ModER: Graph-based Unsupervised Entity Resolution using Composite Modularity Optimization and Locality Sensitive Hashing

Islam Akef Ebeid, John R. Talburt, Nicholas Kofi Akortia Hagan, Md Abdus Salam Siddique
Department of Information Science
University of Arkansas at Little Rock
Little Rock, Arkansas

***Abstract*—Entity resolution describes techniques used to identify documents or records that might not be duplicated; nevertheless, they might refer to the same entity. Here we study the problem of unsupervised entity resolution. Current methods rely on human input by setting multiple thresholds prior to execution. Some methods also rely on computationally expensive similarity metrics and might not be practical for big data. Hence, we focus on providing a solution, namely ModER, capable of quickly identifying entity profiles in ambiguous datasets using a graph-based approach that does not require setting a matching threshold. Our framework exploits the transitivity property of approximate string matching across multiple documents or records. We build on our previous work in graph-based unsupervised entity resolution, namely the Data Washing Machine (DWM) and the Graph-based Data Washing Machine (GDWM). We provide an extensive evaluation of a synthetic data set. We also benchmark our proposed framework using state-of-the-art methods in unsupervised entity resolution. Furthermore, we discuss the implications of the results and how it contributes to the literature.**

***Keywords*—Entity resolution; data curation; database; graph theory; natural language processing**

I. Introduction

Entity resolution is critical in data cleaning, curation, and integration [1]. It also refers to finding duplicate records within the same table, across various tables, or multiple databases that might refer to the same entity. Traditional and rule-based entity resolution relies heavily on human input to guide the entity matching process using predefined rules. Defining those rules depends on handcrafting simple lexical, semantic, and syntactic conditions for matching records based on attribute similarity, such as in [2] and in [3]. However, moving toward automating entity resolution for data cleaning, curation, and integration has become a sought-after goal in many domains. Thus, unsupervised entity resolution methods have increased.

Nevertheless, unsupervised approaches suffer from higher inaccuracies than other methods due to relying solely on approximate string matching. Approximate string matching algorithms can quantify the similarity between strings based on character or token frequency and location [4]. String similarity metrics can vary in granularity from character-based to context-based. For example, character-based approaches such as Levenshtein's Edit Distance [5], Affine Gap Distance [6], Smith-Waterman Distance [7], Jaro Distance [8], or n-gram based algorithms [9] are better suited for data where the order of the tokens matter in identifying unique entities [4]. On the other hand, token-based approaches such as Overlap, Cosine, Dice, Monge-Elkan [10], and Jaccard [11] rely on tokenizing the text into a finite set and then comparing the intersection and union between the sets, which makes them better suited for data characterized by typographical errors. The TF-IDF algorithm [12] is another type of string similarity metric, which is more context-based and depends on token frequencies in a corpus.

Unsupervised entity resolution methods typically follow an automated processing pipeline that consists of preprocessing, blocking, matching, clustering, profiling, and canonicalization. Preprocessing refers to multiple steps that involve merging and parsing data files, tokenizing, and normalizing the unstandardized documents. Blocking is the strategy used to mitigate the quadratic complexity of pairwise comparisons in unsupervised entity resolution. That strategy relies on quick and dirty techniques that divide the preprocessed unstandardized references into chunks or blocks, avoiding string matching across the whole dataset. As a result, each block can be processed separately, where pairwise string similarity can be applied with less computational cost.

Generally, unsupervised entity resolution systems resort to matching threshold setting and end the entity matching process at that stage, such as in [13]. Other systems further expand the pipeline to identify entity profiles generalizing the entity matching output to more than two entity clusters. The clustering process aims to resolve conflicts in pairwise matching and find records that indirectly match. Those conflicts typically occur due to the reliance on frequency-based blocking [1]. Thus, to increase automation, reduce the amount of human input, and increase efficiency in the unsupervised entity resolution process, we aim to reduce the number of input parameters needed and to step away from direct approaches in approximate string matching. We introduce a graph-based approach to entity profiling in unsupervised entity resolution systems that leverage graph clustering algorithms' maturity and autonomy.

More specifically, we address the following challenges in graph-based unsupervised entity resolution systems represented by [13] and iterative self-assessing systems represented by [1]:

- The processing pipeline in iterative self-assessing systems might need to be applied multiple times due to the low accuracy of relying on approximate string

matching alone without rules as in traditional methods. That is clear in approaches such as the Data Washing Machine (DWM) [1].

- Using approximate string-matching similarity measures that rely on heuristics, though providing robust results, sometimes undermines processing speed. Moreover, those algorithms are exhaustive with quadratic time complexity running time, such as in [14]. In addition, though [13] introduced a learned similarity function, the algorithm is relatively expensive and does not provide an entity profiling capability.
- Setting matching thresholds as in [13], and cluster quality thresholds as in [1] might be problematic for the user's perspective. Interpreting those thresholds depends on the fed data, and the user might not have a baseline reference to compare.

A. Contribution

Here we developed a solution that relies on exploiting the transitivity property characterized by the output of the matching process, allowing us to recast the matched documents as a graph of weighted edges. We expand on the work that our group has done, mainly the Graph-based Data Washing Machine (GDWM) [15] and the original Data Washing Machine (DWM) [1]. We address the previously mentioned problems as follows:

- ModER avoids the extra computational cost of iterating multiple times over the processing pipeline to maximize a similarity threshold and optimize a cluster quality threshold [1]. First, the recast graph is divided into smaller subgraphs using a connected component detection algorithm exploiting the transitivity property of pairwise matching. Second, a document-word bipartite graph is formed where an initial modularity optimization runs to initialize cluster memberships of document nodes on the block level subgraph. Finally, a conditional greedy modularity maximization algorithm further breaks down the detected clusters.
- Instead of costly computing token-based similarity measures, the weighted edges representing the approximated similarity between every two documents on the block level are estimated using a Locality Sensitive Hashing scheme [16]. The similarity weight is approximated using a MinHash Jaccard estimator [17]. In addition, we exploit the fact that documents or records almost always include highly discriminative terms representing a fingerprint for each document.
- Instead of relying on the user to set similarity matching and cluster quality thresholds, we overcome the need to set algorithm-related thresholds by using Modularity as an optimized cluster quality metric guiding the matching and linking processes. The only parameters that the user needs to input are data-related: the percentile of blocking words, the percentile of stop words, and the percentile of discriminative words. That allows the user to study the data before running our framework statistically. The user can then provide those parameters as a function of the statistical analysis of the dataset.
- To our knowledge, Modularity based graph clustering has not been adapted before to the problem of unsupervised entity resolution. Therefore, we make our code publicly available as a git repository through the link under the directory ModER[1].

II. Related Work

There is a large number of entity resolution systems in the literature in general targeting many problems such as ZeroER [18], DITTO [19] and Swoosh [20]. In this literature review we focus specifically on papers that are within the scope of graph-based unsupervised entity resolution. Despite their sparsity in the literature, graph-based methods and algorithms have been adapted before to entity resolution.

A. Token-based Graph Entity Resolution

In token-based graph entity resolution, the goal is to construct a bipartite undirected graph of token nodes and record nodes and cluster the record nodes into unique entities using methods such as SimRank [21]. In [22], the authors introduced a graph-based entity resolution model. The model transformed the input data set into a graph of unique tokens where connectivity reflects the co-appearance of tokens in references. The graph was clustered using a weight-based algorithm that considered three types of vertices: exemplar, core, and support vertices. The algorithm then constructed r radius maximal subgraphs from the original token graph to discover clusters related to unique entities. Token-based methods, however, are computationally expensive and memory intensive due to the lack of an integrated blocking strategy.

B. Record-Record Simiarlity-Based Graph Entity Resolution

Record-record similarity graphs link structured unstandardized references in a weighted undirected graph where the nodes represent unique records. The connectivity represents the degree of similarity between individual references. That approach of constructing a record graph allows to directly utilize a whole set of graph clustering algorithms that graph theory and network science researchers have already developed. While [23] applied a graph clustering algorithm to optimize minimal cliques in the graph. The algorithms approximated the NP-hard graph clique problem through pruning. Moreover, [24] developed the FAMER framework to combine multi-source data using blocking, matching, and clustering schemes. The framework modeled the merged data as a similarity-record graph and then leveraged graph clustering techniques to resolve the entities.

Other work has leveraged the graph's structure instead of just the weights between records. In [25], the authors proposed three algorithms to cluster the similarity graph based on structure rather than edge weights. They argue that graph-based transitive closure, such as in [26], produces high recall but low precision because the graph's structure is not considered during clustering. They justified using maximal clique algorithms to leverage the graph's structure, which increases precision. There are also centrality and node importance-based methods where the edge weights are not considered, and node scores are propagated, such as in [27]. In addition, the authors introduced

[1]https://bitbucket.org/oysterer/dwm-graph/src/master/ModER/

the notion of a node resistance score in a co-authorship graph to model entity similarity. Node resistance can be considered a PageRank score [28], where a random walker computes the probability of getting from the source node to the target node iteratively until convergence. Also, in [29], the authors introduced a graph-based model that linked two graph datasets by aggregating similarity scores from neighboring record nodes. However, record-record similarity methods are complicated and require extensive graph theory knowledge to tune the adapted methods.

C. Hybrid Graph-based Entity Resolution

Hybrid methods that combine token-based bipartite graphs and similarity-based record-record graphs have been investigated in [13] and [30]. The authors proposed an algorithm that combines text similarity with a graph-based algorithm. They first partition the data into a bipartite graph of record pair nodes and frequent term nodes to learn a similarity score of the record pair nodes. Then, the result was used to construct a record-record graph and used to power the CliqueRank algorithm, which runs on the blocks of records identified by the first part, known as the ITER algorithm. The probability of a matching pair of records is then updated iteratively. The authors combined two distinct methods: the random walk-based approach and the graph clustering-based approach. However, the authors used the graph approach to match pairs of records without introducing any clustering approach that would resolve the entity profiles.

The following section describes the framework, method, and algorithm used in ModER.

III. Preliminaries

A. Problem Definition

Let us assume that we have a collection of merged documents in one file where every single document has a unique identifier and a reference body. More formally, the set of merged documents are D consisting of tuples $d_i = (u_i, r_i)$ where $u_i \in U$ is a unique identifier that is either provided in the input file or is automatically generated by ModER and $r_i \in R$ is the reference body. Let us also assume that there exists a latent variable τ representing the underlying hidden unique entity profiles in the data file pointing to the probability $P(\tau) = \sum_{i=0}^{N} P(d_i \in \tau)$ where N is the number of documents In the file. Hence $|\tau| <= N$. That reformulates the problem as an estimation of $P(d_i \in \tau)$ for each document d_i.

B. Graph Formulation and Modularity

Consider a set of document unique identifiers u_i and their tokenized unique reference bodies r_i. Consider two ways of remodeling the input corpus as a graph. First a document-document graph $G = (V, E)$ where each vertex/node $v \in V$ represents a unique document $u_i \in U$ where $u \equiv v$. While an edge $e \in E \; where \; E \subseteq V \times V$ represents whether two records $e_i = (r_i, r_j)$ are matched, and an edge weight $w_e \in W(E) : E \to \mathbb{R}$ represents the normalized similarity between the two nodes. A graph G could be represented as an adjacency matrix A of size $|V| \times |V|$, where each cell in the matrix contains either a 1 if an edge exists between two nodes or 0 if an edge does not exist. Each cell value containing 1 could be multiplied by the edge weight to represent a weighted adjacency matrix. We also define a set of clusters $Q \subseteq P(V)$ where Q elements are a subset of V and P is a partition of V. The clustered graph conventionally can be seen as a graph of subgraphs where each meta-vertex represents each subgraph of vertices or records as follows:

$$V\prime = Q \tag{1}$$

$$E\prime = (Q_i, Q_j) : \exists\, (v_i, v_j) : v_i \in Q_i, v_j \in Q_j, (v_i, v_j) \in E \tag{2}$$

Second, the corpus of input data could be modeled as a bipartite graph $G = (V, Y, E)$ with two types of nodes V and Y where an edge $e \in E \; E \subseteq V \times Y$ can only exist between two nodes of different types. In our case, the first type of node $v \in V$ represents a unique document $u_i \in U$ where $u \equiv v$ and the second type of nodes $y \in Y$ represents a unique token $t_i \in T$ where $t \equiv y$. Edges can exist between a document node and a token node if the token exists in the document reference body. We also define the notion of node membership q_i where a node n_i can only be a member of one cluster q_i. Finding the best suitable cluster membership for a node n_i could be achieved through optimizing cluster quality heuristics such as Modularity [31] or Conductance [32]. Modularity quantifies the cluster quality in a graph by comparing the edge density in each cluster to a randomly rewired hypothetical network. Recall the notions defined in equation 1 and 2. Modularity can then be conceived as:

$$M = \frac{1}{2m} \sum_{i,\, j} \left[A_{i,\, j} - \frac{k_i k_j}{2m} \right] \partial\,(Q_i, Q_j) \tag{3}$$

Where m is the number of edges in the graph and k is the degree of a node. In addition A is a weighted adjacency matrix constructed from $E\prime$. And, i and j are indices for each unique record in the file, represented as a vertex in the graph as $v\prime \in V\prime$. And $e\prime \in E\prime \; E\prime \subseteq V\prime \times V\prime$ and $e\prime \equiv A_{i,\, j}$.

C. Similarity and Transitivity

Consider that if record a matches record b and record b matches record c, then by transitivity, record a matches record c. The former definition of transitivity is the notion that binds our assumptions that lead us to create a graph from a set of matched documents. This assumption can only hold if the probability of record a matching record b and the probability of record b matching record c care is high enough [33]. A high enough probability in approximate string matching is considered above 50% [34]. we interpret normalized approximate string similarity measures such as the Jaccard index and Levenstein ratio as matching probabilities. Hence, a Jaccard similarity between two documents below 50% is not accepted as a link between two records, which is crucial for the transitivity assumption to remain valid. Note also that the transitivity assumption is what allows us to form a document-document graph; otherwise, it does not make logical sense to apply a transitive closure algorithm on a formed graph if transitivity does not hold or, in other terms, if the edge weight between nodes representing the matching probability is less than 50%. That assumption could also be corroborated by interpreting similarity at 50% as extreme uncertainty of whether the two documents are similar instead of the intuition that a 10% similarity indicates

uncertainty. On the contrary, a 10% similarity holds more certainty that the two documents are dissimilar. Hence a 50% similarity is an appropriate baseline for interpreting approximate string similarity measures as matching probabilities.

IV. Method

In this section we extensively describe the proposed framework as shown in Fig. 1.

A. Merged Input Corpus

The input corpus is defined in one file. The file is merged from multiple data sources. The user manages this step where the only requirement is a single merged file. In the future, we intend to address having to merge multiple data sources as part of the framework developed here. The assumption is that each line in the file represents a document containing a reference body where the number of latent entity profiles is less than or equal to the number of documents. In addition, as mentioned in the preliminaries section, it is assumed that documents share at least one token so that the assumption of transitivity is not broken III-C. Note that the unique identifier $u_i \in U$ is either provided in the input file or is automatically generated by ModER.

B. Preprocessing

The parsing process includes normalization, cleaning, tokenization, and filtering and is central to the framework. Tokenization is preceded by the normalization and cleaning step, where the text from the reference bodies of the documents is cleaned from particular characters and converted to lowercase. If a unique character appears in the token's middle, it is removed, and the entire string is compressed. We use the standard approach to tokenizing text in English, splitting the text based on spaces between tokens. The parsed data are then loaded into memory, and unique token frequencies and length dictionaries are computed. Stop words are also removed if their frequencies exceed a parameter sigma σ. More formally, as referred to before in the preliminaries Section III-B, the corpus D consists of tuples $d_i = (u_i, \ r_i)$ where $u_i \in U$ and $r_i = w_i \in W$ contains a unique distinct set of tokens T with different counts C given that t_i is a distinct unique token where $t_i \in T \subseteq w_i$ and c_i is the corresponding count of each unique token t_i where $c_i \in C \ and \ C = |t_i \in T|$. In addition, a token length dictionary is computed where $l_i \in L \ and \ L = len(t_i \in T)$. First each document $d_i \in D$ is processed using σ to filter out tokens $t_i \in r_i$ with frequency $c_i \in C \ and \ C = |t_i \in T|$ above σ to filter out stop words.

C. Blocking

The blocking process aims at reducing the potential number of pairwise matching across a data file as much as possible. We use a frequency-based blocking algorithm that assumes that two records that refer to the same entity share at least one token. We adapted the frequency-based blocking technique presented in the DWM [1]. The blocking method relies on the parameter beta β. For each reference token $t_i \in r_i$ with frequency $c_i \in C$ and $C = |t_i \in T|$ below β and above 2 is considered a blocking token $t_{Bi} \in T$. Blocking tokens are identified for each reference in a list L where the filtered records are repeated. The list is then grouped by blocking tokens regardless of reference. Each block includes all the references where the same blocking token appeared, and the number of blocks was equivalent to the number of unique blocking tokens in the dataset. This process is formalized in Algorithm 1.

Algorithm 1: Blocking

Input : $C(T)$, R, β, σ: unique token frequencies, record set
Result : B: list of blocks
$B \leftarrow list$
for $r \in R$ **do**
  if $c_i \in C \ for \ each \ t_{ij} \in r \ where \ t_i \in T \geq \sigma$ **then**
    $r \leftarrow remove \ t_{ij} \ from \ r$
  end
  if $c_i \in C \ for \ each \ t_{ij} \in r \ where \ t_i \in T \leq \beta$ **then**
    $L \leftarrow (t_{ij}, r)$
  end
end
$L_s \leftarrow sort(L, t_i \in T)$
$T_u \leftarrow unique(t_{ij} \in L_s)$
for $t_i \in T_u$ **do**
  for $l_s \in L_s$ **do**
    if $t_i = t_i \in l_s$ **then**
      $b_i \leftarrow r \in l_s$
      $B \leftarrow B + b_i$
    end
  end
end
return B

D. Fast Matching using Locality-Sensitive Hashing

The goal here is to provide a quick, fast, and multilevel way of grouping documents that might belong to the same latent unique entity. However, matching every document in the corpus would result in an algorithm that runs as $O(N^2)$ in time. So, our efforts are concentrated on reducing the number of possible matching operations through 3 steps. First, after the preprocessing phase, we apply a Locality Sensitive Hashing (LSH) [16] algorithm to allow for a quick approximation of a similarity function such as Levenshtein ratio, Cosine distance, or Jaccard index. Even at the block level, computing the former similarity metrics can be expensive for larger files. LSH aims at using a hashing algorithm to approximate a similarity function such as Jaccard index [11]. Jaccard similarity involves computing the set intersection and union across tokenized words between the two documents being compared. Computing both a union and an intersection could be computationally expensive, misaligning with our overarching goal of reducing string-wise comparisons as much as possible. An LSH algorithm operates on a metric, a threshold, an approximation factor, and a set of probabilities. The goal is to design a hash function that approximates a metric by optimizing a threshold. The approximation is achieved by making sure that the set of probabilities holds.

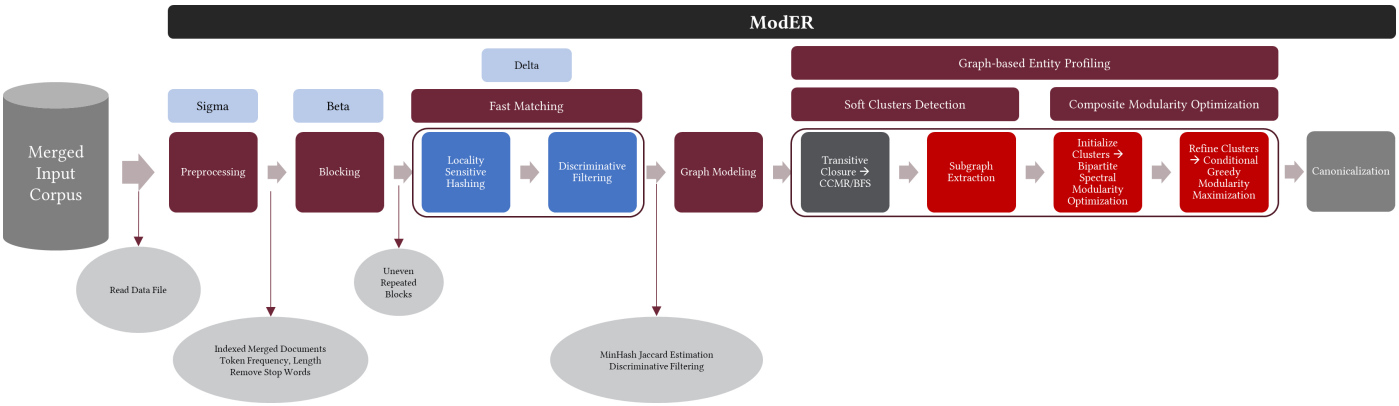


Fig. 1. An Overview of the ModER Framework.

Here we adapt the MinHash algorithm [17] to estimate the similarity between records in the same block. The MinHash algorithm is designed to approximate the Jaccard index between two sets. Sets are columns in a matrix where each row represents an element from the union of the two sets. A cell value of 1 represents the existence of that element in one of the two sets. The critical observation is that the Jaccard similarity could be approximated by counting the number of collisions between the two sets after hashing each element in each set. So counting the number of collisions of hashes between the two sets and normalizing them by the total number of hashes approximates the Jaccard similarity [35] [16]. That technique reduces the number of operations needed to compare two documents from $O(N^2)$ to $O(N)$ in time in best cases and $O(N\ log(N))$ in worst cases. we adapt MinHash by precomputing the approximating hash functions over the entire corpus. We provide a simple implementation of the MinHash algorithm in our code; however, while running experiments, we used the highly optimized implementation introduced by [36]. Each word is hashed for each document in the corpus using a 32-bit SHA algorithm [37]. The hashed unique tokens represented in each document are permutated and then drawn randomly from each document. A signature is then computed for each unique word in the documents, and the minimum signature is chosen to represent that word in the documents. The Jaccard index is then estimated linearly by counting the number of similar signatures in the same position over the total number of signatures appearing in the two documents according to the proof presented initially in [17]. We did not formalize MinHash here as formalizations of the algorithm are widely available.

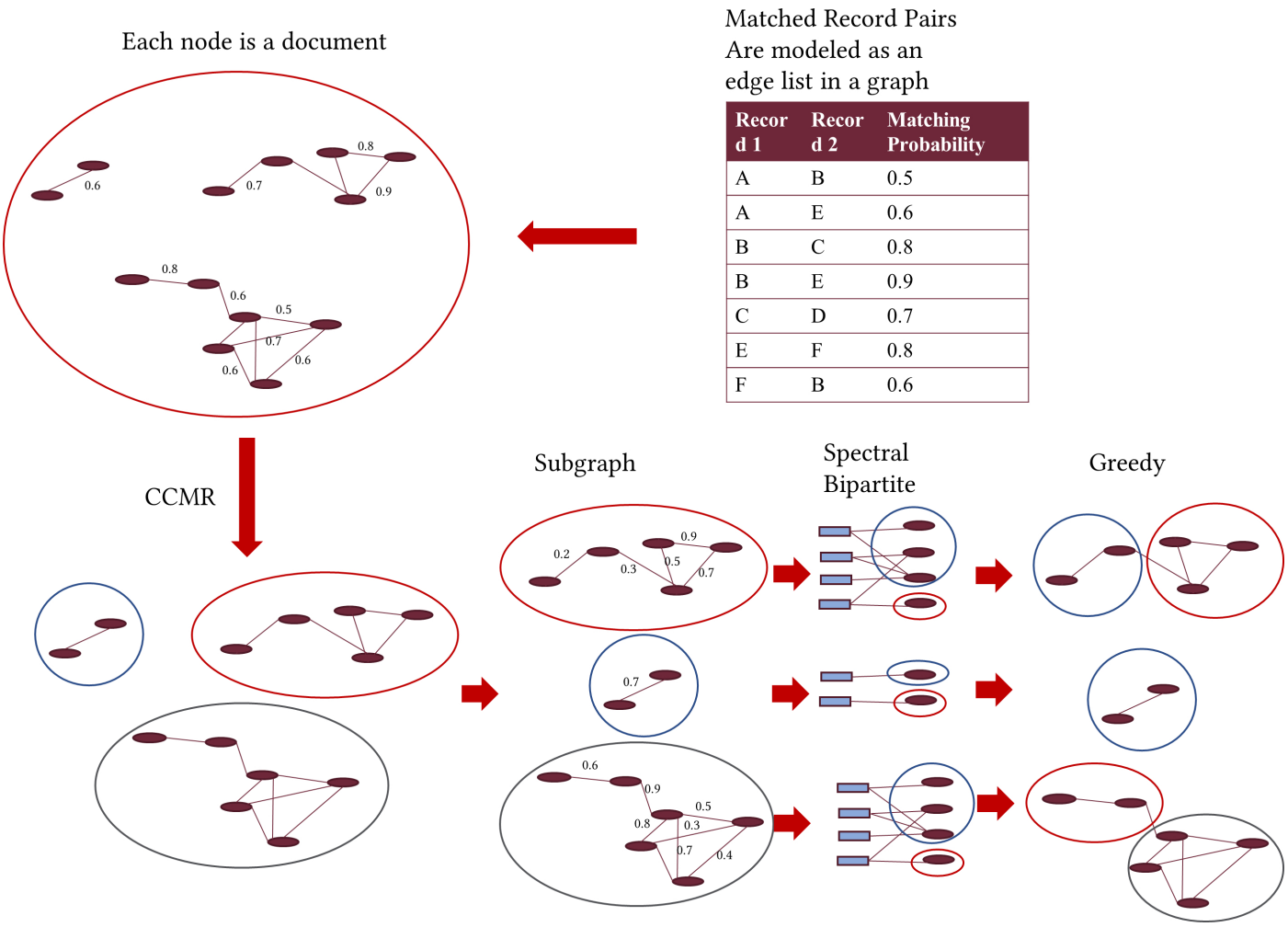
The second part of our matching scheme, also done on the block level, is that we do not take the estimated Jaccard similarity at face value. We first filter the documents and extract what we call discriminate tokens. Those are tokens longer in length than a parameter delta δ. Discriminate tokens represent tokens that, by looking at them, you can quickly determine whether two documents are similar. Those tokens might be social security numbers, credit card numbers, long street names, long last names, scientific names, product numbers, or models. Those tokens are usually highly discriminative in determining whether two documents are similar. Hence before estimating the Jaccard index on the block level, we check whether the two documents have tokens in common that are longer than delta and their Levenshtein distance is less than 2. We consider 2 to be the threshold that defines a typo. The strength of our framework lies in the fact that we do not use a similarity threshold to match documents on the block level. Instead, we allow the data to automatically match the documents based on the characteristics represented in the parameters derived from the token frequencies and length. Hence, on the block level, we link documents with a similarity above 0 without any threshold setting. That process is described semi-formally in the pseudo-code presented in Algorithm 2.

Algorithm 2: Pairwise Matching using MinHash

```
Input : B, F: list of computed blocks, unique token
        lengths
Result : E: linked weighted pairs
E ← list
for b ∈ B do
    for r1 ∈ b do
        for r2 ∈ b do
            if r1 < r2 then
                d1 ← f1 ∈ F
                d2 ← f2 ∈ F
                if d1 = d2 or levenshtien(d1, d2) ≤ 2
                 then
                    s ← 1.0
                end
                else
                    s ← minHash(r1, r2)
                end
                if s > 0.0 then
                    E.append((r1, r2, s))
                end
            end
        end
    end
end
return B
```

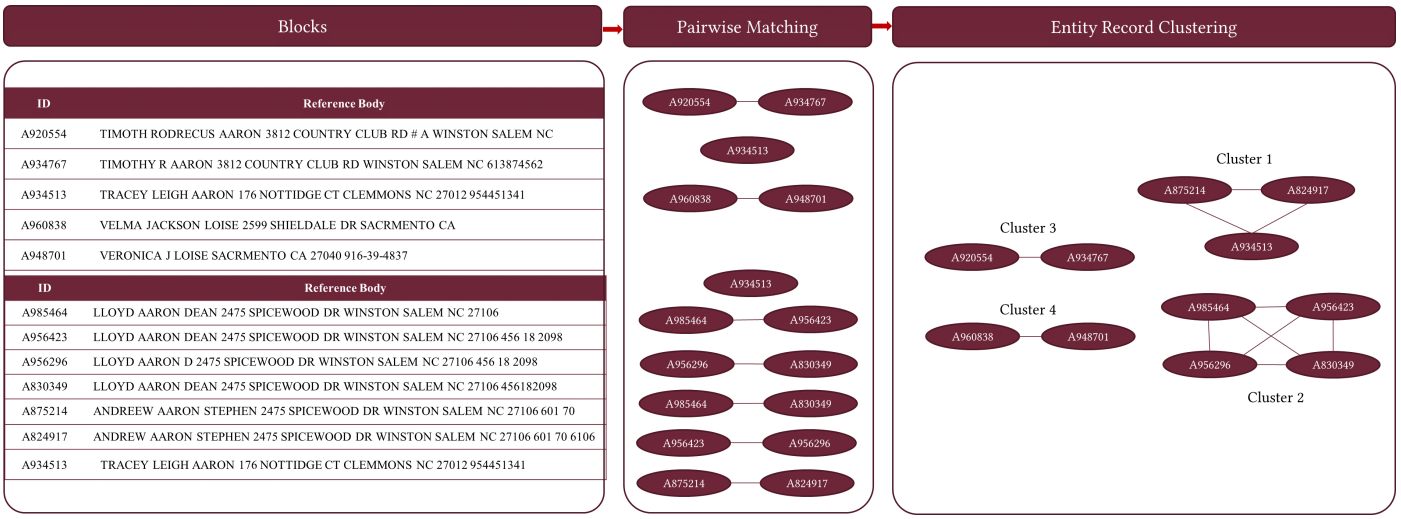
E. Graph Modeling

The unordered list of matched records can be considered an edge list or an adjacency matrix representing a graph of



Record 1	Record 2	Matching Probability
A	B	0.5
A	E	0.6
B	C	0.8
B	E	0.9
C	D	0.7
E	F	0.8
F	B	0.6

Fig. 2. Graph Modeling. First, the Matched Records are Modeled as One Large Weighted Graph with Expected Disconnected Components Due to the Transitivity Property. Second, Connected Component Detection Algorithms Output the Nodes in each Connected Component. Third, each set of Nodes is then Modeled as a Subgraph.



ID	Reference Body
A920554	TIMOTH RODRECUS AARON 3812 COUNTRY CLUB RD # A WINSTON SALEM NC
A934767	TIMOTHY R AARON 3812 COUNTRY CLUB RD WINSTON SALEM NC 613874562
A934513	TRACEY LEIGH AARON 176 NOTTIDGE CT CLEMMONS NC 27012 954451341
A960838	VELMA JACKSON LOISE 2599 SHIELDALE DR SACRMENTO CA
A948701	VERONICA J LOISE SACRMENTO CA 27040 916-39-4837

ID	Reference Body
A985464	LLOYD AARON DEAN 2475 SPICEWOOD DR WINSTON SALEM NC 27106
A956423	LLOYD AARON DEAN 2475 SPICEWOOD DR WINSTON SALEM NC 27106 456 18 2098
A956296	LLOYD AARON D 2475 SPICEWOOD DR WINSTON SALEM NC 27106 456 18 2098
A830349	LLOYD AARON DEAN 2475 SPICEWOOD DR WINSTON SALEM NC 27106 456182098
A875214	ANDREEW AARON STEPHEN 2475 SPICEWOOD DR WINSTON SALEM NC 27106 601 70
A824917	ANDREW AARON STEPHEN 2475 SPICEWOOD DR WINSTON SALEM NC 27106 601 70 6106
A934513	TRACEY LEIGH AARON 176 NOTTIDGE CT CLEMMONS NC 27012 954451341

Fig. 3. The Typical Unsupervised Entity Resolution Approach Starts with Frequency-Based Blocking, Pairwise Matching, and Entity Clustering.

records. The blocking algorithm considers two similar records having a high probability of representing the same entity if

they have a minimum of one token in common. That increases the number of records representing one entity, as shown in Fig. 3. As a result, a record may appear in more than one entity, making clustering more expensive and the underlying graph of matched records more complex. In addition, the matching process also implies that one record might be similar in an indirect way to another record by transitivity, provided that the matching probability is above 50%. The output of algorithm two is modeled as a graph, as mentioned in the preliminaries in Section III-B and as seen in Fig. 2. A subgraph is then extracted on the block level.

F. Graph-based Entity Profiling

1) Transitive Closure: In transitive closure, the aim is to find the set C of sets $V\prime$ where each set represents a strongly connected component in the graph. A strongly connected component is a set of nodes where each node has at least one link to another node. Strongly connected components are separated by unlinked nodes. We also refer to strongly connected components as soft clusters. Here in ModER, one of two approaches could be used. First, an algorithm named CCMR was introduced by [26] and used and reimplemented in the original DWM [1] as a single-step clustering method. The algorithm starts by creating star subgraphs from the matched pair list. That is done by simply grouping the pair list by the smallest node. Then, the algorithm iteratively checks whether its vertices are assigned to subgraph components where the center of the subgraph component is the smallest vertex in its first-order neighborhood, relying on the MapReduce framework for scalability. The DWM provides an efficient implementation of the algorithm without relying on MapReduce. The algorithm is formalized and described thoroughly in [26] and [1]. The second approach that could be used is a simple breadth-first search algorithm [38] to extract the connected components or soft clusters. We provide implementation and describe both algorithms semi-formally in Algorithms 3 and 4.

2) Subgraph Extraction: The subgraph extraction process aims to avoid recomputing edge weights that have been computed before during the fast-matching process. A subgraph is simply the set of soft clusters that have been computed using the transitive closure process. Each soft cluster is represented by a set of unique nodes or documents. Each document or node can appear in only one soft cluster. To extract a subgraph from the larger graph, we find the edges where all the nodes in the current soft cluster are represented exclusively. Subgraph extraction is formalized in Algorithm 5.

3) Composite Modularity Optimization: In the Composite Modularity Optimization step, the goal is to initialize the cluster membership of each node in the extracted subgraph based on token memberships in each document. Next is to refine the clusters discovered in the initial clustering to more precise memberships. In the following two subsections, we outline how we implement that process.

a) Bipartite Spectral Modularity Optimization: This step exploits the relationship between documents and unique tokens across all documents. First, we project the unique document identifiers and unique tokens across all documents as a bipartite graph similar to Fig. 4.

Algorithm 3: Transitive Closure using Adapted CCMR

Input : S, μ: pairs of matched records and their computed similarity scores, similarity threshold
Result : P: list of soft clusters as pairs indexed by the least record in the cluster as the first element of the pair

```
P ← list of soft clusters
R_P ← initialize list of tuples
for s ∈ S do
    if s_i ≥ μ then
        R_P ← append s_i
    end
end
R_P ← sort by first element in pair
while no convergence do
    for (r_i, r_j) ∈ R_P do
        P ← append pair belonging to connected component
            when assuring that all record
            nodes belong to the connected component
            with the smallest record id
            node at the center in their neighborhood;
    end
end
return P
```

Algorithm 4: Connected Components Breadth First Search

Input : $G = (V, E)$: graph
Result : V': set of nodes as component

```
seen ← set
components ← list
for v ∈ V do
    if v! ∈ seen then
        visited ← set
        queue ← list
        visited.add(v)
        queue.add(v)
        while queue do
            dequeued ← queue.pop()
            neighbors ← G.getNeighbors(dequeued)

            for n ∈ neighbors do
                if n! ∈ visited then
                    visited.add(n)
                    queue.append(neighbors)
                end
            end
        end
        seen.add(visited)
        components.append(visited)
    end
end
return components
```

A920554	1	TIMOTHY	RODRECUS	AARON	3812 COUNTRY CLUB RD # A	WINSTON SALEM	NC	27104	613-87-4562
A934767	2	TIMOTHY	RODRECUS	AARON	3812 COUNTRY CLUB RD # A	WINSTON SALEM	NC	27104	613-87-4562
A934513	3	TRACEY	LEIGH	AARON	176 NOTTIDGE CT	CLEMMONS	NC	27012	954451341
A960838	4	VELMA	JACKSON	AARON	2599 SHIELDALE DR	WINSTON SALEM	NC	27107	559-64-4215
A948701	5	VERONICA	OLIVIA	AARON	3401 LOCHURST CT	PFAFFTOWN	NC	27040	335-39-4837

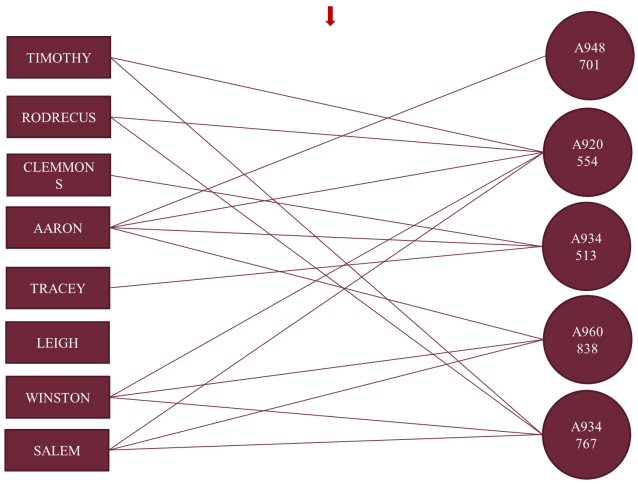


Fig. 4. A Token-Record Bipartite Graph.

Algorithm 5: Subgraph Extraction

Input : V', A: component, adjacency
Result : $G' = (V', E')$: graph
$E' \leftarrow list$
for $v \in V'$ **do**
  $neighbors \leftarrow G.getNeighbors(v)$
  for $nn \in neighbors$ **do**
    if $nn \in V'$ **then**
      $E'.append((n, nn, A[n][nn]))$
    end
  end
end
return (V', E')

Recall equation 3; we defined the Modularity heuristic M as the sum of the difference between the edges of the current graph versus a null model graph where edges are rewired randomly within clusters. Here we approach Modularity optimization on the bipartite network of documents and unique tokens or words using a spectral approach or, in other words using matrix form. The goal is to optimize Modularity across both types of nodes in the bipartite graph and extract only the optimized cluster memberships of document nodes. First, a clustered index matrix S is defined where the rows are the nodes in the graph, and the columns are the number of clusters in the graph. Initializing this matrix at the beginning means that each node has its cluster, so the number of rows equals the number of columns. The values in S are either 0 or 1, indicating node memberships. Recall from equation 3 defining Modularity, the term $\frac{k_i k_j}{2n}$ defines the probability of an edge in the null model and can be denoted by $P_{i,j}$. Hence the term $\left[A_{i,\ j} - \frac{k_i k_j}{2n}\right]$ can be rewritten as $[A_{i,\ j} - P_{i,j}]$ and can be referred to as $B_{i,j} = [A_{i,\ j} - P_{i,j}]$. According to [39], the previous matrix form can be combined with equation 3 to redefine Modularity in matrix form as:

$$M = \frac{1}{2n} Tr(S^T B S) \tag{4}$$

Where n is the number of edges in the graph since a bipartite graph is naturally partitioned into a minimum of two initial clusters [39]. That naturally help us derive a definition of Spectral Bipartite Modularity that penalizes any random choices of edges if the nodes belong to the same cluster, thus the bipartite Modularity in a non-matrix form could be defined as:

$$M = \frac{1}{n} \sum_{i=1}^{p} \sum_{j=1}^{q} \left[\widetilde{A_{i,j}} - \frac{k_i d_j}{n}\right] \partial(Q_i, Q_{j+p}) \tag{5}$$

Where $\widetilde{A}$ is the bi-adjacency matrix. k_i is the degree of node i from the document nodes and d_j is the degree of node j from the word nodes. n is the total number of edges in the graph and Q_i is the partition of a document node i while Q_{j+p} is the partition of a word node j indexed as $j = i + p$. Thus equation 5 in matrix form becomes:

$$M = \frac{1}{2n} Tr(R^T \widetilde{B} T) \tag{6}$$

Where R is the cluster index matrix similar to S but for document nodes, while T is the cluster index matrix similar to S but for token or word nodes. $\widetilde{B}$ is the difference between

the biadjacency matrix $\widetilde{A}$ and the reformulated bipartite null model $\widetilde{P}$ where $\widetilde{B} = \left[\widetilde{A} - \widetilde{P}\right]$. The goal is to maximize Modularity in matrix form as in equation 6. The term $\widetilde{B}T$ in equation 6 could be rewritten as $\left[\widetilde{AT} - \widetilde{P}T\right]$ indicating that the bipartite null model is calculated on the token nodes only. We refer to this case as $\widetilde{T}$. Or it could be written as $\left[\widetilde{AR^T} - \widetilde{P}R^T\right]$ indicating that the bipartite null model is calculated on the document nodes only. We refer to this case as $\widetilde{R}$. In terms of summation over the matrix in an optimization scheme, these two cases can be rewritten as:

$$M = \frac{1}{n}\sum_{i=1}^{p}\left(\sum_{k=1}^{c}\left[R_{i,k}\widetilde{T_{i,k}}\right]\right) \tag{7}$$

$$M = \frac{1}{n}\sum_{j=1}^{q}\left(\sum_{k=1}^{c}\left[\widetilde{R_{i,k}}T_{i,k}\right]\right) \tag{8}$$

Where c is the maximum number of clusters in the graph, note that this is equivalent to hypothetically assigning document nodes to clusters of word tokens and vice versa, similar to traditional modularity optimization approaches such as Louvain [40] hence the delta modularity, in this case, was the summation of both delta modularities from both node types. Given the previous formulation, we maximize Modularity using a greedy approach described in Algorithm 6.

b) Conditional Greedy Modularity Maximization: This second step aims first to recast the documents as a document-document graph as as in Fig. 5. Second to break down the soft clusters in a hierarchical manner by maximizing Modularity M using a conditional modularity maximization algorithm as seen in Fig. 2. That step is seen as further filtering. The assumption here is that clusters in a graph are often defined by the density of edges between a set of nodes. Hence, the number of connections or edges between clusters characterized by high Modularity is often meager. Modularity is defined in equation 3 as M. It measures the difference between the actual number of edges and an expected number of edges between nodes. An expected number of edges between nodes can be considered a random rewiring of the graph given the same nodes. Optimizing Modularity through maximization in a graph is a complex problem and is often tackled through various ways to reduce the number of comparisons between all nodes in a graph. During maximization, we use the shorthand equation provided in [40] to reduce the computational cost of computing delta Modularity of all nodes as shown in equation 9.

$$\Delta M = \left[\frac{\sum_{in}+k_{i,in}}{2n} - \left(\frac{\sum_{tot}+k_i}{2n}\right)^2\right] - \tag{9}$$

$$\left[\frac{\sum_{in}}{2n} - \left(\frac{\sum_{tot}}{2n}\right)^2 - \left(\frac{k_i}{2n}\right)^2\right] \tag{10}$$

Where M is Modularity, in are incident nodes to cluster, tot all nodes inside cluster, k is the degree of the node, n is the total number of edges. We also apply a condition that a node is only assigned to the maximum Modularity difference cluster if the delta modularity is positive and the matching probability between both document references is 100%. That

Algorithm 6: Spectral Bipartite Modularity Optimization

Input : V', D', W': set of nodes in current soft cluster, document references, unique words
Result : updated node labels
$edges \leftarrow list$
$p = length(V')$
$q = length(W'[V'])$
$c = p + q$
$rg = index(W'[V'])$
$gn = index(V')$
$A = zeroMatrix(p, q)$
for $v \in V'$ **do**
  for $w \in D'[v]$ **do**
    $edges.append((v, w))$
    $A[gn[v], rg[w]] = 1.0$
  end
end
for $i = 0$ *and* V' *and* $i++$ **do**
  $assignMembership(V'[i], i)$
end
for $i = i$ *and* $W'[V']$ *and* $i++$ **do**
  $assignMembership(W'[V'][i], i)$
end
$ki = sum(A, 1), dj = sum(A, 0), kd = kidj$
$m = sum(ki), B = A - (kd/m)$
$T0 = initializeModularityMatrix(gn, V')$
$R0 = initializeModularityMatrix(rg, W'[V'])$
$minimumDeltaModularity = min(1/m, 0)$
$deltaModularity = 1, modPrevious = 0$
while
 $deltaModularity > minimumDeltaModularity$
 do
  $Tp = T0^T.B$
  $maximumModularityIndex = argMax(Tp^T)$
  $R = zeroMatrix(q, c)$
  for $i, length(maximumModularityIndex)$ **do**
    $R[i, maximumModularityIndex[i]] = 1$
  end
  $Rp = B.R$
  $maximumModularityIndex = argMax(Rp)$
  $T = zeroMatrix(p, c)$
  for $i, length(maximumModularityIndex)$ **do**
    $R[i, maximumModularityIndex[i]] = 1$
  end
  $T0 = T$
  $modCurrent = modPrevious$
  $RtBT = T^T.B.R$
  $sumMod = (1/m) * RtBT$
  $modCurrent = sum(modCurrent)$
  $deltaModularity =$
   $modCurrent - modPrevious$
end
$updateNodeMemberships(extractMemberships(T))$

Cluster ID	Reference ID	Reference Body
A824917	A985464	LLOYD AARON DEAN 2475 SPICEWOOD DR WINSTON SALEM NC 27106
A824917	A956423	LLOYD AARON DEAN 2475 SPICEWOOD DR WINSTON SALEM NC 27106 456 18 2098
A824917	A956296	LLOYD AARON D 2475 SPICEWOOD DR WINSTON SALEM NC 27106 456 18 2098
A824917	A830349	LLOYD AARON DEAN 2475 SPICEWOOD DR WINSTON SALEM NC 27106 456182098
A824917	A875214	ANDREEW AARON STEPHEN 2475 SPICEWOOD DR WINSTON SALEM NC 27106 601 70 6106
A824917	A824917	ANDREW AARON STEPHEN 2475 SPICEWOOD DR WINSTON SALEM NC 27106 601 70 6106

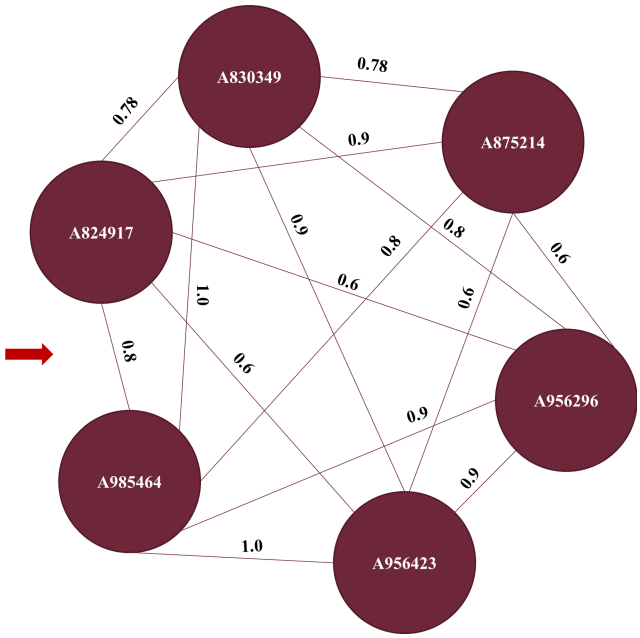


Fig. 5. Modeling a Data Set as a Record-Record Weight Graph based on the Similarity between the Records.

ensures that precision is not compromised by breaking precise clusters discovered in the first step. In addition, that is also to assure that discriminative terms are factored into the composite Modularity optimization method. Our Conditional Modularity Maximization algorithm is formalized in Algorithm 7.

Algorithm 7: Conditional Greedy Modularity Optimization

Input : $G' = (V', E')$: subgraph nodes
Result : updated node labels
$M_{initial} \leftarrow computeModularity(V', E')$
for $i \in V'$ **do**
  $iCom = G'.getMembership(i)$
  $neighbors = G'.getNeighbors(i)$
  for $j \in neighbors$ **do**
    $jCom = G'.getMembership(j)$
    $weight \in G'.getEdgeWeight(i, j)$
    $\Delta M \leftarrow$ compute difference in Modularity M according to Louvain's equation
    $L_M \leftarrow (j, \Delta M)$
  end
  $c_i \leftarrow \arg\max(L_M)$
  find the largest cluster with the largest delta Modularity
  Δ M
  and the corresponding weight
  $v2 \leftarrow v1$ assign the current node to the maximum cluster if edge weight is 1
  $updateMemberships(V')$
end
$M_{final} \leftarrow computeModularity(V', E')$

G. Canonicalization

The clusters representing unique entity profiles are defined and persisted to the disk in this final step through a link index file. The link index file describes the final entity clusters of the unsupervised entity resolution framework as a list of ordered pairs where the first element of each pair represents the entity profile identifier to which the record is assigned. The entity profile identifier is simply the least recorded identifier in the cluster. The second element of each pair represents a member's unique record identifier.

V. Experiments

A. Datasets

1) Synthetic Datasets: ModER is tested on a synthetic benchmark dataset. The reason lies in the fact that there is a lack of benchmark datasets for the task of entity resolution. Most entity resolution systems are evaluated against benchmark datasets [41] that are designed for an entity matching task and not an entity resolution task. Hence we use the synthetic dataset described in [42] as seen in Fig. 5 that is specifically designed for the task of entity resolution. This simulator-based data generator uses probabilistic approaches to generate coherent individual data for persons that do not exist except for $S3$ and $S6$, which both represents generated addresses and names of restaurants that do not exist and were introduced in [43]. The data fields are names, addresses, social security numbers, credit card numbers, and phone numbers mixed in several layout configurations. Some samples are labeled as mixed layout, meaning that each row might come with a different order of those attributes and might not be delimited. The standard label means that all the rows in the data file have the same order and attributes. The generator described in [44] used a probabilistic error model to inject various errors in the previously developed simulated dataset. For example, in this excerpt of a generated data file shown in Fig. 5, the first four records are almost

identical except for that record $A956296$ has a missing last name and the format of the phone numbers or whether they exist at all, all are errors injected and generated on purpose. In addition, the last two records are almost identical except for the first name, where an intentional error was introduced. A ground truth set recording the actual clusters of the simulated records is then sampled from the generated synthetic database. Next, the corresponding references are pulled from the generated synthetic database to create various sample files with different sizes, levels of quality, and layouts. Sizes of files can vary from 50 to $20K$ rows. For a more detailed description of the dataset please refer to [15].

2) Benchmark Dataset: **Abt-Buy** is an e-commerce benchmark dataset introduced and curated by [41]. Despite the lack of benchmark datasets that are desgined specifically for the task of entity resolution rather than entity matching we used this dataset as a means to compare results with other systems. The dataset represents an entity matching task between products listed on 2 e-commerce websites[2]. The dataset appears as one data file containing attributes, product names, descriptions of products, manufacturer of the products, and price. The data set also is provided with a ground truth file containing a perfect matching between entities across both websites. The number of rows in this dataset is 1081 from the first source file and 1092 from the second data file.

3) Benchmark Models: **Data Washing Machine (DWM):** the Data Washing Machine (DWM) [1] implements a fully probabilistic, iterative, and token frequency-based approach. The DWM produces remarkable results on simulated datasets of various sizes, layouts, and qualities. The DWM is computational, iterative, and probabilistic; it follows a traditional data cleaning and merging pipeline, tokenization, blocking, matching, clustering, profiling, and canonicalization. The DWM relies on the idea of starting with a set of configuration parameters and then iteratively incrementing the parameters according to a self-administered evaluation of the quality of the clusters using an entropy-based metric. The DWM has been shown to provide robust results in large datasets. In addition, the current implementation is very modularized, allowing for room for improvement.

Magellan: Magellan, introduced in [45], is an end-to-end solution to entity resolution developed as a user-centric approach. It has been used and adopted widely. The model relies on providing guides that tell users what to do in entity matching scenarios. The framework also provides tools to cover the entire entity matching pipeline using a simple, approachable implementation.

B. Evaluation Metrics

For evaluation we measured the precision and recall against the generated ground truth entity clusters. The ground truth is a list of each record and its membership cluster identifier. After canonicalization, the saved link index is grouped by the least record identifier in each cluster. Thus, all records belonging to the same cluster have the same record identifier as the first element in the link index pair. We then loop on each pair in the canonicalized link index and examine whether they belong together in the ground truth. Finally, we measure the following statistics against the ground truth for each sample run as shown in Table I. That is also shown in the increased balanced accuracy [46], which is a valuable measure for problems such as entity resolution. Entity resolution is characterized by having a class imbalance as the number of matched pairs is usually way less than the number of unmatched pairs causing a very high number of true negatives.

[2]www.buy.com and www.rakuten.com

C. Results

We ran our model on an Intel Core $i7 - 4720HQ$ CPU @ $2.60GHz$ and $32GB$ of RAM. We ran ModER and DWM first on the samples $S1$ through $S18$, as described in Tables III and II. To determine optimal parameters for ModER relative to maximum F1-Scores, we ran each sample 10 times on incrementing beta, sigma, and delta and chose the parameters that gave the best F1-Scores. Please note that we do not set them directly as numbers when setting parameters beta β, sigma σ, and delta δ. Instead, we set them using a percentile formula where the percentile of stop words is σ, the percentile of blocking words is β, and the percentile of word length is δ. For running the DWM, we set the parameters to the equivalent in our model. For example, we assume that our baseline matching threshold μ is 0.5 or 50% quantifying maximum uncertainty. Also, in the DWM runs, the quality epsilon ϵ is tuned based on our previous work's data [15]. we also informed the setting of β and σ similar to what we did in the GDWM and set them to 6 and 7, respectively.

Table II shows the results from running the DWM on the 18 samples with equivalent set parameters. Even though it is challenging to compare both techniques due to their differences, it is helpful to compare both in relatively limited runs. Table V compares the final F1-Scores of both the DWM and ModER. On the other hand interpreting Table IV is tricky because more experimentation needs to be done to get a complete picture of the performance of the developed approach concerning what has already been done. However, ModER improves overall precision at the expense of recall from those results. Balancing precision and recall is challenging for most classifiers.

Composite Modularity Optimization as a technique for entity profiling is very efficient at detecting large clusters and breaking them down due to the reliance on a bipartite technique of modeling document nodes and unique words. That initialization is used in the second stage of Conditional Greedy Modularity maximization to refine the detected clusters further. Note that the conditional aspect of the greedy Modularity Maximization technique is intentionally injected into the algorithm to ensure that no nodes are assigned to new clusters if the delta modularity change is positive yet too little. That is also to ensure that we only see similarity at 100% or 1.0 as the most certain value indicating matching, and anything else below 100% is mere speculation and prediction and should be treated that way in other entity resolution systems.

In addition, when averaging all samples, as in Table V, ModER offers less F1-Scores performance than the GDWM and the DWM. Note that those averages were directly taken from [15]. Those values were for matching probabilities set at 70% and higher. That might not be fair because ModER does not rely on matching threshold settings. ModER could be seen as a quick and initial entity profiler before using other

TABLE I. EVALUATION METRICS AND STATISTICS

Statistic	Symbol	Description
True Positives	TP	The number of record pairs that appeared together in the same cluster correctly
True Negatives	TN	The number of record pairs that did not appear in the same cluster correctly
False Positives	FP	The number of record pairs that appeared together in the same cluster falsely
False Negatives	FN	The number of record pairs that did not appear in the same cluster falsely
Precision	P	TP / (TP + FP)
Recall	R	TP / (TP + FN)
F1-Score	F1	2 × (P × R) / (P + R)
Balanced Accuracy	A	TP × (TN + FP) + TN × (TP + FN) / (TP + FN) × (TN + FP)

TABLE II. RESULTS FROM RUNNING DWM ON EQUIVALENT PARAMETER SETTINGS

Sample	Precision	Recall	F1-Score
S1	76.47	96.3	85.25
S2	62.69	87.5	73.05
S3	41.15	95.54	57.52
S4	70.47	87.27	77.98
S5	71.84	87.94	79.08
S6	81.95	75.63	78.66
S7	73.57	86.65	79.58
S8	67.32	36.64	47.45
S9	64.22	17.48	27.48
S10	72.85	27.58	40.01
S11	70.69	26.19	38.22
S12	73.27	26.37	38.78
S13	76.64	83.32	79.84
S14	70.43	84.73	76.92
S15	68	82.04	74.36
S16	74.02	26.84	39.4
S17	70.59	24.39	36.25
S18	69.79	30.33	42.28

systems or rely on a highly unsupervised system that does not require the user to set a matching threshold. In addition to a system that does not rely on matching threshold settings, unlike most state-of-the-art systems, the results are comparable and tell something about the need for matching thresholds. When a user uses such a system to detect entity profiles in a file, they have no experience with what a matching threshold might mean. Hence in ModER, we only let the user set 3 explainable parameters using percentiles, assuming they have studied their data statistically before running any entity resolution system.

In Table VI, we benchmarked ModER using the Abt-Buy dataset introduced in [41]. In addition, Table VII provides insight into multiple runs of ModER on the Abt-Buy dataset with different parameter configurations. Table VI reports running ModER with parameters that resulted from 10 times increasing parameters with the best F1-Score. As seen, ModER outperforms our previous system, DWM, in addition to Magellan. That is mainly due to the Composite Modularity Optimization algorithm efficiency. In addition, our reliance on discriminative words has favored ModER since most data contain a higher percentage of discriminative keywords. In Table VII, it appears that the number of single nodes varied widely as with the number of edges in the graph formed after matching. That is due to the threshold of the varying parameter. In addition, those initial parameter settings affect the initial Modularity widely regardless of the final Modularity. A Higher F1-Score was tied to lower σ levels, suggesting that filtering stop words are always beneficial before running any entity matching algorithm.

TABLE III. RESULTS FROM THE SYNTHETIC DATA SET EXPERIMENTS ON MODER. BETA IS THE PERCENTILE OF BLOCKING WORD FREQUENCIES ACROSS THE WHOLE FILE. SIGMA IS THE PERCENTILE OF STOP WORD FREQUENCY ACROSS THE WHOLE FILE. FURTHERMORE, DELTA IS THE PERCENTILE OF TOKEN LENGTHS ACROSS THE DATA FILE. BA IS THE BALANCED ACCURACY AND IS COMPUTED AS DESCRIBED IN TABLE I. FINALLY, THE FINAL MODULARITY IS MEASURED ON THE FULL DISCONNECTED GRAPH AFTER ASSIGNING THE NEW MEMBERSHIPS AFTER THE COMPOSITE MODULARITY OPTIMIZATION STEP.

Sample	Beta	Sigma	Delta	BA	Precision	Recall	F1	Modularity
S1	80%=3	99%=31.16	100%=17	98.15	100	96.3	98.11	76.74
S2	80%=3	100%=95	80%=9	93.73	95.45	87.5	91.3	66.76
S3	80%=2	95%=7	69%=9	94.19	83.19	88.39	85.71	96.13
S4	90%=5	95%=8	70%=9	86.82	91.35	73.64	81.54	61.21
S5	90%=5	95%=8	70%=9	87.55	93.78	75.1	83.41	62.11
S6	94%=6	99.99%=1972	40%=6	64.03	71.37	28.06	40.28	33.72
S7	93%=6	99%=36.5	95%=9	85.69	92.17	71.39	80.46	56.66
S8	95%=7	100%=400	50%=5	58.99	67.33	18.04	28.45	20.91
S9	95%=7	100%=322	60%=6	58.93	59.47	17.93	27.56	32.88
S10	80%=6	99%=32.5	60%=6	58.58	71.15	17.21	27.71	53.76
S11	80%=5	100%=1458	50%=6	56.25	76.39	12.51	21.5	50.36
S12	80%=4	100%=1859	30%=5	54.51	79.43	9.0	16.2	47.88
S13	90%=6	100%=1887	45%=5	81.05	68.65	62.13	65.23	42
S14	90%=7	100%=4738	90%=9	77.8	79.82	55.6	65.54	43.98
S15	90%=7	100%=9447	90%=9	76.81	81.82	53.62	64.79	46.34
S16	80%=5	100%=713	30%=5	57.01	77.0	14.05	23.76	51.84
S17	75%=4	100%=17.91	30%=5	54.11	78.92	8.23	14.9	50.52
S18	75%=4	100%=3405	30%=5	53.84	78.26	7.69	14.0	51.53

VI. DISCUSSION

A. Overall Effectiveness

Ideally, we need a better measure to quantify the balance between precision and recall, as seen in Fig. 6. The F1-score or the harmonic mean between precision and recall fails to differentiate between instances where recall or precision was very high and when they were balanced. A better entity resolution system always provides a balance between both. In that light, ModER appears to balance both precision and recall on $S1$, $S2$, $S3$, and $S13$. Despite their relative diversity, the common characteristic between those samples is relatively higher quality. That is, the difference between document references injected errors is not significant. They indicate that the first bipartite spectral Modularity optimization did all the work to detect entity profiles. The problem is that bipartite Modularity optimization is a memory-intensive optimization for more significant clusters even though its time complexity is at $O(2N)$, returning only two passes on the Modularity matrix to compute the difference. That points us to address this limitation in the future of balancing space and time complexities.

B. The Effect on Modularity

Here we refer to the final Modularity as the Modularity computed on the final overall graph that has been projected before the entity profiling step. The final cluster memberships have been computed using our Composite Modularity Optimization approach.

In Fig. 7, modularity is more correlated with precision than recall. The more clusters are broken down during the entity profiling step, the higher the Modularity is. Note that Modularity is weakly correlated with the F1 score, meaning that higher modularity values do not necessarily mean higher F1 scores. Modularity is a comparison of edge densities between

TABLE IV. COMPARISON BETWEEN DWM AND MODER

Samples	DWM F1-Score	ModER F1-Score	Performance
S1	85.25	98.11	Improved
S2	73.05	91.3	Improved
S3	57.52	85.71	Improved
S4	77.98	81.54	Improved
S5	79.08	83.41	Improved
S6	78.66	40.28	Worse
S7	79.58	80.46	Improved
S8	47.45	28.45	Worse
S9	27.48	27.56	Improved
S10	40.01	27.71	Worse
S11	38.22	21.5	Worse
S12	38.78	16.2	Worse
S13	79.84	65.23	Worse
S14	76.92	65.54	Worse
S15	74.36	64.79	Worse
S16	39.4	23.76	Worse
S17	36.25	14.9	Worse
S18	42.28	14.0	Worse

TABLE V. THE AVERAGE SAMPLE RUNS ON DWM, GDWM, AND MODER

Method	Precision	Recall	F1-Score	Balanced Accuracy
DWM	72.43	57.82	62.97	76.89
GDWM	84.78	68.62	71.47	84.31
ModER	80.308	44.24	51.691	72.113

TABLE VI. BENCHMARK RESULTS ON THE ABT-BUY DATASET

Model	F1-Score
DWM	10.83
Magellan	43.6
ModER	58.82

TABLE VII. BENCHMARK RESULTS ON THE ABT-BUY DATASET. BA IS THE BALANCED ACCURACY AND IS COMPUTED AS DESCRIBED IN TABLE I. THE INITIAL AND FINAL MODULARITY IS MEASURED ON THE FULL DISCONNECTED GRAPH AFTER ASSIGNING THE NEW MEMBERSHIPS AFTER THE COMPOSITE MODULARITY OPTIMIZATION STEP.

Beta	Sigma	Delta	# Nodes	# Single Nodes	# Edges	Initial Modularity	Final Modularity	Precision	Recall	F1-Score	BA
5	819	9	2173	274	3477	0.0827	0.4577	0.0182	0.3292	0.0345	0.6604
5	92	34	2173	854	2173	0.505	0.647	0.138	0.0367	0.058	0.5183
5	26	8	2173	273	3407	0.083	0.31	0.1669	0.1905	0.1779	0.595
2	92	8	2173	428	1611	0.2697	0.76134	0.6686	0.525	0.5882	0.7625

the current memberships and hypothetical random memberships, also known as the null model.

C. The Interplay between Precision and Recall

Finally, we plot precision as a function of recall, also known as the precision-recall curve in Fig. 8. The difference is that precision-recall curves are usually interpreted in the case of binary classification. In our entity resolution system, we are not assessing a binary classifier. We are, however, assessing cluster quality. Nevertheless, plotting precision as a function of recall on the 18 samples arranged in an ascending order shows that recall tends to be more stable than precision affirming our conclusion that ModER provides higher precision but not

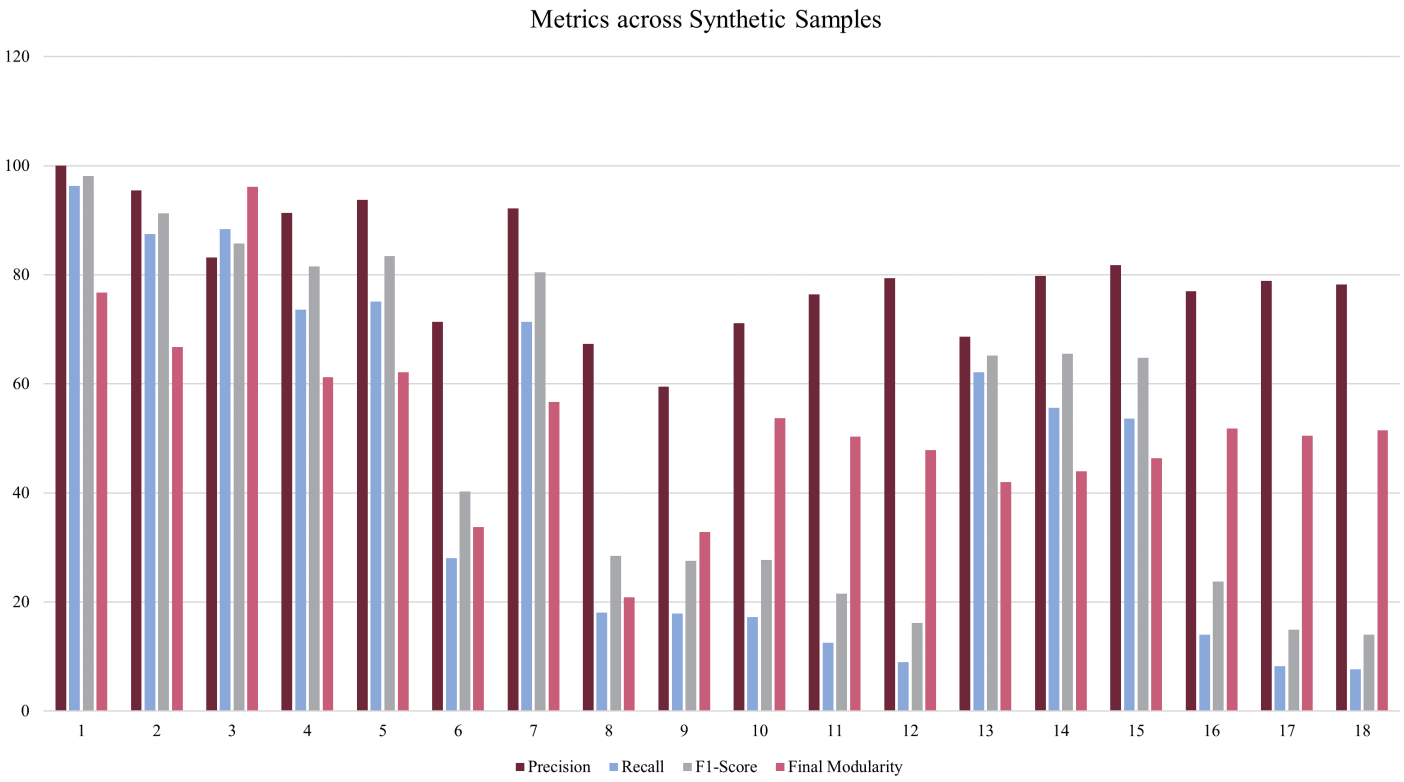


Fig. 6. A Bar Chart of Precision, Recall, f1-Scores, and Final Modularity after Running on ModER on the 18 Synthetic Samples.

necessarily recall.

D. Limitations

It is imperative and expected of course to understand the limitations of our approach here. First the 3 parameters that are controlled by the user β, σ and δ determine the sizes of the blocks. The size of the blocks eventually determines the sizes of the clusters in the modeled graph.

E. Takeaways and Future Work

In this work, we focused on the problem of unsupervised entity resolution for identifying unique entity profiles in ambiguous data. We defined ambiguous data as documents or records that do not adhere to a schema and come in unknown layouts and sometimes with inferior quality. In addressing this problem, We first focused on the meaning of similarity and matching. The problem lies in that if two documents are 100% similar, that does not mean that they refer to the same entity. In contrast, if 2 documents match at 10%, that does not mean that we are 100% sure that both do not refer to the same entity. Hence the inherent uncertainty and the nature of the problem. We started from the position that the only sure thing is that similarity between 2 documents of 50% is extreme uncertainty. Then assumed that below 50% of matches tend to ensure that the two documents do not refer to the same entity. On the other hand, two documents with a matching probability of more than 50% have some certainty that they might refer to the same entity. In the GDWM [15], we designed the system based on the later observation that higher mating probabilities should be detected with more certainty. While in ModER, we generalized to all cases. That generalization came with some cost in performance, but it was not that significant, and compared with other methods in similar conditions, it gave respectable results. The point is that unsupervised entity resolution is a complex problem that needs to be addressed in a more sophisticated way. In addition, the concept of string similarity needs to be reconsidered as traditional similarity functions are the actual bottlenecks in this process. Some deep learning, machine learning, and graph approaches introduced learned similarities, such as in [13] and in [47]. In addition, this problem of unsupervised entity resolution could be generalized to other topics in natural language processing and information retrieval since it resembles the entity recognition problem and the search problem.

VII. Conclusion

Here we introduced ModER, which stands for Modularity Composite Optimization for Entity Resolution, a framework combining multiple steps and algorithms. The method can quickly identify entity profiles in highly ambiguous data, overcoming the need to set matching thresholds. The method also limits user input to 3 parameters set using statistical percentile approximations. We based our work on the state-of-the-art unsupervised entity resolution, the DWM. In addition to the GDWM. To our knowledge, this technique has not been explored before. Our Composite Modularity Entity profiling step is innovative and can provide better results when benchmarked. In the future, we plan to address challenges such as the breakdown of high recall clusters even though they might not be imprecisely profiled. In addition, we address the memory-intensive bipartite approach posing a bottleneck for large

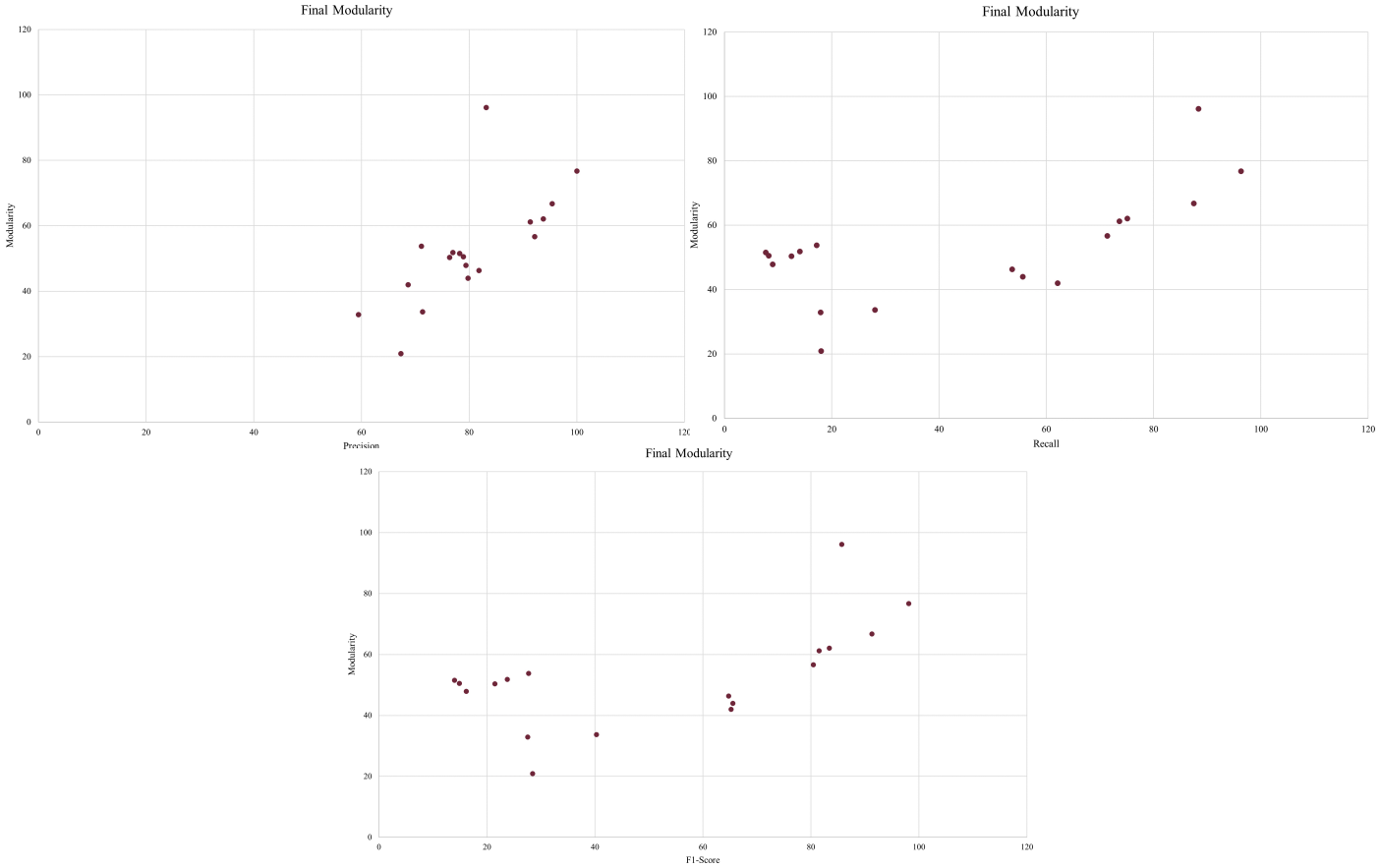


Fig. 7. A Scatter Plot of Values of Precision, Recall and f1-Score against Final Modularity Values on ModER after Running the 18 Synthetic Datasets.

profiles.

Conflict of Interest Statement

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Author Contributions

IAE created the ModER framework and wrote the article, JRT created the DWM framework and provided feedback and guidance on creating ModER as main PI, NH benchmarked the ModER framework and the GDWM framework and MAS tested the ModER framework and the GDWM framework on synthetic datasets and preprocessed the benchmark datasets.

Funding

This material is based upon work supported by the National Science Foundation under Award No. OIA-1946391.

References

[1] J. R. Talburt, D. Pullen, L. Claassens, R. Wang *et al.*, "An iterative, self-assessing entity resolution system: First steps toward a data washing machine," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 12, 2020.

[2] D. G. Brizan and A. U. Tansel, "A. Survey of Entity Resolution and Record Linkage Methodologies," vol. 6, no. 3, p. 11, 2006.

[3] J. R. Talburt and Y. Zhou, "A Practical Guide to Entity Resolution with OYSTER," in *Handbook of Data Quality: Research and Practice*, S. Sadiq, Ed. Berlin, Heidelberg: Springer, 2013, pp. 235–270. [Online]. Available: https://doi.org/10.1007/978-3-642-36257-6_11

[4] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate Record Detection: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 1, pp. 1–16, Jan. 2007, conference Name: IEEE Transactions on Knowledge and Data Engineering.

[5] V. I. Levenshtein *et al.*, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8. Soviet Union, 1966, pp. 707–710.

[6] M. S. Waterman, T. F. Smith, and W. A. Beyer, "Some biological sequence metrics," *Advances in Mathematics*, vol. 20, no. 3, pp. 367–387, 1976.

[7] T. F. Smith, M. S. Waterman *et al.*, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[8] M. A. Jaro and V. C. Walker, *Unimatch: A record linkage system: Users manual*. The Bureau, 1978.

[9] J. R. Ullmann, "A binary n-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words," *The Computer Journal*, vol. 20, no. 2, pp. 141–147, 1977.

[10] A. E. Monge, C. Elkan *et al.*, "The field matching problem: algorithms and applications." in *Kdd*, vol. 2, 1996, pp. 267–270.

[11] P. Jaccard, "Étude comparative de la distribution florale dans une portion des alpes et des jura," *Bull Soc Vaudoise Sci Nat*, vol. 37, pp. 547–579, 1901.

[12] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, pp. 11–21, 1972.

[13] D. Zhang, D. Li, L. Guo, and K. Tan, "Unsupervised Entity Resolution

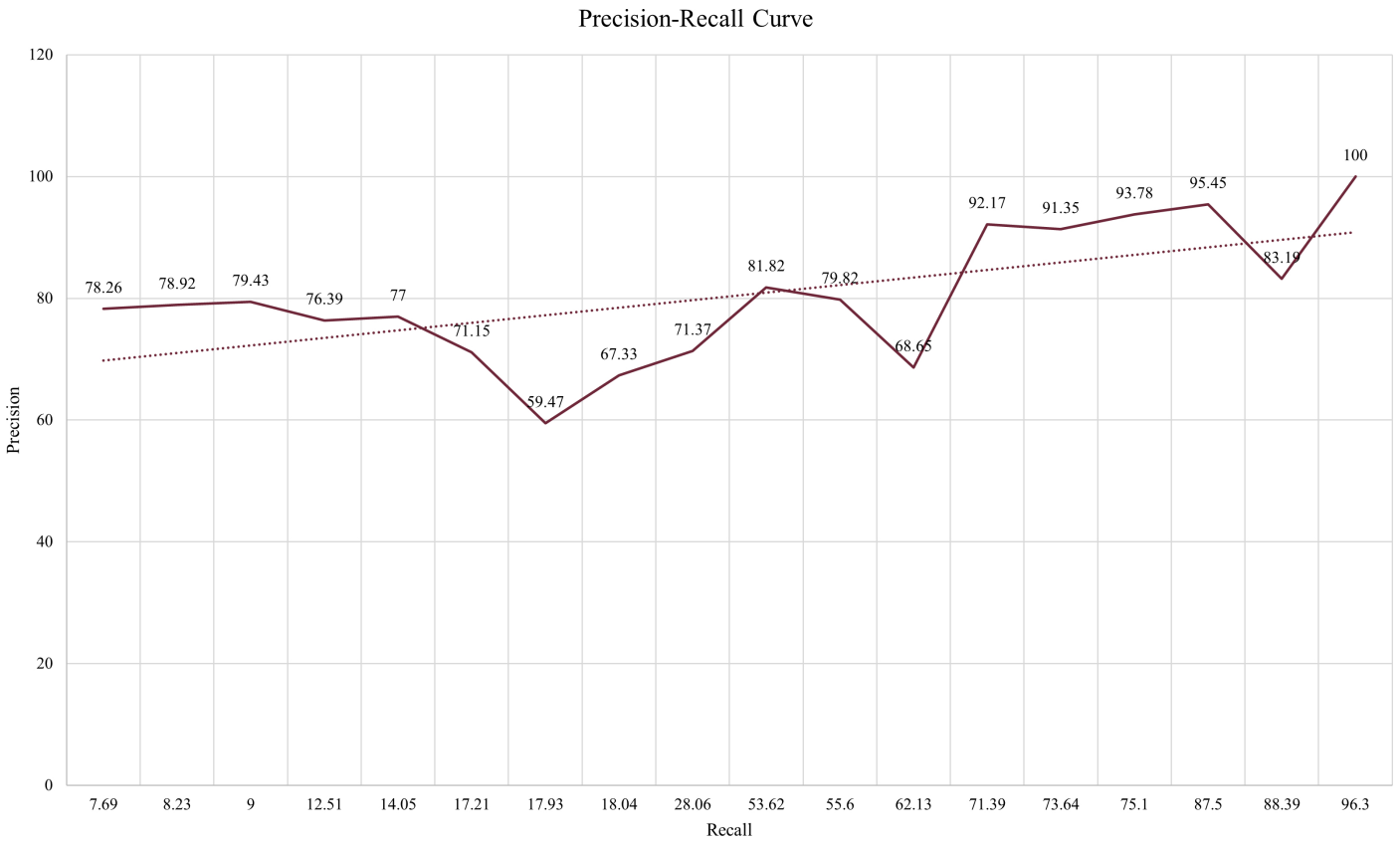


Fig. 8. Precision-Recall Curve.

with Blocking and Graph Algorithms," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020.

[14] B. Li, W. Wang, Y. Sun, L. Zhang, M. A. Ali, and Y. Wang, "GraphER: Token-Centric Entity Resolution with Graph Convolutional Neural Networks," *AAAI*, vol. 34, no. 05, pp. 8172–8179, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6330

[15] I. A. Ebeid, J. R. Talburt, and M. A. S. Siddique, "Graph-based hierarchical record clustering for unsupervised entity resolution," in *ITNG 2022 19th International Conference on Information Technology-New Generations*. Springer, 2022, pp. 107–118.

[16] J. Wang, H. T. Shen, J. Song, and J. Ji, "Hashing for Similarity Search: A Survey," *arXiv:1408.2927 [cs]*, Aug. 2014, arXiv: 1408.2927. [Online]. Available: http://arxiv.org/abs/1408.2927

[17] A. Broder, "On the resemblance and containment of documents," in *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, Jun. 1997, pp. 21–29.

[18] R. Wu, S. Chaba, S. Sawlani, X. Chu, and S. Thirumuruganathan, "Zeroer: Entity resolution using zero labeled examples," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1149–1164.

[19] Y. Li, J. Li, Y. Suhara, A. Doan, and W.-C. Tan, "Deep entity matching with pre-trained language models," *arXiv preprint arXiv:2004.00584*, 2020.

[20] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom, "Swoosh: a generic approach to entity resolution," *The VLDB Journal*, vol. 18, no. 1, pp. 255–276, 2009.

[21] G. Jeh and J. Widom, "SimRank: a measure of structural-context similarity," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '02. New York, NY, USA: Association for Computing Machinery, Jul. 2002, pp. 538–543. [Online]. Available: https://doi.org/10.1145/775047.775126

[22] F. Wang, H. Wang, J. Li, and H. Gao, "Graph-based reference table construction to facilitate entity matching," *Journal of Systems and Software*, vol. 86, no. 6, pp. 1679–1688, Jun. 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121213000484

[23] H. Wang, J. Li, and H. Gao, "Efficient entity resolution based on subgraph cohesion," *Knowledge and Information Systems*, vol. 46, no. 2, pp. 285–314, 2016.

[24] A. Saeedi, M. Nentwig, E. Peukert, and E. Rahm, "Scalable Matching and Clustering of Entities with FAMER," *Complex Systems Informatics and Modeling Quarterly*, vol. 0, no. 16, pp. 61–83, Oct. 2018, number: 16. [Online]. Available: https://csimq-journals.rtu.lv/article/view/csimq.2018-16.04

[25] U. Draisbach, P. Christen, and F. Naumann, "Transforming Pairwise Duplicates to Entity Clusters for High-quality Duplicate Detection," *J. Data and Information Quality*, vol. 12, no. 1, pp. 3:1–3:30, Dec. 2019. [Online]. Available: https://doi.org/10.1145/3352591

[26] L. Kolb, Z. Sehili, and E. Rahm, "Iterative computation of connected graph components with MapReduce," *Datenbank-Spektrum*, vol. 14, no. 2, pp. 107–117, 2014, publisher: Springer.

[27] N. Kang, J.-J. Kim, B.-W. On, and I. Lee, "A node resistance-based probability model for resolving duplicate named entities," *Scientometrics*, vol. 124, no. 3, pp. 1721–1743, Sep. 2020. [Online]. Available: https://doi.org/10.1007/s11192-020-03585-4

[28] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[29] M. Sadiq, S. I. Ali, M. B. Amin, and S. Lee, "A Vertex Matcher for Entity Resolution on Graphs," in *2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, Jan. 2020, pp. 1–4.

[30] D. Zhang, L. Guo, X. He, J. Shao, S. Wu, and H. T. Shen, "A Graph-Theoretic Fusion Framework for Unsupervised Entity Resolution," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, Apr. 2018, pp. 713–724, iSSN: 2375-026X.

[31] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences of the United States*

of America, vol. 103, no. 23, pp. 8577–8582, Jun. 2006. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1482622/

[32] L. Hagen and A. Kahng, "New spectral methods for ratio cut partitioning and clustering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 9, pp. 1074–1085, Sep. 1992. [Online]. Available: http://ieeexplore.ieee.org/document/159993/

[33] S. V. Ovchinnikov, "On the transitivity property," *Fuzzy Sets and Systems*, vol. 20, no. 2, pp. 241–243, Oct. 1986. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0165011486900801

[34] G. Navarro, "A guided tour to approximate string matching," *ACM computing surveys (CSUR)*, vol. 33, no. 1, pp. 31–88, 2001.

[35] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2011.

[36] E. Zhu, F. Nargesian, K. Q. Pu, and R. J. Miller, "LSH Ensemble: Internet-Scale Domain Search," *arXiv:1603.07410 [cs]*, Jul. 2016, arXiv: 1603.07410. [Online]. Available: http://arxiv.org/abs/1603.07410

[37] J. H. Burrows, "Secure Hash Standard," DEPARTMENT OF COMMERCE WASHINGTON DC, Tech. Rep., Apr. 1995, section: Technical Reports. [Online]. Available: https://apps.dtic.mil/sti/citations/ADA406543

[38] C. Y. Lee, "An algorithm for path connections and its applications," *IRE transactions on electronic computers*, no. 3, pp. 346–365, 1961.

[39] M. J. Barber, "Modularity and community detection in bipartite networks," *Physical Review E*, vol. 76, no. 6, p. 066102, Dec. 2007, arXiv: 0707.1616. [Online]. Available: http://arxiv.org/abs/0707.1616

[40] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, Oct. 2008, arXiv: 0803.0476. [Online]. Available: http://arxiv.org/abs/0803.0476

[41] H. Köpcke, A. Thor, and E. Rahm, "Evaluation of entity resolution approaches on real-world match problems," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 484–493, 2010.

[42] J. R. Talburt, Y. Zhou, and S. Y. Shivaiah, "Sog: A synthetic occupancy generator to support entity resolution instruction and research." *ICIQ*, vol. 9, pp. 91–105, 2009.

[43] K.-N. Tran, D. Vatsalan, and P. Christen, "Geco: an online personal data generator and corruptor," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013, pp. 2473–2476.

[44] Y. Ye and J. R. Talburt, "Generating synthetic data to support entity resolution education and research," *Journal of Computing Sciences in Colleges*, vol. 34, no. 7, pp. 12–19, 2019.

[45] P. Konda, S. Das, P. Suganthan G. C., A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. Naughton, S. Prasad, G. Krishnan, R. Deep, and V. Raghavendra, "Magellan: toward building entity matching management systems," *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 1197–1208, Aug. 2016. [Online]. Available: https://dl.acm.org/doi/10.14778/2994509.2994535

[46] J. P. Mower, "Prep-mt: predictive rna editor for plant mitochondrial genes," *BMC bioinformatics*, vol. 6, no. 1, pp. 1–15, 2005.

[47] Y. Lin, H. Wang, J. Chen, T. Wang, Y. Liu, H. Ji, Y. Liu, and P. Natarajan, "Personalized Entity Resolution with Dynamic Heterogeneous Knowledge Graph Representations," *arXiv:2104.02667 [cs]*, Apr. 2021, arXiv: 2104.02667. [Online]. Available: http://arxiv.org/abs/2104.02667