Literature review Report
Lenses an “on-demand” approach to ETL

Purva Yogesh Lila
Masters in Computer Science
Illinois Institute Of Technology
Chicago, USA
plila1@hawk.iit.edu

Shubham Singh
Masters in Computer Science
Illinois Institute Of Technology
Chicago, USA
ssingh127@hawk.iit.edu

Suraj Nammi
Masters in Computer Science
Illinois Institute Of Technology
Chicago, USA
snammi@hawk.iit.edu

1. INTRODUCTION

We investigate the construction of a broad, extendable infrastructure for on-demand curation based on probabilistic query processing in this paper. Our experimental results demonstrate that On-Demand ETL is feasible and that our greedy ranking technique for curation jobs, known as CPI, is effective. We demonstrate how such infrastructure may be utilized to smoothly make current ETL work available on-demand.

Data curation usually requires a High quality data and ad-hoc approach by collecting data into data lake and querying with analysts. To make this approach more simple and feasible, On Demand ETL is considered for curation tasks. Systems like Paygo and Hlog are following incremental curation of data which helps the analysts to work more efficiently. These approaches emerged and made on-demand ETL to be more feasible in curating the data for analysts.Effective analytics depends on Accuracy,reliability and High-quality data. The data should be thoroughly cleaned up-front before the ETL process to manifest data quality.This cleansed data is represented in tables in a data warehouse and these tables represent in the form of Star Schemas. Only after loading the parsing the transformed data into the data warehouse analysts can analyze the data. The whole process is summarized into the challenges of composing specialized on-demand curation techniques into a general-purpose workflow, resulting in a unified model for curation called On-Demand ETL.

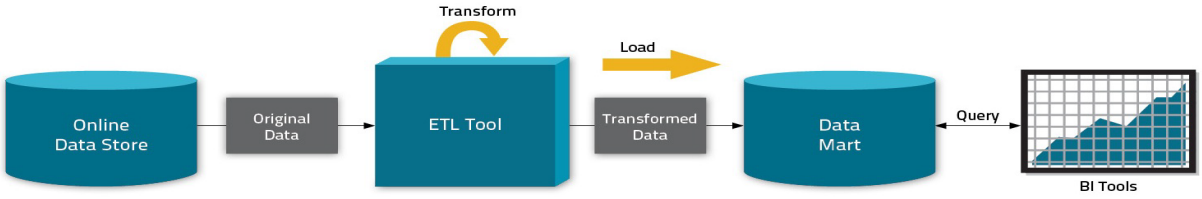
A typical ETL pipeline often involves many distinct curation tasks, requiring that multiple on-demand data curation systems be used in tandem. However, the data representations and quality metrics used by these systems are optimized for very specific use-cases, making composition difficult. This on demand ETL model builds around ordinary SQL, retaining compatibility with existing standards for ETL design, data analysis, and database management. We develop a practical implementation of PC-Tables [23] called Virtual C-Tables that can be safely embedded into a classical, deterministic database system or ETL workflow. The cost of perfect information is a concept that relates the value of a curation task that improves a result's quality, to the cost of performing the task. These links allow for lens-defined curation tasks to be ranked according to their net value to the analyst.

2. BACKGROUND

Traditional ETL Systems:

In the world of data warehousing, there's a need to bring data from multiple different data sources into one, centralized database. For performing such operations, there are some steps that need to be performed.

i) EXTRACT data from its original source
ii) TRANSFORM data by deduplicating it, combining it, and ensuring quality, to then
iii) LOAD data into the target database



On Demand ETL:

Traditional ETL systems require data to be clean in order to process properly. The upfront costs of data cleaning and curation have led many to instead inline curation tasks into the analytical process, so only immediately relevant curation tasks are performed. This deferred approach is more lightweight, but encourages analysts to develop brittle one-off data cleansing solutions, incurring significant duplication of effort or organizational overheads. This form of on-demand curation results in a sanitized data set that is based on a principled trade-off between the quality desired from the data set and the human effort invested in curating it. The result is a unified model for on-demand curation called On-Demand ETL that bridges the gap between these systems and allows them to be gracefully incorporated into existing ETL and analytics workflows.

On Demand ETL Properties:

i) Representing Incomplete Data - Existing on-demand curation systems use specialized, task-specific representations.
ii) Expressing Composition - If the output of a curation technique is non-deterministic, then it must accept non-deterministic input as well. The paper introduces a model for non-deterministic operators called lenses that capture the semantics of on-demand data curation processes.
iii) Backwards Compatibility - For On-Demand ETL to be compatible with traditional data management systems and ETL pipelines, implementation of PC-Tables called Virtual C-Tables can be embedded into a classical, deterministic database system or ETL workflow.
iv) Presenting Data Quality - There can be major quality loss incurred by incomplete curation to end-users. On-Demand ETL computes a variety of quality measures for query results.

BACKGROUND - Virtual C tables, Normal form VG- RA

A deterministic database is a finite collection of relation instances {R1, . . . , Rk} over a schema S = {S1, . . . , Sk}. According to the "possible worlds" semantics a probabilistic database D consists of a pair (W, P), where W is a large collection of deterministic databases, the so called possible worlds, all sharing the same schema S, and P is a probability measure over W.

P(t ∈ D) = Summation Over (P(Wi)) for (Wi∈W|t∈Wi)

If W is representable by a C-Table and Q is a positive query then W0 = {Q(Wi) | Wi ∈ W} is representable by another C-Table (Cured-Table).
An approach VG-RA (variable-generating relational algebra), a generalization of positive bag-relation algebra with extended projection, that uses a simplified form of VG-functions can be adopted. In VG-RA, VG-functions (i) dynamically introduce new Skolem symbols in Σ, that are guaranteed to be unique and deterministically derived by the function's parameters, and (ii) associate the new symbols with probability distributions. Hence, VG-RA can be used to define a new C-Table.

These semantic rules used in C-Tables make extensive use of the lazy evaluation operator [[·]]lazy, which uses a partial binding of Column or Var(. . .) atoms to corresponding expressions. Lazy evaluation applies the partial binding and then reduces every sub-tree in the expression that can be deterministically evaluated.

3. LENSES

It is a data processing component that is used to assess ETL pipelining. Lenses generate a PC-Table (incomplete information tables) that specifies a set of potential outputs and probability measures to accurate output in the actual world. Lenses organize uncertainty in the ETL process and interpret input data. Lenses enable the ETL process to run swiftly and clearly.

LENS Framework

A lens instance is defined over a query Q(D), and is in turn responsible for constructing a PC-Table (W; P). A lens defines W as a C-Table through a VG-RA expression.A lens defines W as a C-Table through a VG-RA expression Flens(Q(D)). Independently, the lens constructs P as a joint probability distribution over every variable introduced by Flens, by defining a sampling process in the style of classical VG-functions, or supplementing it with additional metadata to create a PIP-style grey-box.

So, There are three LENSES which are defined over the PC-Tables for solving the uncertainties in data.

1. **Domain Constraint Repair** (Replace Null values with the approximated values)
2. **Schema Matching**
3. **Archival**

Query to create a LENS

CREATE LENS <lens_name> AS Q
USING <lens_type>(<lens_arguments>);

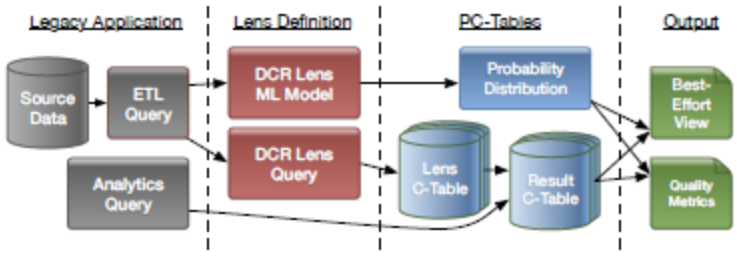
Where "lens_type" can be : DOMAIN REPAIR/SCHEMA_MATCHING/ARCHIVAL

Lens Examples

LENSES functioning example: Lenses as Domain Constraint Repair, There might be some attribute which can't be not null attributes but due to data entry errors or missing information the value of attribute can still be a null value. In this scenario lenses are used to replace reliable values with the null values and help ETL designers to process more efficiently. The replacements are reliable as the values chosen by lens are based on educated guesses of the data domain and approximations.

Lens for Domain Constraint Repair

- Domain constraint repair lens is defined to enforce attribute-level constraints such as NOT NULL.
- Under the assumption that constraint violations are a consequence of data-entry errors or missing values.
- Domain constraint violations can be repaired by finding a legitimate replacement for each invalid value by obtaining reliable replacement values.
- Reliable Replacements require detailed domain Knowledge.
- However Domain Constraints Repair Lens uses Approximations based on data domain and machine learning models.



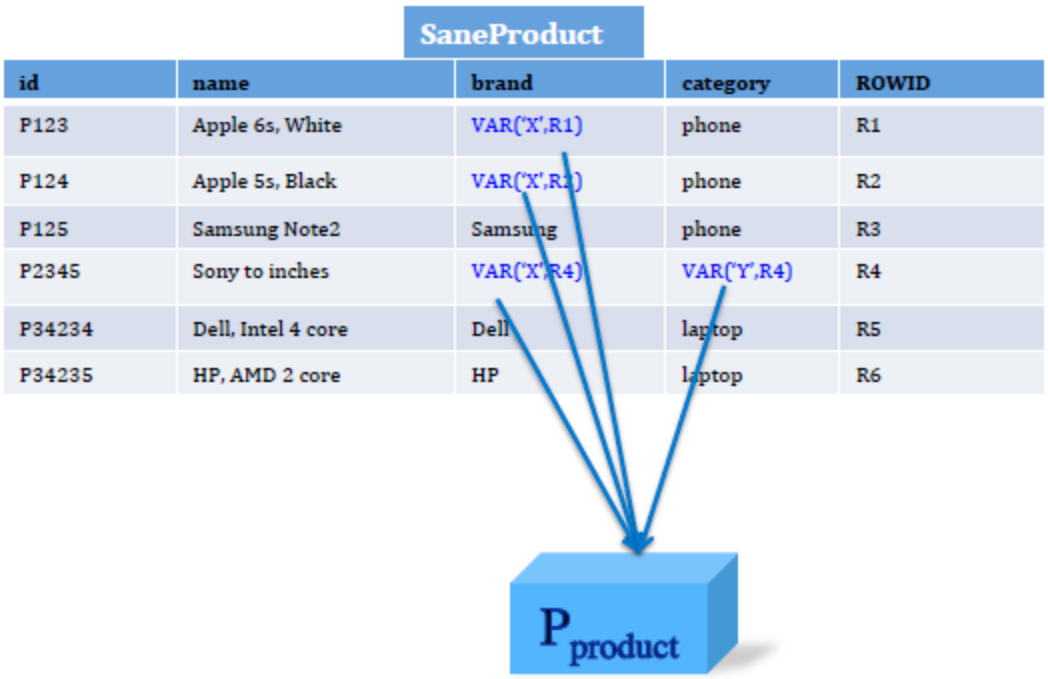
Example of the domain constraint repair lens applied in a legacy application.

Sample Query to create domain constraint repair LENS.

CREATE LENS SaneProduct AS
SELECT * FROM Product USING DOMAIN_REPAIR(category string NOT NULL, brand string NOT NULL);

Lens produces a PC Table, which defines the set of possible outputs, and a probability measure that approximates the likelihood that any given possible output accurately models the real world.

id	name	brand	category	ROWID
P123	Apple 6s, White	NULL	phone	R1
P124	Apple 5s, Black	NULL	phone	R2
P125	Samsung Note2	Samsung	phone	R3
P2345	Sony to inches	NULL	NULL	R4
P34234	Dell, Intel 4 core	Dell	laptop	R5
P34235	HP, AMD 2 core	HP	laptop	R6



CREATE LENS SaneProduct AS SELECT * FROM Product USING DOMAIN_REPAIR(category string NOT NULL, brand string NOT NULL);

Limitations:

Replacements based on the machine learning models need to be trained on a larger number of datasets. Some times approximations cannot be authentic.(As it depends on the approximated results and there is a possibility of wrong predictions)

Lens for Schema Matching

- The schema matching Lens is defined to map the source data schema with the target data schema.
- Schema Matching Lens is more useful in mapping non-relational data like web data,JSON.
- The schema matching lenses define a boolean value for every pair of target schema.
- The Lens is defined to give the probable number propabilities of this boolean value to match the target schema.

Sample Query to create Schema matching LENS

CREATE LENS MatchedRatings2 AS SELECT * FROM Ratings2
USING SCHEMA_MATCHING(pid string, ..., rating float,
review_ct float, NO LIMIT);

CREATE VIEW AllRatings AS SELECT * FROM MatchedRatings2
UNION SELECT * FROM Ratings1;

MatchedRatings2

pid	...	rating	Review_ct
P125	...	If Var('rat=eval') then 3 else If Var('rat=num_rating') then 121 else NULL	If Var('review_ct=eval') then 3 else If Var('review_ct=num_rating') then 121 else NULL
P34234	...	If Var('rat=eval') then 5 else If Var('rat=num_rating') then 5 else NULL	If Var('review_ct=eval') then 5 else If Var('review_ct=num_rating') then 5 else NULL
P34235	...	If Var('rat=eval') then 4.5 else If Var('rat=num_rating') then 4 else NULL	If Var('review_ct=eval') then 4.5 else If Var('review_ct=num_rating') then 4 else NULL

id	Category	rating	Review_ct	Φ(condition)
P123	phone	4.5	50	T
P124	phone	4	100	T
P125	phone	If Var('rat=eval') then ... ...else Var('Z',R10)	If Var('rat=eval') then ... ...else NULL	T
P2345	Var('Y',R4)	Var('Z',R8)	245	(Var('Y',R4) = 'phone') (Var('Y',R4) = 'TV') Var('Z',R8) > 4
P34234	laptop	If Var('rat=eval') then ... ...else Var('Z',R11)	If Var('rat=eval') then ... ...else NULL	Var('rat=eval') Var('rat=num_rating'))= Var('Z',R11) > 4
P34235	laptop	If Var('rat=eval') then ... ...else Var('Z',R12)	If Var('rat=eval') then ... ...else NULL	Var('rat=eval') Or (not Var('rat=num_rating') (and Var('Z',R11) > 4))

CREATE LENS MatchedRatings2 AS SELECT * FROM Ratings2 USING SCHEMA_MATCHING(pid string, ..., rating float, review_ct float, NO LIMIT);

CREATE VIEW AllRatings AS SELECT * FROM MatchedRatings2 UNION SELECT * FROM Ratings1;

The schema matching lens defines a fresh boolean variable V ar($Name_{ij}$) for every pair a_i,b_j , where $<a_i,t_i>$ sch(Q). The probability of Var($Name_{ij}$) corresponds to the probability of a match between a_i and b_j . Flens takes the form:

π{.....$b_j \leftarrow$ Vj….}, where V_j enumerates possible matches for b_j :

if V ar(Name1;j) then a1 else

⋮

if V ar($Name_{ij}$) then a_1 else NULL

Limitations

The incompatible pairs are always ignored. Which can also have a probable match of the data.

LENS for Archival

- An archival lens captures the potential for errors arising from OLAP queries being posed over stale data like queries run in between periodic OLTP to OLAP bulk data copies.
- The lens takes a list of pairs where (T,R) Where R is a reference to a relation in an OLTP database, and T is the period with which R is locally archived.

Sample Query to create Archival LENS

CREATE LENS MatchedRatings2 AS SELECT * FROM Ratings2
USING USING ARCHIVAL((T1,R1), …(Tm;Rm))

- This lens probabilistically discards rows from its output that are no-longer valid according to the lens query $F_{lens} = \sigma_{Var}$(Name,ROWID), where Name is a freshly allocated identifier. In the background, the lens periodically polls for samples drawn from each R_j to estimate the volatility of each relation referenced by Q. Denote by v_j the probability of a tuple in R_j being invalidated at some point during the period T_j . P is dened independently for each row as a binomial distribution with probability $\Pi_{\{j|\, Rj \in Q\}}^{Vj}$.

Limitations

The incompatible pairs are always ignored. Which can also have a probable match of the data.

4. VIRTUAL C-TABLES, NORMAL FORM VG-RA & VIRTUAL VIEWS

VIRTUAL C-TABLES

Virtual CTables or VC-Tables, works by decomposing VG-RA queries into deterministic and non-deterministic components. A small On-Demand ETL shim layer wraps around the database, and provides a minimally-invasive interface for uncertainty-aware users and applications. This shim layer is also responsible for managing several views that provide backwards compatibility for legacy applications. Virtual C-Tables decouple the deterministic components of a query from the non-deterministic components that define a PC-Table.

The data in a table containing NULL entries or incorrect values is corrected as →
if Var(rat = eval) then 3 else if Var(rat = num_r) then 121 else NULL

NORMAL FORM VG-RA

Non-determinism arises in VG-RA queries through expressions containing variable terms — that is, only through projection and selection.

VIRTUAL VIEWS

Normalization allows lenses to be incorporated into existing ETL pipelines. A lens constructs a probability measure P out of its input Q(D), and a C-Table using $F_{lens}(Q(D))$. The lens query and any subsequent queries over it are normalized into a normal form query F'(Q'(D)), and the view Q'(D) is constructed and materialized by the traditional database. F' is stored alongside Q' and defines a virtual view for F'(Q'(D)). The shim interface transparently normalizes queries over virtual views q(F'(Q'(D))) to F''(q'(Q'(D))), allowing q'(Q'(D)) to be evaluated by the traditional database. View definitions and SELECT INTO queries are similarly rewritten, dening new virtual views instead of their normal behavior.

5. ANALYSIS

Using virtual views, queries over lens outputs are rewritten into the normal form F(Q(D)), and Q(D) is evaluated by the database.

The C-Table construction query F is of minimal use in its raw form. We next turn to the construction of user-consumable summaries of F. Using virtual views, queries over lens outputs are rewritten into the normal form F(Q(D), and Q(D) is evaluated by the database.

Summarizing the Result Relation:

A deterministic relation R_{det}, and a best-guess relation R_{guess}. The deterministic relation R_{det} represents the certain answers of the virtual C-Table, and is constructed by replacing every variable reference in each e_i and ϕ with NULL, and dropping rows where $\phi \neq T$.

$$\text{SELECT } e_2(* \rightarrow \text{NULL}) \text{ AS } a_2 \text{ FROM } Q(D) \text{ WHERE } \phi (* \rightarrow \text{NULL})$$

The resulting relation contains all the rows deterministically present in F(Q(D)), with NULL taking the place of any non-deterministic values. R_{det} can be computed entirely within a classical deterministic database, making it backwards compatible with legacy ETL components. In uncertainty aware applications, this annotation is used to indicate which parts of the result are uncertain to the end-user. The shim layer evaluates each e_i and based on the valuation given by $\text{argmax}_v(P(v)$, the most likely possible world. Note that this best-guess estimate is not entirely accurate.

Summarizing Result Quality

Once the user is made aware that parts of a query result may be uncertain, two questions likely to be asked are How bad? and Why?. Answering the latter question is trivial:. F contains a reference to all of the variables that introduce uncertainty, each of which is explicitly linked to the lens that constructed it. In other words, F serves as a form of provenance that can be used to explain sources of uncertainty to the end-user.

The appearance of a tuple t in a query result is determined by the ground truth of its local condition $t.\Phi$. From the PC-Table's probability measure P(v), we get the binomial distribution $P(t.\Phi\ [v])$, often called the confidence of t. It is natural to use Shannon entropy as a metric to quantify the quality of the query result.

We define the entropy of a tuple in terms of its confidence $p_t = P(t.\Phi\ [v])$ as:

$$entropy(t) = -(p_t.\log_2(p_t) + (1 - p_t).\log_2(1 - p_t))$$

6. FEEDBACK

When an analyst is given a query result R that does not meet her quality expectations, they can allocate additional resources for gathering more evidence. Since N(R) depends on the entropy generated by the tuples in R, in expectation, each curation task will reduce the noise seen in the final result.

a) Prioritizing Curation Tasks

Prioritizing curation tasks is a dynamic decision process, where the outcome of one curation task affects the choice of the next one to be performed. In its general form, the problem can be thought of as a Markov Decision Process, having one state for each partial assignment to the variables in Φ and one action for each variable. The application of a policy is an interactive process: the system instructs the analyst to address a particular curation task and the analyst provides the required ground truth, and asks the system for the next move. This feedback loop continues until the deterministic value of phi is obtained.

b) Balancing Result Quality and Cost

The plan here is to minimize the hidden value function V(.) which depends on c(.) and N(.) and is unknown to the system. Clearly, we assume V(.) decreases monotonically as the cumulative cost increases, and increases monotonically as the noise decreases. In simple words, V determines how much the analyst is willing to pay for an improvement in the estimation of the value of , on a case-by-case basis.

We call this trade-off the ***cost of perfect information (CPI).***

Since the details of V(.) are unknown, the goal of the system is to propose several candidate policies.

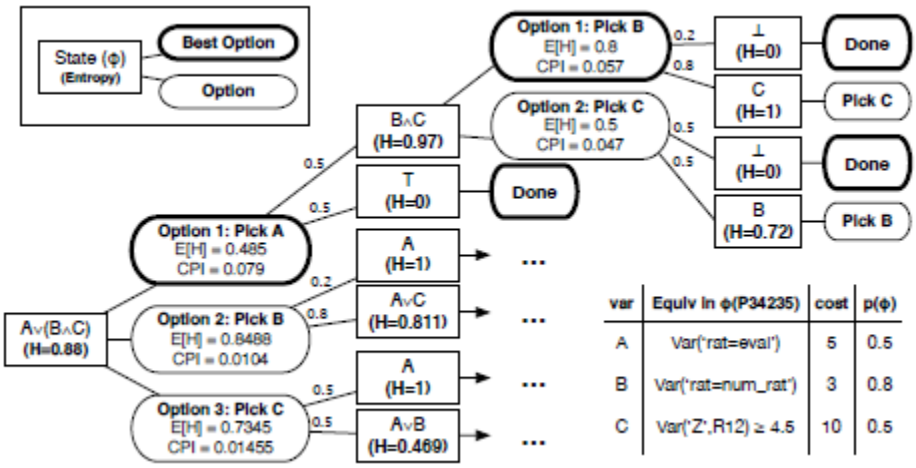
Each policy should guarantee a certain expected entropy at the price of a certain expected cumulative cost. The user is then able to choose the candidate policy that best matches her hidden value function. Since the analyst may be subject to budget constraints hidden to the system, the candidate

list includes greedy versions of the policies, computed progressively over limited planning horizons. Inspired by the algorithms EG2, CS ID3] and CS C4.5, On- Demand ETL supports the following four greedy strategies:

Algorithm	$CPI(v_i)$
EG2	$(2^{IG[\mathcal{R}(v_i)]} - 1)/c_i$
CS_ID3	$(IG[\mathcal{R}(v_i)]^2)/c_i$
CS_IDX	$IG[\mathcal{R}(v_i)]/c_i$
CS_C4.5	$IG[\mathcal{R}(v_i)]/\sqrt{c_i}$

Here, IG denotes the *information gain*, or the reduction in noise produced by the curation task on variable v_i.

Example: Consider the condition for P34235 in, which has the form A∨(B∧C). Below figure illustrates the decision tree that ranks curation tasks (the three variables), given lens-defined ground-truth costs and margina probabilities. The expected entropy after performing the curation task for v_i, denoted by $E[H(v_i)]$, is computed as a weighted average over all possible outcomes of the task. $CPI(v_i)$ is computed according to the CS_IDX formula given above, with $IG[R(v_i)] = H - E[H(v_i)]$.



An example of the CS IDX algorithm optimizing CPI.

Distances Metric	Correct Matches by Index								
	1	2	3	4	5	6	...	9	n/a
Levenstein	24	2	2	0	0	1	0	1	0
JaroWinkler	20	4	2	1	1	0	0	1	1
NGram	25	0	3	1	0	0	0	0	1

Distance evaluated by the index of the correct match in the ranked list of matches output by the algorithm, or n/a if the correct match was discarded.

7. EXPERIMENTS

Experiment, Lens Configuration, CPI, Composition and On-demand ETL.
Problems
Solutions - Author's thoughts and experiments
Effect - Impact of Solutions (What could be better and what not)

Experiment using an Lenses On - Demand:

Using a Domain constraint repair Lenses on a huge dataset which contains a lot of null values in data fields like pincodes ,area code for covid-tracing.

The lenses implemented here can help the dataset and the process of data cleaning much simpler by providing the most accurate approximations using the guess of data domain using machine learning models to analyze the possible attribute to approximate a possible output for the null value.This allows in the process of data cleaning in order to replace the null value with most closest approximations.complete data set values are classified in five categories: active, bayes, stochastic gradient descent, ensemble and tree. And calculate the most probable value of the missing data using tuple values of each attribute is trained using these classifiers.The most probable value for the missing tuple is calculated from results of the classifiers.

In paper examples of Credit data and Real Estate data the statistics are obtained using these classifiers where the author illustrates the data to show the most probable value used to replace using Domain Constraint Repair lens.

Using a Schema matching lens mainly matches the schema based on data type and data range.It eliminates candidates? matches and apply the distance metrics like Jaro-Wrinkler, Ngram, levenstein to and sort the results to get the best match based on similar string. Best results from these three metrics are taken into average and make the probable match.

The drawbacks using Lenses: Domain Constraint repair:

The lenses can't be used simply in the process of ETL as the replacements are approximations and guesses based on data domain. The possible replacements might also go wrong and also lead to more abnormalities in the data which makes the process more inefficient and also makes the data dirty.

Machine learning models in lenses need more time and resources that need to be educated to the lenses. It might also not be accurate as there might be two possible solutions given for the same data attribute.

Compositions and On-Demand ETL:

The On-demand ETL is very effective and also helps the analysts in data curation of large data and improve the quality of noisy data or dirty data with a limited budget. The process involves performing curation tasks iteratively while still achieving the right scores. The On-Demands ETLs play a key and unique role in data cleaning and data curation process which helps the analysts to work more efficiently and produce a perfect analysis for business improvement. Hence On-demand ETL simplifies the process of curation by enabling composing process operators like Lenses which generalizes the specific task and work on it iteratively and provide fully cleaned data which can be queried using standard SQL.

8. CONCLUSION

On-Demand ETL, which generalizes task-specific on-demand curation systems like Paygo, is introduced. On-Demand ETL permits the use of Lenses, which are composable non-deterministic data processing operators that create the illusion of completely cleansed relational data that can be searched using standard SQL. Lenses encode output using PC-Tables and may be deployed in traditional, deterministic database setups utilizing Virtual C-Tables. On-Demand ETL enables best-effort approximations about the contents of a PC-Table, evaluation of quality metrics throughout a PC-Table, and the CPI heuristic family for prioritizing curation activities. We've shown the viability and importance of On-Demand ETL, as well as the efficacy of CPI-based heuristics.

9. REFERENCES

01. Ying Yang, Niccolo Meneghetti, Ronny Fehling, Lenses: An On-Demand Approach to ETL.
02. https://papers.nips.cc/paper/2011/hash/303ed4c69846ab36c2904d3ba8573050-Abstract.html
03. https://www.javatpoint.com/machine-learning-naive-bayes-classifier
04. https://machinelearningmastery.com/classification-as-conditional-probability-and-the-naive-bayes-algorithm/
05. https://www.cuelogic.com/blog/the-levenshtein-algorithm
06. http://www.gabormelli.com/RKB/Jaro-Winkler_Distance_Measure