xER: An Explainable Model for Entity Resolution using an Efficient Solution for the Clique Partitioning Problem

Samhita Vadrevu
University of Illinois
at Urbana-Champaign
samhita3@illinois.edu

Wen-Mei Hwu
University of Illinois
at Urbana-Champaign
w-hwu@illinois.edu

Rakesh Nagi
University of Illinois
at Urbana-Champaign
nagi@illinois.edu

Jinjun Xiong
IBM T. J. Watson Research Center
Yorktown Heights, NY, USA
jinjun@us.ibm.com

Abstract

In this paper, we propose a global, self-explainable solution to solve a prominent NLP problem: Entity Resolution (ER). We formulate ER as a graph partitioning problem. Every mention of a real-world entity is represented by a node in the graph, and the pairwise similarity scores between the mentions are used to associate these nodes to exactly one clique, which represents a real-world entity in the ER domain. In this paper, we use Clique Partitioning Problem (CPP), which is an Integer Program (IP) to formulate ER as a graph partitioning problem and then highlight the explainable nature of this method. Since CPP is NP-Hard, we introduce an efficient solution procedure, the **xER** algorithm, to solve CPP as a combination of finding maximal cliques in the graph and then performing *generalized* set packing using a novel formulation. We discuss the advantages of using xER over the traditional methods and provide the computational experiments and results of applying this method to ER data sets.

1 Introduction

Entity Resolution (ER) is a prominent NLP problem, also referred to as co-reference resolution, de-duplication and record linkage, depending on the the problem set up. Irrespective of the name, the objective is to combine and cluster multiple mentions of a real-world entity from various data sources into their respective real-world entities and remove duplicates. Various techniques such as clustering (Aslam et al., 2004), (Saeedi et al., 2017), rule-based methods (Aumüller and Rahm, 2009), mathematical programming, and combinatorial optimization (Tauer et al., 2019) have previously been applied to ER. In this paper, we formulate and solve ER as a graph partitioning problem.

Representing ER as a graph partitioning problem The transformation from the real-world ER problem domain to the mathematical Integer Programming (IP) formulation setup is essential to understand the model's explainable nature and the solution procedure. A node in the graph represents a mention in the ER domain. An edge between any two nodes has a weight associated with it, representing the similarity score between the two mentions in consideration. This similarity score indicates the probability that these mentions are associated with the same entity. The goal is to ensure that based on the weights, the nodes are optimally allotted to their respective clusters. From a combinatorial perspective, this problem is known as the Clique Partitioning Problem (CPP). A clique is a complete subgraph in which all its nodes are pairwise connected. The weight of a clique is defined as the sum of all its edges' weights. The objective of this mathematical formulation is to find disjoint cliques in the graph such that the total weight of all the cliques is maximized, which, in the ER domain, translates to associating each mention to a single real-world entity with the highest probability association. The constraints in this mathematical formulation enforce that a particular node is mapped to just one clique and ensure that the mentions' transitivity conditions are obeyed.

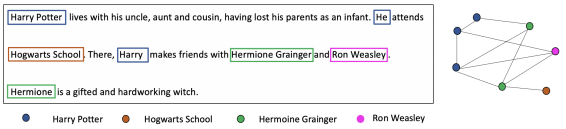

Figure 1: An example of converting a text into a graph.

Bhattacharya and Getoor (2004) was one of the earlier papers that formulated ER as a graphical problem and Bansal et al. (2004) proposed a correlation clustering method for the graphical problem. ER was also approached as a graph partitioning problem in (Nicolae and Nicolae, 2006), (Chen and Ji, 2009), (Chen and Ji, 2010) and the CPP approach outperformed other solution methods for

Proceedings of the First Workshop on Trustworthy Natural Language Processing, pages 34–44

ER (Finkel et al., 2005), (Klenner and Ailloud, 2009). Tauer et al. (2019) formulated ER as CPP, where an incremental graph partitioning approach was applied and solved using a heuristic. Lokhande et al. (2020) formulated ER as a set packing problem by considering the sets of all possible combinations of mentions and then choosing the best combination, based on the weights of the sets. ER has also been approached as a clustering problem. Saeedi et al. (2017) conducted an extensive survey on the clustering methods that had been applied to the entity resolution problem. von Luxburg (2007) solved ER as a spectral graph clustering problem, which is based on the graph's Laplacian matrix. Star Clustering (Aslam et al., 2004) formalizes clustering as graph covering and assigns each node to the highest probabilistic cluster. k-means is also a common technique to solve ER as a clustering problem. However, the mathematical formulation based methods come with a guarantee of optimality. Furthermore, it is easy to obtain an upper bound to these problems by relaxing the integer constraints. These upper bounds provide a guarantee on any feasible solution. In typical clustering algorithms, the number of clusters to produce in the output needs to be provided upfront, while it is decided by the model intrinsically in the CPP framework. The long convergence times and the iterations pose a disadvantage for them to be used as a solution technique for entity resolution (Saeedi et al., 2017). Moreover, from an explainability perspective, in the formulation-based methods proposed in this paper, the explanation is substantiated with mathematical guarantees, while the clustering-based approaches lack this mathematical precision and the heuristic nature further confounds explainability. Ribeiro et al. (2016), Ribeiro et al. (2018), Letham et al. (2015) and Choudhary et al. (2018) have proposed explainable systems for ER using local and if-then-else based global explanations. Ebaid et al. (2019) is a tool that provides explanations at different granularity levels.

Since CPP is NP-hard (Grötschel and Wakabayashi, 1989), a novel two-phase solution is proposed, in this paper, to solve CPP optimally. This solution method can be easily accelerated and scaled to handle large-sized datasets. As a part of this two-phased approach, new and creative formulations for the generalized set packing problem are also proposed. The formulations and the approach to obtain the optimal solution provide a mathematical guarantee on the output, and the results are easily interpretable and explainable. The constraints and objective function mathematically support the explanation behind the predicted output.

The rest of the paper is organized as follows. In Section 2, entity resolution is formulated as CPP. In Section 3, explainability and interpretability of this method is discussed. Section 4 then introduces the two-phase solution approach proposed for solving the NP-hard CPP. Sections 5 and 6 go over the computational experiments and the results.

2 Mathematical Formulation of CPP

As discussed in Section 1, the entity resolution problem is transformed to a graph where each mention is represented by nodes and the weight on an edge between the nodes is the similarity score between the mentions. To obtain the pairwise similarity scores, we use an open-source entity resolution library called Dedupe (Gregg and Eder, 2019), which applies blocking and a logistic regression based model to obtain the similarity scores between mentions. See Section 5 for more details about this.

In this section, the graph partitioning setup is formally represented by a mathematical formulation. Let i, j $(i < j)$ be two nodes in the graph (representing two mentions) and w_{ij} be the weight of the edge between these nodes. x_{ij} is a binary variable that denotes whether i, j are associated or co-referent (belong to the same clique).

$$x_{ij} = \begin{cases} 1 \text{ if nodes } i, j \text{ are associated} \\ 0 \text{ otherwise} \end{cases}$$

The "traditional" math formulation of CPP is:

$$CPP(\boldsymbol{w}) = \max \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} w_{ij} x_{ij}; \;\; s.t. \tag{1}$$

$$x_{ij} + x_{ik} - x_{jk} \leq 1, \;\; \forall 1 \leq i < j < k \leq N, \tag{2}$$

$$-x_{ij} + x_{ik} + x_{jk} \leq 1, \;\; \forall 1 \leq i < j < k \leq N, \tag{3}$$

$$x_{ij} - x_{ik} + x_{jk} \leq 1, \;\; \forall 1 \leq i < j < k \leq N, \tag{4}$$

$$x_{ij} \in \{0, 1\}, \;\; \forall 1 \leq i < j \leq N. \tag{5}$$

Constraints (2), (3), and (4) are the transitivity constraints enforced among the nodes. These three constraints ensure that if mention a is the same as b and b is the same as c, then it must also be that a is the same as c. The graph is assumed to be directed to avoid duplication of cliques and memory exhaustion. An optimal solution to this problem results in the best possible solution to the ER for the given similarity scores. However, due to cubic number of constraints, this particular formulation

for CPP, does not scale with the number of nodes. Hence, heuristics are prevalent to find an approximate solution to CPP; see Section 4 for details.

3 Model Explainability and Interpretability

Before we discuss our solution approaches, the explainable nature of this method is highlighted. The definitions of explainability have been studied in various works (Guidotti et al., 2019), (Arya et al., 2019). As defined in Danilevsky et al. (2020) and Guidotti et al. (2019), understanding the level of explainability of models can be interpreted as *outcome explanation problems*, where the emphasis lies in understanding the rationale behind the prediction of a specific output or all outputs in general. In this paper, the definitions and categorizations of explanations are based on the definitions in Danilevsky et al. (2020). Two major categorizations of explanations are emphasized. The first is based on the explanation process's target set, and divided into two types: Local and Global. Suppose the explanation is for a particular individual output. In that case, the explanation type is referred to as *Local*. On the other hand, if the explanation is for the whole model in itself, then it is a *Global* explanation. The second categorization is based on the origin of the explanation process. If the explanation is from the prediction process itself, then it belongs to the *Self Explaining* or the *Directly Interpretable* category (Arya et al., 2019). Otherwise, if post prediction processing is required to explain the output, it can be categorized as *Post-hoc* explanation.

As seen in Tauer et al. (2019), mathematical formulation based methods have a notion of optimality infused in the problems. The design of NLP problems like ER as mathematical formulations ensures that various constraints are met simultaneously, and hence making the output and the prediction process trustworthy and reliable. Since the constraints and the objective function are enforced into the mathematical formulation, the explanation behind any output comes directly from the model itself, making it a self-explainable model model. Moreover, the explanation behind any output is only dependent on the formulation and not on the output itself. This makes the model globally explainable. Therefore, by applying an efficient approach based on mathematical formulations, the solution method discussed in this paper presents an easily interpretable and explainable model for ER.

4 Solution Approach for CPP

As discussed in Section 2, CPP is NP-hard. In this paper, an efficient and scalable solution approach is proposed to solve the CPP.

The solution procedure is divided into two phases: Phase 1 involves finding the maximal cliques in the graph. A maximal clique is a clique that is not a sub-clique of a larger clique (Akkoyunlu, 1973). For Phase 2, we propose a novel *generalized* set packing formulation that not only ensures that each node belongs to a single clique, but it is able to break larger cliques into smaller sub-cliques if necessary. The formulation enables to find the optimal combinations of the cliques, that maximize the weight of the system. The algorithm (Phase 1 + Phase 2), is referred to as **xER** (Explainable ER).

4.1 Phase 1: Finding Maximal Cliques

In this phase, all the maximal cliques in a graph are found and stored. There are many approaches to find maximal cliques, but the most prominent and efficient approach is the Bron-Kerbosch (BK) algorithm (Bron and Kerbosch, 1973). There are multiple variants of BK, and in this paper, we adopt the pivot-based BK algorithm with node ordering. For simplicity, a recursion-based sequential implementation is used for BK. However, a scalable GPU-accelerated implementation for maximal clique listing is currently in progress based on (Almasri et al., 2021).

4.2 Phase 2: Set Packing

The output of Phase 1 is a list of cliques that are not disjoint. This phase aims to find the optimal combination of these cliques such that the cliques are disjoint and the total weight of all these disjoint cliques is maximized. Thus, Phase 2 is a maximum weighted Set Packing Problem (SPP). The original SPP is formulated as:

$$\text{(SPP)} \quad \max \quad W^T x \tag{6}$$

$$\text{s.t} \quad Ax = 1 \tag{7}$$

$$x \in \{0, 1\}. \tag{8}$$

Here, S is the list of sets (cliques) and V is the set of nodes in the graph. W denotes the weight vector, where each entry is the weight of a clique. The binary variable x_t denotes if a set $t \in S$ is chosen or not, $A : V \times S$ is the incidence matrix indicating the presence of a node in a set. $a_{it} \in A$ is 1 if node

$i \in V$ is in the set $t \in S$ and 0, otherwise. The formulation of the original set packing problem is designed to choose the optimal packing of sets that maximizes the system's overall weight. Multiple solution procedures have been developed to solve this set packing problem, and these procedures can be categorized as either exact or approximate algorithms. Rossi and Smriglio (2001) proposed a branch-and-cut approach for solving the SPP. Landete et al. (2013) proposed alternate formulations for SPP in higher dimensions and then added valid inequalities that were facets to the lifted polytope. Kwon et al. (2008) and Kolokolov and Zaozerskaya (2009) also proposed new facets that strengthen the relaxed formulations of SPP. Li et al. (2020) encoded SPP as a maximum weighted independent set and then used a Diversion Local Search based on the Weighted Configuration Checking (DLSWCC) algorithm to solve it. Since SPP is NP-hard (Garey and Johnson, 2009), many heuristics have also been proposed to obtain a solution for SPP in a reasonable amount of time. Rönnqvist (1995) proposed a Lagrangian relaxation based method and Delorme et al. (2004) used a greedy randomized adaptive search procedure (GRASP) to solve SPP. Gandibleux et al. (2004) proposed an ant colony heuristic for SPP.

Lokhande et al. (2020) has recently formulated ER as a set packing problem. All possible combinations of groups of mentions are given as an input to the SPP. Each of these groups is referred to as a hypothesis. Every hypothesis has a weight associated with it, which is computed as the sum of weights on a pair of nodes in that hypothesis. The best combination of the sets is chosen based on the weights. A major drawback of formulating and solving ER as a traditional set packing problem is the huge input size even for considerably small graphs. Table 1 shows a comparison between the number of cliques (|C|) and the number of maximal cliques (|MC|) in small-sized graphs, with number of edges denoted as |E|. The number of maximal cliques is significantly less than the total number of cliques. The number of all the cliques in the graph grows exponentially, much faster than the number of maximal cliques as the graph's size increases. In this paper, our proposed formulation for set packing can break a large set into smaller ones if required. Therefore, it only needs the maximal cliques as an input, contrary to SPP, which requires all the cliques as an input.

Nodes	|E|	|MC|	|C|	Ratio $\lceil \frac{\|C\|}{\|MC\|} \rceil$
38	147	70	528	8
38	203	101	801	8
38	379	433	5619	13
46	223	87	2466	28
46	317	162	3264	20
46	556	829	17114	21

Table 1: Statistics of small graphs and their associated edges.

4.2.1 Proposed SPP Formulation

As discussed in Section 4.2, the formulation of the original set packing problem is designed to choose the combination of sets that are disjoint and maximize the problem's overall weight. Thus, it requires the power set of cliques as an input. In this paper, the traditional set packing formulation is modified to fit the ER problem's requirements and made it more efficient and scalable to handle large datasets. Our novel formulation for set packing is introduced in Section 4.2 requires a much smaller input size. The formulation itself is enabled to carve out sub-cliques of a larger clique while keeping them disjoint. Eventually, the same optimal solution would be found, but the difference is in the manageable input size.

Notation: Here, K is the total number of maximal cliques in the input. Each set of index k, is denoted by S_k (cliques and sets are used interchangeably to accommodate the notation of both the traditional set packing and the new proposed formulation). The inputs to the problem is a set of incidence matrices $\{A_k\}$ corresponding to each set S_k, and W, the weight matrix of arcs in the original graph. The graph is directed, and an edge can only exist between two nodes i, j, with $i < j$ and weight W_{ij}. Each set can be broken down into multiple partitions, and M is the upper bound on the total number of partitions any set can be broken down into. The index for each partition of a set is m and is local to a set S_k, where $0 \leq m \leq M - 1$. z_{ij} denotes the connection between two nodes i, j in the optimal solution and y_{imk} denotes if node i is assigned to partition m of set S_k .

Decision Variables:

$$y_{imk} = \begin{cases} 1 \text{ if node } i \text{ is chosen for partition } m \text{ in set } S_k \\ 0 \text{ otherwise} \end{cases}$$

$$z_{ij} = \begin{cases} 1 \text{ if nodes } i, j \text{ belong to the same partition} \\ 0 \text{ otherwise} \end{cases}$$

All the nodes in V are ordered. E represents the edge set of the graph. $E = \{(i, j) : i < j\}$.

4.2.2 Quadratic Set Packing

The new set packing formulation is as follows:

$$\text{(QSP) max} \sum_{i=0}^{N-1} \sum_{j=i+1}^{N-1} W_{ij} z_{ij}; \quad s.t. \tag{9}$$

$$z_{ij} - \sum_{k} \sum_{m} y_{imk} \times y_{jmk} = 0, \forall i, j \in V, \tag{10}$$

$$\sum_{k} \sum_{m} y_{imk} \leq 1 \qquad \forall i \in V, \tag{11}$$

$$0 \leq z_{ij} \leq 1, y_{imk} \in \{0, 1\}, \quad \forall i, j \in V, m \in M, k \in K. \tag{12}$$

QSP stands for Quadratic Set Packing, deriving the name from the quadratic nature of the constraints. It can be observed that the notation of the variables in this formulation is different from the traditional set packing formulation. In the traditional set packing formulation, the decision variable is the binary variable x_t, denoting the presence of a set t in the optimal solution. However, in QSP, the decision variable y_{imk} denotes the presence of a node i in the partition m of set S_k. If a node i from set S_k should belong to partition m, the value of $y_{imk} = 1$ and 0 otherwise. This shows that y_{imk} is modified to remove nodes from the maximal cliques if necessary, eliminating the need to provide the power set of the maximal cliques as an input to the original SPP formulation. As mentioned before, this ordering avoids duplication of nodes and saves memory. Moreover, due to the nature of the formulation, even though z_{ij} is not explicitly assigned to be an integral solution, solving the QSP optimally results in an integer solution for z_{ij}. An off-the-shelf optimization solver, Gurobi (Gurobi Optimization, 2021) was used to solve the problem optimally. z_{ij} is used to compute the precision, recall and the F1 scores.

Algorithm 1: xER Algorithm

Result: **Resolved datasets with no duplicate mentions**

Step 1 : Perform blocking and compute pairwise similarity scores (§5);
Step 2 : Construct a directed graph with the mentions as nodes and similarity scores as weights on the edges. (§4);
Step 3 : Find maximal cliques in the graph using BK (§4.1);
Step 4 : Perform Set Packing using the QSP formulation (§4);
Step 5 : Use the output of z to compute precision, recall and F1 (§5);

Currently, we are working on developing scalable heuristics for the xER algorithm. As mentioned in Sec 4.1, a GPU accelerated version for Phase 1 is currently in progress based on Almasri et al. (2021). For Phase 2, an accelerated and scalable approach is being developed. The QSP formulation is linearized to provide the Linearized Set Packing (LSP) formulation. We are working on the linear relaxations of LSP and using accelerated computing to solve this and a family of relaxations. Subsequently, one can develop branch-and-bound approaches for solving the integer programming problem to optimality.

5 Computational Experiments

In this section, the xER algorithm's performance is evaluated through experiments on different ER datasets. In this paper, two primary data sources considered: benchmarking datasets (Saeedi et al., 2017) and ECB+ (Cybulska and Vossen, 2014). Datasets from both these sources are used to test the algorithm and analyze the algorithm's performance in terms of the F1 scores, solution times and their potential for scalability. Different blocking and scoring techniques have been applied to both these datasets, and are discussed in detail.

Blocking is a pre-processing technique applied to the datasets. The purpose is to eliminate the need to store similarity scores between those pairs of mentions that are extremely unlikely to being associated to the same entity. This increases the sparsity in the graph, making it easier to process the graph and perform computations. Blocking and similarity score computation techniques are different for different data sources and are discussed below.

5.1 Benchmarking Datasets

Saeedi et al. (2017) provides benchmark datasets, three of which are used in this paper. Table 2 shows the statistics for these benchmarking datasets. An open-source entity resolution library called Dedupe (Gregg and Eder, 2019) is used to preprocess these datasets by applying blocking techniques and generating similarity scores. The blocking technique and the scoring scheme are obtained from the code base of Lokhande et al. (2020). The dataset is divided into training and validation sets, with a split ratio of 50%. Our similarity scores for the benchmark datasets are obtained from the Dedupe library by training a ridge regression model.

5.2 ECB+ Corpus

Event Coreference Bank (ECB) (Bejan and Harabagiu, 2010)) is an event coreference resolu-

Dataset	Entities	Matches	Clusters
patent_example	2379	293785	102
csv_example	3337	6608	1162
settlements	3054	4388	820

Table 2: Statistics of the benchmarking datasets

tion dataset that includes a collection of documents found through Google Search. ECB+ (Cybulska and Vossen, 2014) is an extension of this dataset with newly added documents. Table 3 shows the statistics for this dataset.

The ECB+ dataset comes with the gold standard or the Ground Truth (GT) values used to generate the similarity scores. The ground truth values for two connected (or co-referent) and not connected mentions are $+1$ and -1, respectively. The "synthetic" similarity scores are generated from a normal distribution with a fixed mean and an added noise. If the ground-truth is $+1$ then $\mu = 0.5$ and if it is -1, then $\mu = -0.5$. A variance of 0.3 is added to the generated scores using this distribution. Once the similarity scores are computed, a blocking threshold T is applied to these scores. A pair of mentions with a similarity score less than T is blocked, and the edge between these nodes is removed from the original graph. The mentions in this dataset could belong to the *event* class or the *entity* class. The mention pairs are taken from the same class for the experiments, and xER is indifferent to the class.

Type	Mentions	Chains (clusters)
Event	6833	2741
Entity	8289	2224

Table 3: Statistics of ECB+ datasets

This dataset is broken down into smaller graphs using topic modelling from (Barhom et al., 2019). It facilitated the use of these different sized graphs to experiment with the blocking thresholds, analyze the F1 scores, and understand the xER algorithm's performance.

6 Results

The experiments are performed on an Intel i5 processor with 8GB RAM. The datasets from both sources are preprocessed and converted into graphs given as an input to the xER algorithm. These graphs have mentions as nodes and the pairwise similarity scores as the edges' weight. As shown in the xER algorithm (1), this graph is first passed through Phase 1, which is the Bron-Kerbosch algorithm with pivoting (Bron and Kerbosch, 1973). This step's output is a set of maximal cliques that are not disjoint and passed on to Phase 2 for the set packing step. QSP formulation is modelled using Gurobi (Gurobi Optimization, 2021) and solved optimally. The solution for the z variable from the optimally solved model is used to compute F1 scores. The xER algorithm is applied to all the datasets listed above and is evaluated in terms of F1 scores and computation times, and compared to the other competing algorithms. xER is also compared with the traditional set packing algorithm and the difference in the input sizes between SPP and QSP is highlighted through experiments. Also, to demonstrate the quality of the xER algorithm, the weights on the edges are replaced with Ground Truth (GT) values ($+1$ and -1) instead of similarity scores and tested. This helps in analyzing and confirming the model's consistency and accuracy, irrespective of the method used to compute similarity scores.

6.1 Testing xER on benchmarking datasets

Dedupe is used to perform blocking and compute the similarity scores as mentioned in Section 5.1. First, Dedupe employs specific blocking techniques on the data. A ridge regression model is then trained and used to compute the scores on the validation dataset. The pairwise nodes and the scores are passed on to the xER algorithm, and F1 scores are computed using the solutions from the z variable. These scores are obtained from the code base of (Lokhande et al., 2020) for a fair comparison and the performance of xER is compared with F-MWSP in (Lokhande et al., 2020) and a standard Hierarchical Clustering (HC) approach (Hastie et al., 2009). As mentioned before, M is a hyperparameter, and for these three datasets, we set it to 10. Table 4 shows that xER is at least as good as the other algorithms. For the *settlements* dataset, xER outperforms both F-MWSP and HC. For *csv_example*, xER has the same F1 score as F-MWSP, which is better than that of HC. For *patent_sample*, the F1 score for xER is less than HC and F-MWSP. However, since xER is designed to provide an optimal solution to a graph with a given set of nodes and weights, it is possible that the blocking techniques were too severe or the computational scores were not the best, leading to a lower F1 score. As discussed before, a high-

quality blocking technique and similarity scores will lead to high-quality F1 scores, since the xER algorithm is designed to give the best possible solution to a given input. Another comparison factor considered is the size of the input between SPP and QSP. The size of the input cliques required for a traditional set packing based formulation (F-MWSP) is significantly greater compared to that of the QSP formulation, which can be seen in the Table 1. Thus, a scalable xER algorithm can be useful to produce optimal outputs in lesser time. Moreover, with xER, the outputs and the explanations are supported by mathematical guarantees.

Datasets	Nodes	F1		
		xER	F-MWSP	HC
patent_sample	2379	92.0	**94.8**	92.2
csv_example	3337	**95.1**	**95.1**	94.4
settlements	3054	**95.7**	94.4	95.3

Table 4: Dedupe F1 scores

In addition to the F1 scores, other metrics have also been used to evaluate and compare the algorithms' performance. The dataset *settlements* is considered to analyze the algorithms in terms of all the evaluation metrics and is shown in Table 5.

Metric	xER	F-MWSP	HC
F1	**95.7**%	94.4%	95.3 %
Homogeneity	99.8%	99.8 %	**99.9**%
Completeness	**98.9** %	98.5 %	98.7%
V measure	**99.4** %	99.1 %	99.3 %
Adjusted Rand Index	**0.957**	0.944	0.953
Fowlkes Mallows	**0.958**	0.945	0.953

Table 5: Evaluation metrics for *settlements* dataset

6.1.1 Performance of xER on ECB+ Datasets

As discussed in Section 5, smaller datasets are constructed from the ECB+ dataset by performing topic wise modelling from (Barhom et al., 2019). Moreover, instead of performing entity resolution on the whole corpus, a subset of documents from the topics is considered as the input. Smaller datasets of different sizes are generated this way and are used to test and assess the xER algorithm.

After the similarity scores are computed, blocking techniques are applied based on a threshold of T on the similarity scores, in contrast to the blocking before the similarity score generation technique in the benchmarking datasets. The number of edges and the tightness among the nodes, measured by the Clustering Coefficient (CC) (Wang et al., 2017), is varied by varying this threshold T. The xER algorithm is also tested with the groundtruth values as weights. These tests are listed below and analyzed.

6.1.2 Tests Based on Thresholds

As described in Section 5.2, the similarity scores are generated from the normal distribution with means 0.5 and -0.5 depending on the ground truth, and the threshold values belong to the range $[-0.7, -0.2]$. As the threshold T increases, the graph's size becomes smaller due to the removal of edges with a weight less than the T. To demonstrate the impact of thresholding, a graph of 49 nodes is considered, and different graphs are generated from it by applying varying T values and the results are presented in Table 6.

T	|E|	CC	|C|	|MC|	F1	Time (s)		Total Time(s)	
						Phase 1	Phase 2	xER	SPP
-0.7	863	0.739	-	6425	97.33	0.199	9202.66	9202.86	-
-0.5	598	0.512	16619	827	97.33	0.017	4.772	4.780	99.88
-0.3	309	0.315	2906	174	97.33	0.0027	0.465	0.468	31.71
-0.2	228	0.312	2491	108	97.33	0.0017	0.294	0.296	30.2

Table 6: F1 for varying T on a graph of 49 nodes

The graph is denser and tightly connected with a tight threshold. The number of edges (|E|), the clustering coefficient (CC), the number of maximal cliques (|MC|) and the number of all the cliques in the graph (|C|) decrease with increasing T. For a particular T, the input size of SPP (|C|) compared to the input size of QSP (|MC|) is almost exponential and only increases with the graph's size. This difference is reflected in the solution times and can be seen that the SPP solution time is quite large when compared to the xER solution time. With larger graphs, the formulations will be unable to handle this large SPP input size. For the largest graph with $T = -0.7$, the computation time exceeded the time limit and was terminated. Another observation is that tighter thresholds lead to higher computation times for both phase 1 and phase 2. Thus, a higher T value is preferred in terms of solution time and memory management. However, it is possible that blocking with a higher threshold value might lead to a reduction in the recall and affect the F1 scores. So, a moderate threshold is preferred to balance both the F1 scores and the memory issues. T is treated as a hyperparameter, and the optimal T value can be chosen so that the graph size is small enough to handle, and the F1 scores are acceptable. When testing with ground truth values as weights, all the above graphs resulted in a 100% F1 score.

6.1.3 Evaluation: F1 scores

In addition to the thresholding tests, the xER algorithm is tested on other graphs generated using the same approach described above. The threshold value T is set to -0.3. The F1 scores for these graphs are reported in Table 7. As mentioned previously, xER is also tested using the groundtruth values as weights on the edges. xER results in a 100% F1 score when using the groundtruth, in all these datasets, which is also shown in Table 7. This implies that with the best possible scores (groundtruth), the algorithm works perfectly, which highlights the significance of high-quality similarity scores. M is set to 3 for all these graphs, and when the groundtruth is being used as weights, the value of $M = 10$.

This is because the input graph is fully connected because of no thresholding. So Phase 1 returns the whole graph as the maximal clique and phase 2 is responsible for partitiong the whole graph into smaller cliques, which is done using the M value. So a larger value of M enabled the graph to be partitioned into smaller sets as per the weights.

Nodes	Edges	F1-GT (%)	F1 (%)
46	317	100	97.43
135	2589	100	96.46
226	7727	100	91.62
262	10804	100	91.03

Table 7: F1 scores of graphs from the ECB+ dataset.

6.2 Explainability of xER

We now understand the model's explainable nature in an intuitive way with an example. The dataset with 49 nodes and $T = -0.3$ in Table 6 is considered. Three nodes: (7, 12, 20) that form a 3-clique or a triangle in the groundtruth are picked and analyzed. When xER is executed with weights, the thresholding does not remove one node: 25, that is connected to all these three nodes, thus having the potential to form a 4-clique. However, from the Table 8, the total weight that node 25 brings into the triangle is negative (-3.0) and thus, this 4-clique is not a good choice to be included in the optimal solution. Thus, the model automatically prevents this node from forming a 4-clique with the three nodes, thus ensuring that the precision wouldn't decrease. Another important observation is that blocking with a threshold of $T = -0.2$ would have removed the edge between the nodes 20 and 25, thus totally eliminating the potential of forming a 4-clique.

Node 1	Node 2	Weight	Node 1	Node 2	Weight
7	12	0.456	7	25	-0.076
7	20	0.999	12	25	-0.156
12	20	0.085	20	25	-0.253

Table 8: Weights on the edges of nodes (7, 12, 20, 25)

Another example of explainable ER and the importance of having high-quality scores, is considered for the same graph. Four edges: (3-15), (19-29), (21-25), (23-40) with weights $0.316, 0.095, 0.232, 0.046$, respectively, were included in the optimal solution, while these nodes are not connected in the ground truth. The "noisy" weights between these nodes which should have been negative per the ground truth. This shows that a poor scoring scheme can lead to a low quality solution.

As explained in Danilevsky et al. (2020), the explainability of a model can be evaluated in three ways: *Comparison with the groundtruth, Informal explanations and Human evaluation*. We compared the model with ground truth values and obtained the F1 scores. In addition to it, we also performed experiments with the groundtruth scores and the similarity scores to argue the reasoning behind a particular solution. For evaluation through informal explanation, we considered examples from graphs and understood the reasoning behind this output produced by the model. For future work, we plan to include a viable human evaluation technique for the ER problem.

In this paper, we compared our model to an existing approach for ER from Lokhande et al. (2020). As future direction of research, we aim to develop a scalable approach to handle large datasets that would not depend on an off-the-shelf solver to obtain optimal and explainable solutions(with mathematical guarantee), enabling us to compare the performance of xER with more approaches that have been used for ER.

7 Conclusion

A graph partitioning based approach is proposed to solve the entity resolution problem and is formulated as a clique partitioning problem. A node in the graph represents each mention, and the objective was to assign nodes to cliques optimally, and each clique represents a real-world entity. This mathematical formulation based model is inherently explainable. Since CPP is NP-Hard, a two-phased algorithm called xER is proposed and tested

on multiple datasets. Phase 1 of xER finds all the graph's maximal cliques, which is much more practical than finding all the cliques in the graph. Phase 2 is a generalized set packing formulation and has a much smaller input size than the traditional set packing problem. These contributions help develop a practical and easily parallelizable implementation for xER. xER shows promising performance in terms of accuracy.

A GPU accelerated approach for xER is in progress and will provide a scalable and practical model. Also, xER can be extended to other applications such as Topic modelling, Community Detection, Temporal Analysis. We believe this paper will lead the way to more mathematical formulation-based approaches and NLP problems can be solved using such highly explainable models, thus reducing the dependency on black-box models.

Acknowledgements

This work is supported by the IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as a part of the IBM AI Horizons Network.

References

E. A. Akkoyunlu. 1973. The Enumeration of Maximal Cliques of Large Graphs. SIAM J. Comput., 2(1):1–6.

Mohammad Almasri, Izzat El Hajj, Rakesh Nagi, Jinjun Xiong, and Wen mei Hwu. 2021. Accelerating K-Clique Counting on GPUs. In Proceedings of the 547th International Conference on Very Large Data Bases (VLDB), page submitted.

Vijay Arya, Rachel K. E. Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Q. Vera Liao, Ronny Luss, Aleksandra Mojsilović, Sami Mourad, Pablo Pedemonte, Ramya Raghavendra, John Richards, Prasanna Sattigeri, Karthikeyan Shanmugam, Moninder Singh, Kush R. Varshney, Dennis Wei, and Yunfeng Zhang. 2019. One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques. arXiv:1909.03012 [cs, stat]. ArXiv: 1909.03012.

Javed A. Aslam, Ekaterina Pelekhov, and Daniela Rus. 2004. The Star Clustering Algorithm for Static and Dynamic Information Organization. JGAA, 8(1):95–129.

David Aumüller and Erhard Rahm. 2009. Web-based affiliation matching. pages 246–256.

Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2004. Correlation Clustering. Machine Learning, 56(1-3):89–113.

Shany Barhom, Vered Shwartz, Alon Eirew, Michael Bugert, Nils Reimers, and Ido Dagan. 2019. Revisiting joint modeling of cross-document entity and event coreference resolution. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4179–4189, Florence, Italy. Association for Computational Linguistics.

Cosmin Bejan and Sanda Harabagiu. 2010. Unsupervised event coreference resolution with rich linguistic features. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1412–1422, Uppsala, Sweden. Association for Computational Linguistics.

Indrajit Bhattacharya and Lise Getoor. 2004. Iterative record linkage for cleaning and integration. In Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery - DMKD '04, page 11, Paris, France. ACM Press.

Coen Bron and Joep Kerbosch. 1973. Algorithm 457: finding all cliques of an undirected graph. Commun. ACM, 16(9):575–577.

Zheng Chen and Heng Ji. 2009. Graph-based event coreference resolution. In Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing - TextGraphs-4, page 54, Suntec, Singapore. Association for Computational Linguistics.

Zheng Chen and Heng Ji. 2010. Graph-based clustering for computational linguistics: A survey. 2010 Workshop on Graph-based Methods for Natural Language Processing, (July):1–9.

Pramit Choudhary, Aaron Kramer, and datascience.com team. 2018. datascienceinc/Skater: Enable Interpretability via Rule Extraction(BRL).

Agata Cybulska and Piek Vossen. 2014. Using a sledgehammer to crack a nut? lexical diversity and event coreference resolution. In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), pages 4545–4552, Reykjavik, Iceland. European Language Resources Association (ELRA).

Marina Danilevsky, Kun Qian, Ranit Aharonov, Yannis Katsis, Ban Kawas, and Prithviraj Sen. 2020. A Survey of the State of Explainable AI for Natural Language Processing. arXiv:2010.00711 [cs]. ArXiv: 2010.00711.

Xavier Delorme, Xavier Gandibleux, and Joaquin Rodriguez. 2004. GRASP for set packing problems. European Journal of Operational Research, 153(3):564–580.

Amr Ebaid, Saravanan Thirumuruganathan, Walid G. Aref, Ahmed Elmagarmid, and Mourad Ouzzani. 2019. Explainer: Entity resolution explanations. In 2019 IEEE 35th International Conference on Data Engineering (ICDE), pages 2000–2003.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL '05, pages 363–370, Ann Arbor, Michigan. Association for Computational Linguistics.

Xavier Gandibleux, Xavier Delorme, and Vincent T'Kindt. 2004. An Ant Colony Optimisation Algorithm for the Set Packing Problem. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Marco Dorigo, Mauro Birattari, Christian Blum, Luca Maria Gambardella, Francesco Mondada, and Thomas Stützle, editors, Ant Colony Optimization and Swarm Intelligence, volume 3172, pages 49–60. Springer Berlin Heidelberg, Berlin, Heidelberg. Series Title: Lecture Notes in Computer Science.

Michael R. Garey and David S. Johnson. 2009. Computers and intractability: a guide to the theory of NP-completeness, 27. print edition. A series of books in the mathematical sciences. Freeman, New York [u.a]. OCLC: 551912424.

Forest Gregg and Derek Eder. 2019. Dedupe.

M. Grötschel and Y. Wakabayashi. 1989. A cutting plane algorithm for a clustering problem. Mathematical Programming, 45(1-3):59–96.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2019. A Survey of Methods for Explaining Black Box Models. ACM Computing Surveys, 51(5):1–42.

LLC Gurobi Optimization. 2021. Gurobi optimizer reference manual.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. The Elements of Statistical Learning. Springer Series in Statistics. Springer New York, New York, NY.

Manfred Klenner and Étienne Ailloud. 2009. Optimization in coreference resolution is not needed: A nearly-optimal algorithm with intensional constraints. In Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, EACL '09, page 442–450, USA. Association for Computational Linguistics.

Alexander A. Kolokolov and Lidia A. Zaozerskaya. 2009. On Average Number of Iterations of Some Algorithms for Solving the Set Packing Problem. IFAC Proceedings Volumes, 42(4):1510–1513.

Roy H. Kwon, Georgios V. Dalakouras, and Cheng Wang. 2008. On a posterior evaluation of a simple greedy method for set packing. Optim Lett, 2(4):587–597.

Mercedes Landete, Juan Francisco Monge, and Antonio M. Rodríguez-Chía. 2013. Alternative formulations for the Set Packing Problem and their application to the Winner Determination Problem. Ann Oper Res, 207(1):137–160.

Benjamin Letham, Cynthia Rudin, Tyler H. McCormick, and David Madigan. 2015. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. The Annals of Applied Statistics, 9(3).

Ruizhi Li, Yupan Wang, Shuli Hu, Jianhua Jiang, Dantong Ouyang, and Minghao Yin. 2020. Solving the Set Packing Problem via a Maximum Weighted Independent Set Heuristic. Mathematical Problems in Engineering, 2020:1–11.

Vishnu Suresh Lokhande, Shaofei Wang, Maneesh Singh, and Julian Yarkony. 2020. Accelerating Column Generation via Flexible Dual Optimal Inequalities with Application to Entity Resolution. arXiv:1909.05460 [cs]. ArXiv: 1909.05460.

Cristina Nicolae and Gabriel Nicolae. 2006. BestCut: a graph algorithm for coreference resolution. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing - EMNLP '06, page 275, Sydney, Australia. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. KDD '16, page 1135–1144, New York, NY, USA. Association for Computing Machinery.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In AAAI Conference on Artificial Intelligence (AAAI).

Fabrizio Rossi and Stefano Smriglio. 2001. A set packing model for the ground holding problem in congested networks. European Journal of Operational Research, 131(2):400–416.

Mikael Rönnqvist. 1995. A method for the cutting stock problem with different qualities. European Journal of Operational Research, 83(1):57–68.

Alieh Saeedi, Eric Peukert, and Erhard Rahm. 2017. Comparative Evaluation of Distributed Clustering Schemes for Multi-source Entity Resolution. In Mārīte Kirikova, Kjetil Nørvåg, and George A. Papadopoulos, editors, Advances in Databases and Information Systems, volume 10509, pages 278–293. Springer International Publishing, Cham. Series Title: Lecture Notes in Computer Science.

Gregory Tauer, Ketan Date, Rakesh Nagi, and Moises Sudit. 2019. An incremental graph-partitioning algorithm for entity resolution. Information Fusion, 46:171–183.

Ulrike von Luxburg. 2007. A Tutorial on Spectral Clustering. arXiv:0711.0189 [cs]. ArXiv: 0711.0189.

Yu Wang, Eshwar Ghumare, Rik Vandenberghe, and Patrick Dupont. 2017. Comparison of Different Generalizations of Clustering Coefficient and Local Efficiency for Weighted Undirected Graphs. Neural Computation, 29(2):313–331.