An Iterative, Self-Assessing Entity Resolution System: First Steps toward a Data Washing Machine

John R.Talburt[1], Awaad K. Al Sarkhi[2]
College of Science, Technology, Engineering, and Mathematics, The University of Arkansas at Little Rock
Little Rock, USA

Leon Claassens[4]
PiLog Group
Centurion,
South Africa

Daniel Pullen[3]
Noetic Partners,
New York, NY

Richard Wang[5]
Sloan Management School, MIT
Boston, MA

***Abstract*—Data curation is the process of acquiring multiple sources of data, assessing and improving data quality, standardizing, and integrating the data into a usable information product, and eventually disposing of the data. The research describes the building of a proof-of-concept for an unsupervised data curation process addressing a basic form of data cleansing in the form of identifying redundant records through entity resolution and spelling corrections. The novelty of the approach is to use ER as the first step using an unsupervised blocking and stop word scheme based on token frequency. A scoring matrix is used for linking unstandardized references, and an unsupervised process for evaluating linking results based on cluster entropy. The ER process is iterative, and in each iteration, the match threshold is increased. The prototype was tested on 18 fully-annotated test samples of primarily synthetic person data varied in two different ways, good data quality versus poor data quality, and a single record layout versus two different record layouts. In samples with good data quality and using both single and mixed layouts, the final clusters had an average F-measure of 0.91, precision of 0.96, and recall of 0.87 outcomes comparable to results from a supervised ER process. In samples with poor data quality whether mixed or single layout, the average F-measure was 0.78, precision 0.74, and recall 0.83 showing that data quality assessment and improvement is still a critical component of successful data curation. The results demonstrate the feasibility of building an unsupervised ER engine to support data integration for good quality references while avoiding the time and effort to standardize reference sources to a common layout, design, and test matching rules, design blocking keys, or test blocking alignment. Also, the paper proposes how unsupervised data quality improvement processes could also be incorporated into the design allowing the model to address an even broader range of data curation applications.**

Keywords*—Unsupervised entity resolution; data curation; frequency blocking; entropy regulated; data washing machine

I. INTRODUCTION

As organizations ingest and process larger amounts of data, the time and effort it takes to prepare and integrate data into useful products are also increasing, and many researchers are working to alleviate this bottleneck using several different approaches [1], [2], [3]. The root cause of the time delay is human supervision of the curation steps including data quality analysis, data cleansing and standardization, entity resolution (ER), and data integration [4]. The goal of ER is to link two references if, and only if, the references are equivalent [5], [6]. The problem is only exacerbated by Big Data [7], [8]. Because of the time delay between receiving data and its availability for use, data analysts often face the choice of waiting for the preparation to be complete, or to by-pass the curation process and engage in their attempts at data preparation which may or may not follow the best practices.

Many organizations are beginning to recognize this time and effort gap between data ingestion and final information product, and are moving to remedy this situation by increasing the level of automation in data curation processes [9]. These organizations along with software vendors and university researchers are trying to understand how to apply the same AI and ML techniques used for the data analytics at the end to the automation to the preceding data preparation processes [10], [11]. While many of these employ AI and ML [12], [3], [13], they still largely rely on some level of standardization in the source data. The ultimate goal is to develop systems for unsupervised data curation (UDC) which are metadata agnostic and can directly ingest and process raw data. The objective of UDC is to develop methods and techniques to process data at scale and successfully produce information products without manual intervention. Key components of the data curation process and prime targets for automation are the largely manual processes of data quality analysis, building transformation for data cleansing and standardization, and developing and testing rules for entity resolution and data integration (fusion).

UDC has been likened to a "data washing machine" [14]. When using a household washing machine for laundry, the user first loads the dirty laundry, and detergent then selects the cycles. The washing machine automatically executes the cycles, and in the end, produces clean laundry. Similarly, the user of the data washing machine loads dirty data with appropriate reference data, then selects the data cycles (control parameters). The data washing machine then executes the cycles to produce clean data (an information product) appropriate for use in a particular application.

The focus of this research is to describe a proof-of-concept (POC) prototype to serve as both a starting point and a foundation upon which a more complete UDC can be built [15]. The primary goal is to develop unsupervised methods and techniques for both data cleaning and data integration (ER) capable of operating at scale. The current code for the POC described in this paper can be found at https://bitbucket.org/Awaad_Al_Sarkhi/dwm-datawashingmachine/src/master/

II. A Proof-of-Concept (POC) for Unsupervised Data Curation (UDC)

The purpose of the POC is to demonstrate the feasibility of cleaning and integrating entity references in an automated fashion for certain types of data and certain phases of the curation process. The primary use case addressed by the POC is "multiple sources of the same information" as described in [16] as one of ten root causes of data quality problems. The novelty of the POC is it attempts to perform unsupervised entity resolution (ER) first rather than data cleaning, the opposite of most supervised processes. The objective of the POC is to minimize human intervention to analyze and transform the data and still obtain usable results as measured by the accuracy of clustering, i.e. a working data washing machine for data deduplication.

The POC for the data washing machine was written in Python and Java and uses frequency-based blocking, a multi-token scoring matrix as its ER matching process, and entropy-based quality evaluation of clustering [17] , [18] , [19], [20] The assumptions of the POC are

- The input to the process is a text file in a comma-separated values (CSV) format.
- Each text line is a reference to the same type of entity such as person entities (patients, customers, students), business entities, or materials (product listings, machine parts).
- The references are not assumed to be standardized with a uniform metadata tagging. No metadata is used in the POC process. Any metadata in the form of a header record is discarded.
- The first string value in each text line is a unique reference identifier.

To facilitate experimentation with various unsupervised techniques, the POC was developed as a series of sub-processes or phases to facilitate experimentation. Currently, phases have been implemented, and the fourth phase for token correction is under development. The organization of this paper is as follows:

- Phase I: Punctuation removal, upper casing, and tokenization.
- Phase II: Global standardization (replacement) of non-numeric tokens at the file level.
- Phase III: Removal of stop words, blocking, and clustering of equivalent references (entity resolution).

A. Phase I - Tokenization

The first Phase reads each reference as a line of text and performs a series of operations. The first is to separate the reference identifier, convert all letters to uppercase, and replace the field delimiters (typically a comma) with a blank character. Next, all non-word characters (\W) are replaced. For experimentation, two methods of replacement for non-word characters were tried. In the first method called "Compress," the non-word characters are replaced by a null character. For example, if a field has the value "123-456", then after replacing the hyphen character with a null character is becomes the single string "123456". In the second method called "Splitter," each non-word character is replaced by a blank character. The same example "123-456" becomes two strings (tokens), "123" and "345".

The motivation for the Compress method was to transform characteristic values with punctuation such as telephone numbers and dates into a single string. Interestingly, for the data used for the initial validation of the POC, the Splitter method generally gave better results than the Compress method.

In addition to non-word character replacement, upper casing, and tokenization, the first Phase also has an option to de-duplicate tokens. If the duplicate token option is employed, any duplicates of tokens within the same reference are removed, otherwise, duplicates are left in the reference. In the end, the cleaned tokens from each reference are reassembled into a blank delimited string and written to the tokenized reference file.

B. Phase II – Global Token Replacement

Phase II attempts an unsupervised correction of misspelled tokens based on the token frequency and string similarity. The replacement uses the assumption, if a high frequency, the non-numeric token is very similar to a low-frequency, non-numeric token, the low-frequency token is likely to be a misspelling of the high-frequency token and can be replaced by the high-frequency token. The validity of this assumption is dependent upon several factors. These include, what is a high frequency, what is a low frequency, and what is very similar.

The process is controlled by four parameters:

- MinFreqStdToken – The minimum frequency of a token that can be used to replace another token, i.e. can function as a "standard" token.
- MinLenStdToken – The minimum string length of a standard token.
- MaxFreqErrToken – The maximum frequency of a token that can be replaced by a standard token, i.e. can be treated as an "error" token.
- MaxStringDist – The maximum string (character) distance between a standard token and an error token before the error token can be replaced (usually 1 as measured by Levenshtein edit distance).

The replacement table has a one-to-many relationship between standard tokens and error tokens. One standard token could replace many different error tokens, but an error token

can only be replaced by one standard token. Some actual examples of rows from the Replacement Table for Sample S8 are shown in Table I where

- MinFreqStdToken =10
- MaxFreqErrToken = 3
- MinLenStdToken = 4
- MaxStringDiff = 1

Table I shows some examples of token replacements generated in Phase II. It is important to note the token changes made in Phase II are not permanent changes to the source data. The token changes in Phase II are intended to improve the cluster (ER) results in Phase III.

TABLE I. EXAMPLE ROWS FROM REPLACEMENT

	Std Token	Freq	Error Token	Freq
1	APT	82	APTZ	1
2	APT	82	APLT	1
3	APT	82	APTR	1
4	CALIFORNIA	58	CALFORNIA	3
5	CALIFORNIA	58	CALIFORANIA	1
6	TEXAS	48	TEAS	3
7	TEXAS	48	TEXAYS	1
8	APARTMENT	32	PARTMENT	2
9	APARTMENT	32	APARTMENTS	1

Research is continuing on the development of Phase IV to make more accurate token corrections (standardization) at the cluster level. If it can be demonstrated the clusters produced by Phase III are reasonably accurate, then the criteria for identifying misspellings described for Phase II can be more aggressive when applied at the cluster level than at the file level. For example, while the replacements shown in Table I at the file level risks overwriting valid tokens, the same replacement is more likely to be valid within a cluster of references believed to be for the same person. Changes at the cluster level could also be applied to numeric tokens. For example, if five out of six references in a cluster have the token "413", and the sixth reference has “431”, and all six instances are preceded and followed by the same token, then it is not unreasonable to assume “431" is a mistyped version of “413”.

C. Phase III – Clustering (ER)

The purpose of Phase III is to cluster records for the same entity in support of data deduplication and data integration. This phase is more complex than Phases II and III and involves iterating over the tokenized source records coming out of Phase II. The clustering phase is a series of 13 processes labeled P1 through P13.

1) Process P1: Tokenize and compute token frequencies: Because the references have already been tokenized in Phase I, the re-tokenization here is simply a matter of separating the reference identifier and splitting the remaining substring by blank (white) space. While computing token frequencies is redundant with the same process in Phase II, for experimental purposes this was done to make Phase II an optional process allowing the evaluation of data integration results with and without token replacement.

2) Process P2: Tokenizing references and appending blocking tokens: Process P2 is the start of an iterative process on the “reprocess file”. Initially, the reprocess file is a copy of the original input file from which the frequency dictionary was created in Process P1. However, as the POC progresses, the reprocess file becomes a smaller and smaller subset of the original input source until there are there no more references to the process ending the iterations.

Process P2 repeats the tokenization process described in Process P1 in which each reference is split into a list of tokens. However, Process P2 has access to the token frequency dictionary previously build in P1. Process P2 has two primary functions:

- To rebuild each input reference as a string of blank-separated tokens, omitting all tokens found to have a frequency above the stop word frequency threshold (σ) creating “skinny references.”
- To output a copy of the skinny reference for each blocking token found in the reference.

Again, a blocking token is simply any token with a frequency below the blocking frequency threshold β. This means the output from P2 will have more records than the input assuming almost all references have at least one blocking token, and many have more than one.

Example: Suppose an input reference has the form

R13, John Doe, Oak St, Anyville AL, 793-1234

The tokenization of this reference would produce 9 tokens “R13”, “JOHN”, “DOE”, “OAK”, “ST”, “ANYVILLE”, “AL”, “793”, and “1234” (using Splitter tokenization). Also, suppose the tokens “JOHN”. “DOE”, and “OAK” have a frequency below β, and the tokens “AL”, “ST”, and “793” have a frequency above σ. Then P2 will generate three outputs.

R13: JOHN: JOHN DOE OAK ANYVILLE 1234

R13: DOE: JOHN DOE OAK ANYVILLE 1234

R13: OAK: JOHN DOE OAK ANYVILLE 1234

Because the input reference R13 contains three blocking tokens, P2 will output three skinny references, one for each blocking token. To simplify parsing, the output reference is divided into three segments using the colon (:) character. The first segment is the reference identifier, the second the blocking token, and the third the body of the reference.

3) Process P3: Sorting by blocking tokens to create blocks: The purpose of Process P3 is to sort the output of the reference from process P2 into ascending order by blocking token (Segment 2 of the rebuilt references). Each sequence of consecutive references with the same blocking token will form a block for input to the ER process for record linking.

4) Process P4: Iterate blocks: Process P4 is the start of an iterative process (P5) to be performed on each block. P4's primary function is to detect the sequences of consecutive records having the same blocking token, then pass this block of references on to P5.

5) Process P5: Link reference pairs in blocks: In Process P5, each block undergoes a process to generate pairs of linked references. The technique implemented in the POC is to use a multi-token comparator. Every pair of references in the block is compared. For a block of N references, there will Nx(N-1)/2 pairs.

Any pairwise matching process can be inserted at this point including machine learning (ML) algorithms for linking. Because the entity references are text, this approach usually requires an additional process to convert references from the text to numeric vectors, a process called text embedding. Some results from using the DBScan clustering algorithm with doc2vec text embedding are shown in this paper (Table III).

Most of the work described here used the scoring matrix. In this case, a variation of the Monge-Elkan method [21] for comparing multi-token values, but with the removal of stop words. When the scoring matrix processes a pair of references, each reference is first transformed into a list of tokens (words), then the stop word tokens are removed from the list. The remaining tokens from the first reference are used to label the rows of the matrix, and the remaining tokens from the second string label the columns of the matrix. The cell value of the matrix is a normalized similarity measure, i.e., a value in the interval [0,1], between the two tokens. In the POC, the normalized Damerau-Levenshtein Edit Distance (nLED) function was used.

To illustrate the operation of the scoring matrix, consider the following two references:

A045, Smith, John, Apt 21, 345 Oak St, Anytown, NY

B167, Jon Smith, 345 Oak Street #21, Anytown, NY

Furthermore, suppose the threshold for the comparator has been set to 0.80, and the list of stop words contains the token "NY." The resulting token matrix would then appear. The process begins by finding the largest similarity value in the matrix. This value is the initial value of a total running value. After the largest similarity value is used to initialize the total value, all of the values in the same row and column are removed (set to zero). In the next iteration, the largest similarity value from the remaining values in the matrix is identified and added to the overall total.

Again, all of the nLED values in the same row and column as the largest value are removed. The process continues in subsequent iterations until all of the similarity values have been removed from the matrix. In Fig. 1, the cells with underlined and bold font are the surviving similarity scores from this process. After the last iteration, the running total is divided by the number of iterations. If the calculated average value is greater than or equal to a threshold value provided by the user, then references are linked. At the end of the algorithm, the final matrix score for a pair of references in Fig. 1 is 0.83.

	JON	SMITH	345	OAK	STREET	21	ANYTONW
SMITH		**1.00**			0.17		0.14
JOHN	**0.75**			0.25			0.14
APT		0.20			0.17		0.29
21						**1.00**	
345			**1.00**				
OAK				**1.00**			0.14
ST		0.40			**0.33**		0.14
ANYTOWN	0.29	0.14		0.14			**0.71**

Fig. 1. Example Scoring Matrix (Zero Similarity Values Omitted).

6) Process P6: Linked pair generation: The purpose of Process P6 is to form the graph edges between pairs of references in the same cluster. Because the clusters are all formed from references in the single token block, they only represent the connections found between references sharing the token forming the block.

7) Process P7: Post-resolution transitive closure: Unlike traditional match key blocking, frequency-based blocking does not produce a true partition of the input references where each input reference is in one, and only one, block. In frequency-based blocking, each reference is replicated by the number of blocking tokens it contains as in the example for Process P3. To create the final set of clusters, in which each reference occurs in one, and only one, cluster, the clusters created from the blocks must be merged and undergo a transitive closure process.

The POC implements a very efficient sorting closure process described by Kolb et al [22]. While the sorting transitive closure is implemented in the POC as an in-memory, Java application, the algorithm is a highly-scalable, map/reduce process for execution in the Hadoop Distributed File System (HDFS) environment.

8) Process P8: Iterate clusters: Process P8 transforms the transitive closure output into clusters of linked references. Because the output of the sorting closure process is already in sorted order by cluster identifiers, the clusters are simply groups of consecutive references with the same cluster identifier.

9) Process P9: Entropy Calculation: Process P9 uses a variation of the Shannon entropy calculation [23] to assess the level of organization in each cluster of two or more references. The formula for the calculation of the entropy of a cluster is

$$\mathrm{E} = -\sum_{j=1}^{N} p(t_j) \cdot \log_2 p(t_j) \quad (1)$$

Where t_j is the j-th vertical token group in the cluster, and $p(t_j)$ is the probability of t_j.

For the POC, a vertical token group is defined to be the same token counted only once in each reference of a cluster. Thinking of the cluster as a matrix where the references are the rows and the columns are the tokens, then a vertical token group is a vertical grouping of the same token across different references. However, each token is counted only once in each reference. This means the maximum size of a vertical token

group is equal to the number of references in the cluster. The probability of a vertical token group is the size of the token group divided by the number of references in the cluster. For example, consider the following cluster of 3 references.

R1: JOHN GRANT 123 GRANT ST

R2: MARY GRANT 21 OAK STREET

R3: MARY GRANT 21 OAK ST

The first vertical token group is for the token "JOHN" which only occurs once in R1 forming a vertical token group of size 1 with a group probability of 1/3. The second vertical token group is for "GRANT" which has 3 tokens, one token each from R1, R2, and R3 giving this group a probability of 1.0 (3/3). The second "GRANT" in R1 is not part of this token group because each token is only counted once in each reference. The token group for "123" has a probability of 1/3, the second "GRANT" group has a probability of 1/3, and the "ST" group a probability of 2/3.

After exhausting all of the tokens in R1, there are still four uncounted tokens in R2 forming the "MARY" group with probability 2/3, "21" group probability 2/3, the "OAK" group probability 2/3, and the "STREET" group probability 1/3. Finally, there are no remaining uncounted tokens in R3. In total, there are 9 vertical token groups in the example cluster. The total entropy of the cluster is calculated from Formula (2) by:

$$E = -(\frac{1}{3}\cdot log\left(\frac{1}{3}\right) + 1\cdot log(1) + \frac{1}{3}\cdot log\left(\frac{1}{3}\right) + \frac{1}{3}\cdot log\left(\frac{1}{3}\right) + \frac{2}{3}\cdot log\left(\frac{2}{3}\right) + \frac{2}{3}\cdot log\left(\frac{2}{3}\right) + \frac{2}{3}\cdot log\left(\frac{2}{3}\right) + \frac{2}{3}\cdot log\left(\frac{2}{3}\right) + \frac{1}{3}\cdot log\left(\frac{1}{3}\right)) \; E = 3.67318$$

Entropy is a measure of the organization of a cluster in terms of having similar tokens [24]. The entropy of a cluster decreases as references in a cluster have more and more similar tokens. By this measure, a cluster will have an entropy of 0 if, and only if, all of the references have the same tokens.

10)Process P10: Assessment of clusters based on entropy: In Process P10, the entropy of each cluster as calculated in Process P9 is assessed against the user-defined entropy threshold ε. If a cluster has an entropy less than ε, it is judged as an acceptable cluster, and the reference identifier and cluster identifiers from the clusters are written to the Saved Clusters output file. Otherwise, the cluster identifiers are discarded, and the references are written to the Reprocess file. References written to the Reprocess file will go through the entire blocking and ER process again but at a higher match threshold. By definition, all clusters of size one (singleton clusters) have an entropy of 0 and are written directly the Saved Clusters file.

For each cycle of the POC, the size of the Saved Clusters file increases while the size of the Reprocess file decreases. The Reprocess file will eventually become empty as the match threshold μ approaches 1.0. At very high match thresholds, the references in a block can only form clusters if they are highly similar and generate clusters of very-low entropy, otherwise, they break down into singleton clusters. In either case, they will eventually pass to the Saved Clusters file and the Reprocess file will be empty. An example of this process is shown in Table I. The statistics are produced as part of the statistics report when running a sample. In this case, the statistics are for Sample S4 of 1,912 references. As shown in Table IV, the parameters for this run were β=12, σ=22, ε=4.2, and the starting value of μ=0.5.

The volume of work continually decreases with each iteration. Note that some references written to the reprocess file will not be used in the next iteration. This is because, at the beginning of the next iteration, the reprocess file is re-blocked and re-clustered. During the clustering process, reference-to-reference links are only produced for references linked to at least one other reference. For example, 27 references were written to the reprocess file at the end of the μ=0.8 iterations, but only 14 of these references survived to form 6 clusters of two or more references when the match threshold μ was increased to 0.90.

11)Process P11: Reprocess decision: As described in Process P10, at some point the Reprocess file will be empty. When this happens, the reprocessing cycle stops, and the final join (Process P13) is performed.

12)Process P12: Increasing Match Threshold: If there are references to be reprocessed, then the match threshold is increased before the reprocess is started. Increasing the match threshold will require references to be more similar before they are linked into the same cluster. In all of the results reported here, the increment value was 0.1 (10%).

13)Process P13: Final join to original source: Although no further iterations are necessary when the Reprocess file is empty, there are still two tasks to complete. The first task is to ensure every reference in the source is represented in the final set of clusters. Some references in the source may not be transferred to the Saved Clusters file. Depending upon the value of blocking frequency threshold β, some references may not contain blocking tokens and are not output from Process P2.

The second task is to append the final cluster identifier to each reference in the source. The goal is to create a Final output comprising every reference in the source along with its proper cluster identifier. Both of these tasks can be completed by performing an outer join by reference identifiers between the original Reference Source file and the Saved Clusters file.

D. Cluster Cleaning

While this process has not been implemented in the POC, work is currently underway to develop unsupervised techniques for cleaning and standardizing tokens within the same cluster. The current approach is very much the same as the Global Token Replacement described in the Second Phase. However, replacements can be more aggressive at the cluster-level versus the file level. Across an entire reference file, there could easily be an entire sequence of house numbers, such as 123, 124, 125, and so on. For this reason, numeric tokens are specifically excluded from replacement globally. However, at the cluster level, it is much more probable that if 5 of 6

references have the token 123 and the sixth reference has 124, the replacement of 124 by 123 would be a correction.

III. POC Test Samples and Results

To test the POC, 18 samples were taken from four fully annotated reference sources. Aside from having equivalent references to match, the samples also exhibited combinations of two other characteristics – high data quality (DQ=Good) versus low data quality (DQ=Poor) and uniform record layout (Mixed=No) versus mixed (Mixed=Yes) record layout. In all cases, the Splitter Tokenization Method with token de-duplication was used in Phase I. To gauge the effect of Phase II (Global Token Replacement) each sample was run with, and without, the global token replacement. In the cases where token replacement was run, the settings described in Section on the Second Phase the parameters were fixed at

- MinFreqStdToken =5
- MaxFreqErrToken =3
- MinLenStdToken = 5
- MaxStringDiff = 1

To establish a baseline, all samples were run with 0.50 as the initial value of μ and 0.10 as the increment value forμ. The initial values for β and σ were set using the linear regression prediction formulas (3) and (4) for the non-iterative model [25]. However, the actual values for β, σ, and ε were set manually by observing the correlation between the F-measure of each cluster and the computed entropy as logged by the system (Table IV). Then exploring a range of values around these estimates using a grid search automated with a robotic Python process to run each range of settings and collect the precision, recall, and F-measure results. The results for all 18 samples using the best parameter settings are given in Table IV.

A. Samples with Good Data Quality

Stratified samples S1, S2, S4, S5, S7, S13, S14, and S15 were drawn from a corpus of approximately 800K references created using the R-package "generator", and degraded with data quality errors using the R-package "relErrorGeneratoR" from GitHub.com. While some reference-level errors such as misspelling, truncation, mixed formatting, and missing values were injected into the data during generation, the individual references in the 800K corpus are of relatively high quality.

The majority of the data quality errors introduced into the 800K corpus were data redundancy (duplicate record) errors to make the corpus more useful for entity resolution research. Shown here are two references from Sample S4 with Record Layout A. The only variations between the two references are the name truncation (initial) and different formats for telephone numbers and identification numbers.

A926344: ANDREW, AARON, STEPHEN, 2475 SPICEWOOD DR, WINSTON SALEM, NC, 27106, 601-70-6106, (159)-928-5341

A930444: A, AARON, STEPHEN, 2475 SPICEWOOD DR, WINSTON SALEM, NC, 27106, 601706106, (159)9285341

Sample S6 was produced by the GeCo synthetic data generator [26], and Sample S3 is a file of 866 references to restaurants (businesses) from two public sources, Zagat's and Fodor's restaurant guides. The references contain restaurant names, addresses, city, phone, and type of cuisine. The file has been manually annotated and is known to have 112 pairs of equivalent references [27]. Examples of references from S3 are shown here.

A001: Arnie Morton's of Chicago 435 S. La Cienega Blvd. Los Angeles 310-246-1501 Steakhouses

A002: Arnie Morton's of Chicago 435 S. La Cienega Blvd. Los Angeles 310/246-1501 American

B. Low Data Quality Samples

Samples S9, S10, S11, S12, S16, S17, and S18 were taken from the SOG (Synthetic Occupancy Generator) project [28]. The SOG corpus has approximately 270K references with three different record layouts A, B, and C. The SOG corpus has a much higher level of data quality errors than the 800K corpus. Most records exhibit at least one error such as missing value, misspelling, truncation, inconsistent formatting, nicknames, and name and address changes. Shown here are three equivalent references from Sample S8 exhibiting a number of these data quality issues.

A960175,lucia,r,oster,t20672,southwood,oaks,dr,porter,,tx, 77365,,,10896980,,

A966807,lucia,r,wilson,12006,MOUNTAIN,RIDGE,RD,HOUSTON,,TEXAS,77043,PO,BOX,280034, houston,,tx,77228,10896980,1917

A971069,LUCIA,R,WILSON,20672,SOUTHWOOD,OAKS,DR,PORTE,,TEXAS,77365,,,001-89-6980,,

C. Mixed Layout Samples

In addition to variations in quality, Sample S7 and Samples S10 through S18 were selected with mixed (heterogeneous) record layouts. For example, in Sample S7 about half of the references were in Record Layout A and the other half in Record Layout B. The two layouts used a different order for names and have different identity attributes, e.g. social security number in Layout A and date-of-birth in Layout B. An example of a pair of references from S7 is shown here.

A944353,VICTOR,AGWU,KINGSLEY,1608 W NORTHWEST BLVD # O,WINSTON SALEM,NC,27104.0,730-69-2869

B867674,VICTOR K AGWU,1608,W NORTHWEST BLVD # O,WINSTON SALEM,NC,27104,12/14/37,

For samples selected from the SOG corpus, the difference in layouts was much greater. Here is an example of a pair of references from Sample S10.

A993286,chavez,4149 WALSH LN,GRAQND PRAIRIE, TEXAS 75052,OFFICE BOX 54331,grand prairie, tx 75054,10525947,,,

A994281,barbie chavze,11R881 GULF POINTE DR APT E38,HOUSTON, TX 77089,,,,,,,

B904140,Barby ,CHAVEZ ,11881,GULF POINTE DR ,Apt E38,HOUSTON,TX,77089,(713)165.7474,,

As noted, the values for β, σ, ε, and the starting value of μ were found by a grid search. Prior research in using the scoring matrix for ER on samples from these same corpora [25] [29], [30], [31] provided some guidance of the best values for the blocking threshold frequency threshold β and the stop word frequency threshold σ based on the size of the sample and the standard deviation of its token frequency distribution. These formulas are

$$\beta = (7.8864) + (0.0023)*Std\_Dev + (0.0005)*Size \quad (3)$$

$$\sigma = (-83.3106) + (2.9647)*Std\_Dev + (0.0037)*Size \quad (4)$$

However, the previous research did not involve the entropy-based, self-evaluation, or iteration with incrementally increasing match thresholds use in this research, thus it did not provide any guidance about the best setting for the entropy threshold ε. Instead, the estimated value for ε was found by observing the entropy measure of each cluster and comparing it to the actual F-measure of the cluster. This was possible because all of the test samples were fully annotated. The F-measure assessment of each cluster was an augmentation to Processes P5. As the entropy of each cluster is calculated, the cluster was also sent to an ER metrics program to determine the actual F-measure of the cluster as compared to the annotated truth set.

The entropy and the actual F-measure of each cluster were captured in a Cluster Analysis text file. Table II shows a segment of the report produced when running Sample S2. The table shows the results of three iterations. Row 1 of Table II shows the last cluster produced by the initial value μ at 0.5, Rows 2 through 6 show the entire second reprocess iteration of four clusters where the value μ was 0.6. Row 7 is the first cluster of the last iteration where μ was 0.7. As each cluster is formed, its entropy is calculated as shown in the column labeled "Entropy." If the cluster's entropy is above the value of ε (set at 4.3 for this run) the cluster is judged to be "bad" and is written to the reprocess file for re-linking at the next higher value of the match threshold μ.

On the other hand, if the entropy is less than or equal to ε, it is written to the "good" file as a final cluster. Table II shows for Rows 4 and 5 these were correct decisions. In both cases, the F-Measure was less than 1.0 when the entropy was above 4.3. However, Row 2 is an exception. Even though the entropy of 9.0446 is above 4.3, the cluster had an F-Measure of 1.0 and was correctly linked. However, because the entropy was above the threshold, the references were put back for reprocessing in the third iteration. In the end, the F-Measure for S2 at the end of the process was 0.8842 as shown in Table II.

TABLE II. SEGMENT OF ENTROPY VS. F-MEASURE REPORT FOR S2

Row	μ	ε	Size	Entropy	F-Meas	Precision	Recall
…							
1	**0.5**	**4.3**	**2**	**0.00**	**1.0**	**1.0**	**1.0**
2	**0.6**	**4.3**	**3**	**9.04**	**1.0**	**1.0**	**1.0**
3	**0.6**	**4.3**	**2**	**1.00**	**1.0**	**1.0**	**1.0**
4	**0.6**	**4.3**	**3**	**5.53**	**0.5**	**0.33**	**1.0**
5	**0.6**	**4.3**	**4**	**5.50**	**0.5**	**0.33**	**1.0**
6	**0.7**	**4.3**	**3**	**9.04**	**1.0**	**1.0**	**1.0**
7	**0.7**	**4.3**	**2**	**2.00**	**1.0**	**1.0**	**1.0**
…							

D. Example Results using Machine Learning for P5

In this example, Sample S4 was processed using DBScan (Density-Based Spatial Clustering of Applications with Noise) [32] as the ML clustering algorithm and using the doc2vec [33] word embedding algorithm to create the numeric vectors as input for DBScan. As implied by its name, the doc2vec algorithm converts an entire document into a vector. For the POC, each reference was considered a document so there is a one-to-one correspondence between each input reference and each vector clustered by DBScan.

The doc2vec algorithm was applied to each block using the following parameters.

- vector size = 5
- min count = 2
- epoch = 20
- alpha = 0.25
- min_alpha=0.00025

DBScan algorithm was imported from the Python 3.7 library learn. cluster. This version has two control parameters "eps" and "min_samples". The eps parameter controls the neighborhood reach (proximity) of vectors to be in the same cluster, and the min_samples parameter defines the minimum size of "core samples", i.e. the minimum number of vectors within eps distance of each other. The results from using this configuration are shown in Table III.

TABLE III. ER RESULTS USING DOC2VEC FOLLOWED BY DBSCAN

Sample	Size	eps	min_ samples	Results		
				Precision	Recall	F-Measure
S1	50	0.6	0.9	0.8519	1.0000	0.9200
S2	100	0.4	1	0.9070	1.0000	0.9513
S4	1,912	0.01	1	0.9555	0.9111	0.9328

TABLE IV. SHOWS THE RESULTS FROM THE POC

ID	Size	Token	DQ	Mix	Mu	Mu+	Beta	Sigma	Epsilon	Precision	Recall	F-Measure
S1	50	Split	Good	No	0.50	0.10	6	7	4.9	1.00	0.963	0.9811
S2	100	Split	Good	No	0.50	0.10	6	7	4.7	1.00	0.8958	0.9451
S3	868	Split	Good	No	0.50	0.10	9	95	4.0	0.8889	0.9286	0.9083
S4	1,912	Split	Good	No	0.50	0.10	12	22	3.1	0.9854	0.8869	0.9335
S5	3,004	Split	Good	No	0.50	0.10	12	53	3.0	0.9911	0.8729	0.9282
S6	19,998	Split	Good	No	0.50	0.10	35	403	15.1	0.9457	0.9737	0.9595
S7	3,000	Split	Good	Yes	0.50	0.10	14	24	3.0	0.9464	0.8665	0.9047
S8	1,000	Split	Poor	No	0.60	0.05	14	145	35	0.7880	0.8881	0.8350
S8	1,000	Comp	Poor	No	0.60	0.05	14	144	33	0.7877	0.8827	0.8324
S9	1,000	Split	Poor	No	0.50	0.05	15	135	28	0.6824	0.8453	0.7552
S9	1,000	Comp	Poor	No	0.62	0.02	23	150	28	0.7806	0.8116	0.7958
S10	2,000	Split	Poor	Yes	0.50	0.05	31	280	31.5	0.7235	0.8348	0.7752
S10	2,000	Comp	Poor	Yes	0.62	0.02	32	280	34	0.7455	0.9041	0.8172
S11	4,000	Split	Poor	Yes	0.55	0.02	38	280	31.5	0.7010	0.8386	0.7636
S11	4,000	Comp	Poor	Yes	0.62	0.02	43	258	26	0.7571	0.7764	0.7666
S12	6,000	Split	Poor	Yes	0.50	0.05	20	580	29	0.7699	0.7424	0.7560
S12	6,000	Comp	Poor	Yes	0.62	0.02	21	570	26.4	0.7825	0.7723	0.7774
S13	2,000	Split	Good	Yes	0.50	0.05	14	23	5.3	0.9478	0.8116	0.8745
S13	2,000	Comp	Good	Yes	0.01	0.02	14	110	2.4	0.9707	0.8003	0.8773
S14	5,000	Split	Good	Yes	0.48	0.02	24	118	5.2	0.9419	0.8464	0.8916
S14	5,000	Comp	Good	Yes	0.01	0.02	24	310	2.6	0.9579	0.8226	0.8851
S15	10,000	Split	Good	Yes	0.50	0.02	20	100	5	0.9543	0.8242	0.8845
S15	10,000	Comp	Good	Yes	0.1	0.03	27	108	2.8	0.9622	0.8247	0.8882
S16	2,000	Split	Poor	Yes	0.50	0.05	21	142	32.8	0.6908	0.8160	0.7666
S16	2,000	Comp	Poor	Yes	0.54	0.02	30	154	31.3	0.6500	0.9331	0.7663
S17	5,000	Split	Poor	Yes	0.50	0.05	35	480	32.8	0.6954	0.81578	0.7508
S17	5,000	Comp	Poor	Yes	0.60	0.02	26	466	26	0.7560	0.80716	0.7807
S18	10,000	Split	Poor	Yes	0.50	0.05	33	449	31.8	0.6995	0.7514	0.7245
S18	10,000	Comp	Poor	Yes	0.52	0.02	26	444	34	0.7511	0.7948	0.7724

IV. CONCLUSION AND FUTURE RESEARCH

The results are shown in Table IV suggest entropy can be an effective way to regulate an unsupervised clustering process. The POC using the scoring matrix performs extremely well when processing good quality references such as Samples S1 – S7 and S13 – S15. The average F-Measure for these samples was 0.9124 with an average precision of 0.9609.

The average F-Measure for the poor quality samples S8 – S12 and S16 – S18 was somewhat lower at with an average F-Measure of 0.7772 and precision of 0.7351. The results also indicate the POC is more sensitive to data quality issues than to mixed record formats. The good-quality, mixed-format Samples S7, S13, S14, and S15 had an average F-Measure of 0.8866 compared to an average F-Measure of 0.9426 for good-quality, single format samples.

For the good quality samples where the clustering precision was 96%, the hope is that applying a more comprehensive cleaning and standardization at the cluster level will be able to provide much better results. The goal for future research is, just as linking results can be continually improved through iterative reprocessing, the same reprocessing loop will also incorporate processes to continually improve the quality of the references, which in turn, would further improve the linking results. The POC described in this paper shows the unsupervised ER improvement part of this positive feedback loop is feasible. The next step will be to integrate additional unsupervised data quality improvement processes.

A. Industry Testing

As an experiment, a commercial company tested the POC (data washing machine) approach using a real-world dataset of 70,500 business names and address references with mixed record layouts. Because the dataset was not annotated, it was

not possible to calculate the exact F-measure of the overall clustering results. However, the company did undertake an extensive manual review of the POC results in comparison to results from their standard process. The company determined the POC results to be as good as, and in many cases better than, the results from their standard process, but with the added advantage of avoiding the time and effort to analyze and prepare the data required by their standard process.

Besides, the company is experimenting with some variations of the original POC design described in this paper. In particular, they have been able to improve the clustering accuracy for their datasets by using a computed value for ε, the entropy threshold. In their approach, they consider five factors when assessing the entropy of each cluster. These are

- The match threshold μ used to form the cluster.
- The numbers of references in the cluster (size).
- The maximum number of tokens in any one reference in the cluster (maxT).
- The minimum number of tokens in any one reference in the cluster (minT).
- The average number of tokens for all references in the cluster (avgT).

In Process P10, instead of comparing the entropy of the cluster to a static value of ε as in the original POC, they compute a dynamic threshold based on the factors listed above. In particular, ε is computed as

$$\varepsilon = \log(1 + sizeC) + mu \cdot \{\log(avgT) + \log(1 + maxT - avgT) + \log(1 + maxT - minT)\} \quad (5)$$

In another change, they were able to improve the precision of the clustering by modifying the scoring matrix used to link references in Process P5. The comparator was modified to use a Boolean similarity of the match (1.0) and no-match (0.0) when comparing numeric tokens while still using the normalized Damerau-Levenshtein edit distance when comparing non-numeric tokens.

B. Predicting Parameters and Scalability

However, there are still two gaps that must be bridged to make the POC a practical solution for most real-world use cases. The first is a reliable method for setting the optimal values of the key parameters β, σ, and ε. In a research environment using fully annotated references, these values can be found by simply observing where the best results were obtained based on comparisons with the correct linking. When working with real data, this is not generally possible. A practical unsupervised ER system needs a way to predict these parameters for a given set of input references. Creating such predictive models is still research in progress.

The second consideration is scalability. The current POC is implemented in a combination of Python and Java, and as written, it is not very scalable. The blocking and the stop word removal process can be combined with the token counting process to avoid the need for storing an in-memory token frequency table.

The POC can be converted to an HDFS Map/Reduce process. The references can easily be tokenized in the mapping process which then reduces on the token. The reducer can then emit two kinds of key-value pairs for each token group. The first is (RefID, Token) where the token has a frequency below σ (not a stop word). The second is (Token, RefID) where the token has a frequency below β (a blocking token). Sorting and reducing the first pairs on the RefID as the key will create the skinny references of Process P2 while sorting and reducing the second pairs on Token as the key will create the blocks. The join of these two outputs on RefID will be the equivalent of creating and sorting the blocked file in Process P3. Next, the blocks can be mapped to distributed nodes for pairwise linking in parallel with the assurance no block will be larger than β. The outputs are the Process P6 Linked Pairs. The transitive closure of the pairs in Process P7 using the algorithm of Kolb et al [22] is already an efficient map/reduce process. Process P8 then becomes a map of the clusters to parallel processing work nodes performing the entropy calculation (Process P9) and triage of clusters (Process P10) into "good" and "bad" cluster outputs.

The POC described here was built uses the simplest of approaches which could no doubt be dramatically improved through additional research and experimentation including investigating different starting values forμ, and exploring its sensitivity to the increment value currently fixed at 0.1. Also, building prediction models for these parameters. Another is investigating whether the results are improved by modifying the values of β, σ, or ε for each reprocesses iteration, and if it does, how should they be modified to produce the best linking results.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Award No. OIA-1946391 and by the PiLog Group. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the PiLog Group.

REFERENCES

[1] A. Saeedi, E. Peukert and E. Rahm, "Incremental Multi-source Entity Resolution for Knowledge Graph Completion," German Federal Ministry of Education and Research, Leipzig, Germany, 2020.

[2] M. Stonebraker and I. F. Ilyas, "Data Integration: The Current Status and the Way Forward," Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, pp. 3-9, 2018.

[3] R. Yuji, H. Geon and E. W. Steven, "A Survey of Data Collection for Machine Learning: A Big Data – AI Integration Perspective," IEEE Transactions on Knowledge and Data Engineering, 2019.

[4] F. Azzalini, S. Jin, M. Renzi and L. Tanca, "Blocking Techniques for Entity Linkage: A Semantics-Based Approach," Data Science and Engineering, pp. 1-19, 2020.

[5] A. Alsarkhi and J. R. Talburt, "A method for implementing probabilistic entity resolution," International Journal of Advanced Computer Science and Applications, vol. 9, no. 11, pp. 7-15, 2018.

[6] P. G. Ipeirotis, V. S. Verykios and A. K. Elmagarmid, "Duplicate record detection: A survey," IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 1, pp. 1-16, 2007.

[7] Y. Roh, G. Heo and S. E. Whang, "A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective," IEEE

Transactions on Knowledge and Data Engineering, vol. Early Access, 2019.

[8] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis and K. Stefanidis, " An Overview of End-to-End Entity Resolution for Big Data," ACM Computing Surveys (CSUR), vol. 53, no. 6, pp. 1-42, 2020.

[9] P. Neoklis, R. Sudip, E. W. Steven and Z. Martin, "Data Lifecycle Challenges in Production Machine Learning," A Survey, ACM SIGMOD, vol. 47, no. 2, 2018.

[10] B. J. Dooley, "How Robotic Process Automation Eases Data Management, The Data Warehouse Institute (TDWI)," 18 June 2018. [Online]. Available: https://tdwi.org/articles/2018/06/18/diq-all-how-robotics-process-automation-eases-data-management.aspx.

[11] J. Adams, ", Automating Data Management and Governance through Machine Learning, The Data Administration Newsletter (TDAN)," 7 November 2018 . [Online]. Available: https://tdan.com/automating-data-management-and-governance-through-machine-learning/23972.

[12] R. Pita, L. Menezes and M. Barreto, "Applying Term Frequency-Based Indexing to Improve Scalability and Accuracy of Probabilistic Data Linkage," In LADaS@ VLDB, pp. 65-72, 2018.

[13] A. Moshyedi, T. Kramer, A. Gangopadhyay and S. Pal, "A combined semantic search and machine learning approach for address entity resolution," EasyChair, 2019.

[14] R. Y. Wang, J. R. Talburt and Y. W. Lee, "A framework for analysis of data washing machines," http://mitiq.mit.edu, Cambridge, MA, 2020.

[15] A. Jurek-Loughrey and P. Deepak, ""Semi-supervised and unsupervised approaches to record pairs classification in multi-source data linkage," In Linking and Mining Heterogeneous and Multi-view Data, pp. 55-78, 2019.

[16] Y. Lee, L. Pipino, J. Funk and R. Wang, Journey to Data Quality, MIT Press, 2006.

[17] U. Draisbach, P. Christen and F. Naumann, "Transforming pairwise duplicates to entity clusters for high-quality duplicate detection," Journal of Data and Information Quality (JDIQ), vol. 12, no. 1, pp. 1-30, 2019.

[18] P. Christen, "A survey of indexing techniques for scalable record linkage and deduplication," IEEE transactions on knowledge and data engineering, vol. 24, no. 9, pp. 1537-1555, 2011.

[19] A. Ardalan, A. Doan and A. Akella, "Smurf: Self-service string matching using random forests," Proceedings of the VLDB Endowment, vol. 12, no. 3, pp. 278-291, 2018.

[20] A. Mazeika, M. H. Böhlen, N. Koudas and D. Srivastava, "Estimating the selectivity of approximate string queries," ACM Transactions on Database Systems (TODS), vol. 32, no. 2, pp. 12-es., 2007.

[21] A. E. Monge and C. P. Elkan, "The Field Matching Problem: Algorithms and Applications," in KDD-96 Proceedings, 1996.

[22] L. Kolb, S. E. and E. Rahm, "ITerative Computation of Connected Graph Components with MapReduce," Datenbank-Spektrum, vol. 14, no. 2, 2014.

[23] C. E. Shannon, "A Note on the Concept of Entropy," Bell Systems Technical Journal, 1948.

[24] D. Lee, L. C. Zhang and J. K. Kim, "Maximum Entropy classification for record linkage," arXiv preprint arXiv:2009., p. 14797, 2020.

[25] A. Al-Sarkhi and J. R. Talburt, "Estimating the Parameters for Linking Unstandardized References with the Matrix Comparator," Journal of Information Technology Management, pp. 12-26, 2019.

[26] K. N. Tran, D. Vatsalan and P. Christen, "GeCo: an online personal data generator and corruptor," the 22nd ACM International Conference on Information & Knowledge Management, pp. 2473-2476, (2013, October).

[27] T. S, "Restaurant Benchmark Dataset," [Online]. Available: http://www.cs.utexas.edu/users/ml/riddle/data.html].

[28] J. R. Talburt, Y. Zhou and S. Y. Shivaiah, "SOG: A Synthetic Occupancy Generator to Support Entity Resolution Instruction and Research," MIT International Conference on Informationi Quality, pp. 91-105, 2009.

[29] A. Alsarkhi and J. R. Talburt, "Optimizing Inverted Index Blocking for the Matrix Comparator in Linking Unstandardized References," in ," in in Proceedings of the 2019 International Conference on Scientific Computing, Las Vegas, 2019.

[30] A. Al Sarkhi and J. Talburt, "An analysis of the effect of stop words on the performance of the matrix comparator for entity resolution.," The Journal of Computing Sciences in Colleges, vol. 34, no. 7, pp. 64-71, 2019.

[31] A. Al Sarkhi and J. Talburt, "A Scalable, Hybrid Entity Resolution Process for Unstandardized Entity References," The Journal of Computing Sciences in Colleges, pp. 19-29, 2020.

[32] M. Ester, H.-P. Kriegel, J. Sander and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in Proceedings of Second International Conference on Knowledge Discovery and Data Mining, Portland, OR,, 1996.

[33] J. H. Lau and T. Baldwin, "An empirical evaluation of doc2vec with practical insights into document embedding generation," arXiv preprint arXiv:1607.05368, 2016.