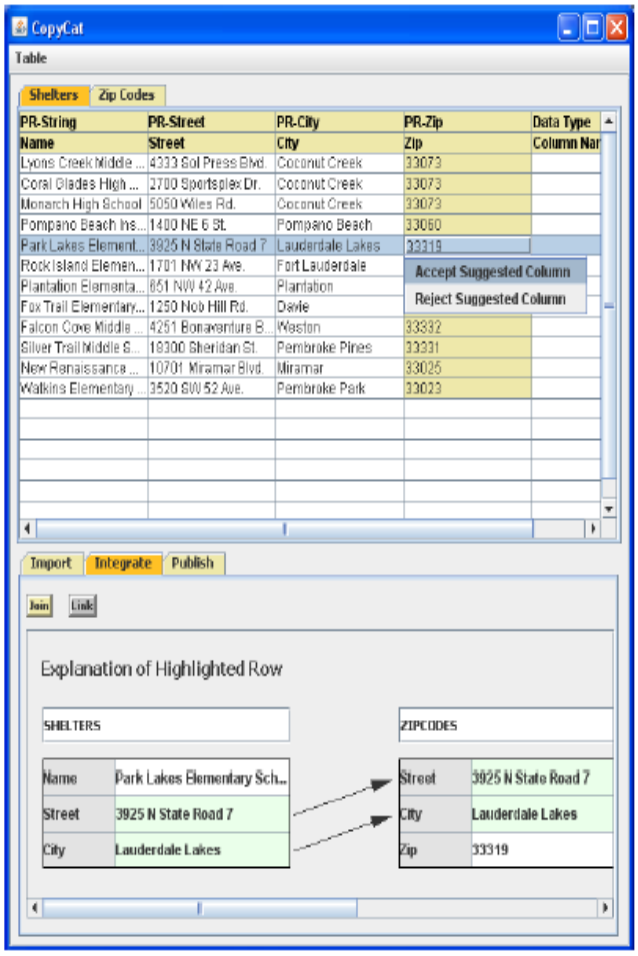


Figure 3: Screen shot showing The Auto completion function in Copy Cat [5].

Structural Learner which in short try to identify the root of copied data and try to have an idea or generalize the operation behind that copy and paste.

Model Learner which in short try to find similar relevant data manipulation to perform to find a useful data or data representation.

Integration Learner which in short learn from the feedback on how to integrate the queries and relevant data. Figure 3 shows the above Learners and the architecture of a prototype using the SCP technique (CopyCat).

Another Technique we mentioned was "Data Curation". As we mentioned before it is the act of discovering data sources that helps the local data sources based on attribute identification and eliminate the duplicates. We observe a system called "Data Tamer" [14] developed by M.I,T that uses Data Curation. A paper showed an interesting comparison [14] between Data Tamer and a web aggregator by using a set of 50 data Sources each one of them have 4000 records. Then it was duplicated then used the two systems with a corresponding statistics showing in Figure 5.

The total number of records is 146690. The Aggregator algorithm reported 7668 pairs. On the other hand, Data Tamer reported 180445 pairs as duplicated pairs. Have in mind that the number of true duplicates is 182453.

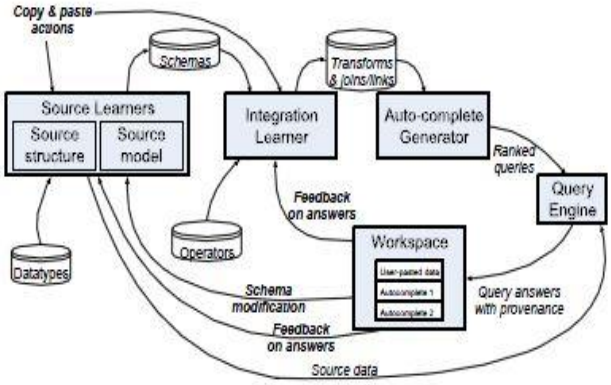


Figure 4. The architecture of a prototype of an SCP System

	Aggregator	Data Tamer
Total records	146690	
Pairs reported as duplicates	7668	180445
Common reported pairs	5437	
Total number of true duplicates (estimated)	182453	
Reported true duplicates (estimated)	7444	180445
Precision	97%	100%
Recall	4%	98.9%

Figure 5. The quality results of both Web Aggregator and Data Tamer [14]

	Data Retrieval	Source Modeling	Data Cleaning	Data Integration
Task1 K	3	7	6	0
Task1 DP	8	10	21	9
Task2 K	9	10	0	0
Task2 DP	18	30	0	28
Task3 K	5	10	4	5
Task3 DP	8	11	16	12

Table1. Evaluation of three tasks. K represent Karma while DP is a combined Dapper and Pipes system.

Another technique discussed was building Mash ups by example we did some research and we found a system called Karma [15].

Karma comparison was found which uses the technique we are interested (the idea of suggesting previous alteration to similar Data sets) with other mashups that does not have that feature: Dapper (Dapper.com), Yahoo Pipes (Pipes.Yahoo.com)

Table 1 shows the number of steps for each tasks, DP stands for Dapper and Pipe while K stands for Karma. As you can see in the table for all three tasks DP take more steps in completing the tasks.

Tasks description:

Task1 is pulling and cleaning the data from one source.

Task2 is extracting from 2 sources. Karma does the extraction in one table while DP does it separately.

Task3 is same as Task2 but we assume the data is already extracted.

7. REFERENCES

[1] Tableau Public. http://www.tableaupublic.com/, 2012.

[2] Kristi Morton, Magdalena Balazinska, Dan Grossman and Jock Mackinlay. 2014. Support the Data Enthusiast: Challenges for Next-Generation Data-Analysis System. *Proceedings of the VLDB Endowment, Vol. 7, No. 6*

[3] T. Dasu et al. 2003. Exploratory Data Mining and Data Cleaning. John Wiley & Sons. New York, NY.

[4] A. Halevy et al. 2012. Principles of Data Integration. Morgan Kaufmann

[5] Z. G. Ives et al. 2009. Interactive data integration through smart copy and paste. In *CIDR*, 2009.

[6] Hanrahan, P. 2012. Analytic database technologies for a new kind of user: the data enthusiast. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, New York, NY.

[7] E. Wu et al. 2013. Scorpion: Explaining Away Outliers in Aggregate Queries. In *VLDB*, 2013

[8] Jiannan Wang, Guoliang Li, Jianhua Feng. 2012. Can We Beat the Prefix Filtering? An Adaptive Framework for Similarity Join and Search. In *SIGMOD*, 2012

[9] Michael D. Lieberman, Jagan Sankaranarayanan, Hanan Samet. 2008. A Fast Similarity Join Algorithm Using Graphics Processing Units. In *Proceedings of the 24th IEEE International Conference on Data Engineering, pp. 1111–1120*, Cancun, Mexico, April 2008

[10] J. Chomicki and J. Marcinkowski. 2004. Minimal-Change Integrity Maintenance Using Tuple Deletions. In *Information and Computation, 197(1-2):90–121*, 2004.

[11] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, Anastasios Kementsietsidis. 2007. Conditional Functional Dependencies for Data Cleaning. In *ICDE*, 2007

[12] Daniel A. Keim, Florian Mansmann, Andreas Sto#el, Hartmut Ziegler. Visual Analytics. University of Konstanz, Germany

[13] Dr. Georges Grinstein, Dr. Bhavani Thuraisingham. Data Mining and Data Visualization. In *Second IEEE Workshop on Database Issues for Data Visualization*

[14] M. Stonebraker et al. 2103. Data curation at scale: The data tamer system. In *CIDR*, 2013.

[15] R. Tuchinda et al. 2008. Building mashups by example. In *IUI*, 2008

[16] C. Bohm, B. Braunmuller, F. Krebs, and H. P. Kriegel, "Epsilon grid order: an algorithm for the similarity join on massive high-dimensional data," in SIGMOD'01, Santa Barbara, CA, May 2001, pp. 379–390.

[17] D. V. Kalashnikov and S. Prabhakar, "Similarity join for low and high dimensional data," in *DASFAA '03*, Kyoto, Japan, Mar. 2003, pp. 7–16.

[18] J. P. Dittrich and B. Seeger, "GESS: a scalable similarity-join algorithm for mining large data sets in high dimensional spaces," in KDD'01, San Francisco, CA, Aug. 2001, pp. 47–56.

[19] Query, Analysis, and Visualization of Hierarchically Structured Data using Polaris Chris Stolte, Diane Tang, Pat Hanrahan Stanford University