ISSN 1816-6075 (Print), 1818-0523 (Online)
Journal of System and Management Sciences
Vol. 13 (2023) No. 4, pp. 16-39
DOI:10.33168/JSMS.2023.0402

TSE-HDR: Effective and Efficient Record Linkage Approach

Reham I. Abdel Monem, Ali Z. El Qutaany, Ehab E. Hasaneen

Information Systems Department, Faculty of Computers and Artificial Intelligence, Cairo University, Cairo, Egypt
reham@fci-cu.edu.eg (corresponding author)

Abstract. Entity Matching (EM) or record linkage is the task of identifying entities (objects, data instances) denoting the same real-world entity. Entity matching is an important and well-known problem and arises in numerous data fields such as data warehouses, information retrieval, heterogeneous databases, data mining, and all types of data analysis. While the state-of-the-art approaches to entity matching accomplish high accuracy, these approaches are computationally expensive for dealing with large data sets. Blocking, filtering, and features engineering techniques are usually utilized to solve this problem to avoid comparing all record pairs. In this paper, we present an approach called Tracking Similar Entities in Heterogeneous Data Records (TSE-HDR). It practices very efficiently and effectively when linking more than two databases (and dealing with applications that have duplication across/within databases). It uses blocking, filtering, and feature engineering techniques to detect similar entities in the big data and scales up the existing recent entity matching approaches called the Improved Common Block Scheme (ICBS), and suffix-based blocking (SUFFBLOCK) and (SAB) efficiency and effectiveness. Performance evaluation of the proposed approach against the ICBS SUFFBLOCK and SAB approaches over a real and big data set shows a great improvement in terms of effectiveness and efficiency. TSE-HDR can handle the tested dataset in a few seconds (while a current state-of-the-art technique requires many hours).

Keywords: Data integration, entity matching, heterogeneous data sources, feature engineering, blocking, filtering

1. Introduction

The problem of identifying different entities of the same real-world object is of paramount importance (Gu and Baxter 2003). This problem arises in the context of data warehouses (N. Polyzotis et al. 2008), heterogeneous databases (Al. 2013), and data analysis and mining (L.Jin et al. 2003)(Al. 2021) and pertains to data cleansing. Duplicate-free datasets are also needed in most modern applications including vital records, health care, and insurance and crime investigation, and become the data pre-processing step. In this work, we consider a problem termed record linkage or entity matching which is the procedure of detecting similar entities.

There has been a lot of attention on creating effective EM approaches to clear and combine data gathered from many sources (Z.Amrapali et al. 2018)(Wang et al. 2012). Enhancing data quality, facilitating, and enriching data analysis is the ultimate purpose of EM technology.

In more detail, similar entities are detected by most EM methods by comparing all features and pairs of entities. The cost for that is very expensive for large datasets (M.Bilenko et al. 2006). Many techniques were presented but some of them mainly focus on comparing entities using string similarity measures only. Depending only on string similarity measures is not accurate, especially when dealing with non-string features. Other techniques use machine learning methods to classify the entity pairs as "match," "non-match," or "possible match" However, these kinds of methods require a large amount of manually labeled data and are time-consuming. Another techniques apply pre-processing heuristics, called "Blocking" (Papadakis et al. 2019). Blocking divides records horizontally into different blocks and compares all features and pairs of records inside the block according to the similarity function. The accuracy accomplished by blocking is not sufficient to accomplish a good EM performance. Feature engineering to deal with large high-dimensional target datasets, redundant, dependent, extra, and non-significant features, which not only make computation more complex but also may reduce the quality of EM results is another challenge that doesn't considered by most of entity matching techniques.

Based on the aforementioned challenges, this study aims to address the following research questions:" How can entity-matching techniques detect similar entities in big data come from different heterogeneous data sources effectively and efficiently?" and "How can we detect similar entities without using any label data effectively and efficiently?" To provide an answer to these research questions; this paper presents an effective and efficient EM approach that accepts as input clean heterogeneous data integration results come from different and distributed data sources and generates a set of blocks, each block having duplicate entities. In comparison with the current state-of-the-art EM approaches, our approach outperforms existing EM approaches. Our approach operates as follows; first, it divides the features into different block types. In these different types of blocks, we apply a variety of similarity metrics to remove redundant features and maximize efficiency, because the same metric can perform differently for different types of features. Second, it eliminates from the target data set any time state-dependent, extra, and non-significant features. Third, it weighs each feature individually and chooses the best alternative of features to share with the EM process. Fourth, it uses a dynamically adjustable filtering technique. A set of applicants for each entity e_i is computed without all not matching with the entity e_i. Finally, we evaluate the performance of our approach by using a real-world dataset, present the empirical results, and discuss them.

The rest of this paper is organized as follows: Section 2 provides a literature review for record linkage. Section 3 introduces our proposed approach (Tracking Similar Entities in Heterogeneous Data Records). In Section 4, we present our experimental analysis. Finally, Section 5 concludes the paper.

2. Literature Review

There are different existing approaches to entity matching in the literature. In this section; some of the existing entity-matching approaches are briefly introduced.

In supervised approaches (S.Krawczyk et al. 2012), labeled data should be available and used to train models however, the cost of manually collected labeled records in big data is very high and are often unavailable. For this reason, research has focused on building unsupervised methods for entity matching.

Unsupervised approaches don't use any label data, they use similarity metrics and clustering algorithms to find similar entities. Although label data isn't used in unsupervised approaches, they often do worse than supervised approaches(S.Krawczyk et al. 2012).

In (O.Benjelloun et al. 2009) a generic entity matching approach and algorithms are introduced based on pairwise decisions. Pairwise means that the two records are matched at a time. In addition, they elaborate on the formal properties of an efficient algorithm. However, no solution is specified to discover the best settings for the algorithm.

Attribute-based methods are presented in (I.P.Fellegi and B.Suntib 1969) where an attribute denotes the property or characteristic of a record. In these approaches, a similarity value is calculated for each pair of records based on their attributes. If a similarity value is above a certain threshold, the records consider similar. Different similarity measures may be used for calculating a similarity value. In general, manual clarification is not practical in big databases. So, we need to use computerized methods.

To increase efficiency, approximate techniques were presented. Such methods use a preprocessing heuristic, called "Blocking". The reduction accomplished by blocking may effect in a major efficiency improvement. Numerous blocking-based techniques were presented including Multi-pass blocking (M.A. Hernández 1995)(L.Kolb et al. 2012), Q-gram-based indexing (M.Hadjieleftheriou et al. 2009), Canopy clustering (A.McCallum et al. 2000), Suffix blocking (A.Aizawa and K.Oyama 2005)(A.Allam et al. 2018), MFIBlocks (B.Kenig and A.Gal 2013), Meta-blocking (Papadakis et al. 2013), and a novel blocking scheme based on attribute value types ICBS (Zhu et al. 2018).

Few efforts discuss the feature selection problem in entity matching. Among them (G.Canalle et al. 2017) (J.Chen et al. 2012)(W.Su et al. 2010). In (J.Chen et al. 2012) a method for entity matching is introduced. This method searches using a training set for the best set of features to be used in the comparison step of the entity-matching process. First, for every feature, a suitable similarity measure is found, by investigating the F-measure value that every measure provided. Then, the sets of features are assessed, where the sets with the maximum values of F-measure are selected as the preeminent ones for the comparison step. In (W.Su et al. 2010) entity matching for results of queries in web data sources is introduced. In this situation, the work emphases on a machine learning algorithm that targets to modify the weights of the features for the similarity assessment. The algorithm can learn how to modify the weights of the features using a data sample that has instances without correspondence from diverse data sources. In (G.Canalle et al. 2017) an feature selection strategy for selecting relevant features for entity matching in data integration systems is presented for the assessment of related features using repetition and density criteria and using metadata associated with the data sources. It doesn't use other significant criteria in the feature selection process and doesn't eliminate all non-significant features.

Some EM techniques used partition-based filtering concepts to match entities. In this category, the set is divided into several non-overlapping segments. Similar entities are found in the same segment.

A signature scheme is created in PartEnum (A.Arasu et al. 2006) based on partitioning and enumeration. If two vectors with hamming distance lower than S are divided into S + 1 partitions with equal size, then they must be found in the same partition. The latter states that if these vectors are divided instead into m > S partitions with equal size, then they must have in shared not less than m − S partitions. A string is portioned into a set of segments PassJoin (G.Li and J.Wang 2011) and creates inverted indices for the segments created. Then, for every string, some of its substrings are selected and used to retrieve candidates from the index. Researchers suggest methods to decrease the number of segments required to get the candidate pairs. An approach to increase the pruning power of partition-based filtering is introduced in PTJ (Deng et al. 2016) by using a mixture of the subsets and their 1-deletion neighborhoods, which are subsets resulting from deleting one element. Essentially, pigeonhole principle is used to establish these methods. pigeonring principle (J.Qin and C.Xiao 2018) is an extension of previous methods, which systematizes the boxes in a circle and constrains the items number in several boxes rather than a single one, thus presenting tighter bounds. Pigeonring always creates a less or equal number of candidates than the pigeonhole principle by applying it to several similarity search problems. Pigeonring-based algorithms can be implemented on top of existing pigeonhole-based ones with slight changes.

We introduce a new effective and efficient approach Tracking Similar Entities in Heterogeneous Data Records (TSE-HDR) to match the entities not only according to blocking which divides entities

into different blocks according to similarity functions but according to the best alternative of selected features to address the limitations of all previously mentioned EM approaches and to take advantage of the feature selection concept advantage in reducing the number of features and thereby reducing the EM process. Additionally, it employs a filtering method to improve accuracy while reducing processing time.

3. Proposed Approach

In this section, our EM approach Tracking Similar Entities in Heterogeneous Data Records (TSE-HDR) is presented. Our goal in this paper is to present an entity-matching effective and efficient approach that takes as input heterogeneous data integration results that come from distributed and heterogeneous data sources and output a set of blocks, such that entities in the same block refer to the same entity and entities in different blocks refer to different entities.

We present a multi-source entity matching approach that contains two or more entity collections and can be done by applying deduplication to the union of all collections. domn is entities domain. E is a set of entities where E = {e_1, e_2,…., e_n}. Feature vector is defined as F = {f_1, f_2,...,f_d}. e = {x_1, x_2, x_3,…..x_d} where xd is the value of feature f_d in entity en.

TSE-HDR consists of three main components as presented in Fig. 1, feature engineering, evaluating the uniqueness measure of the best alternative of selected features, and filtering to find similar entities.

In the following sections, we present each component in detail.

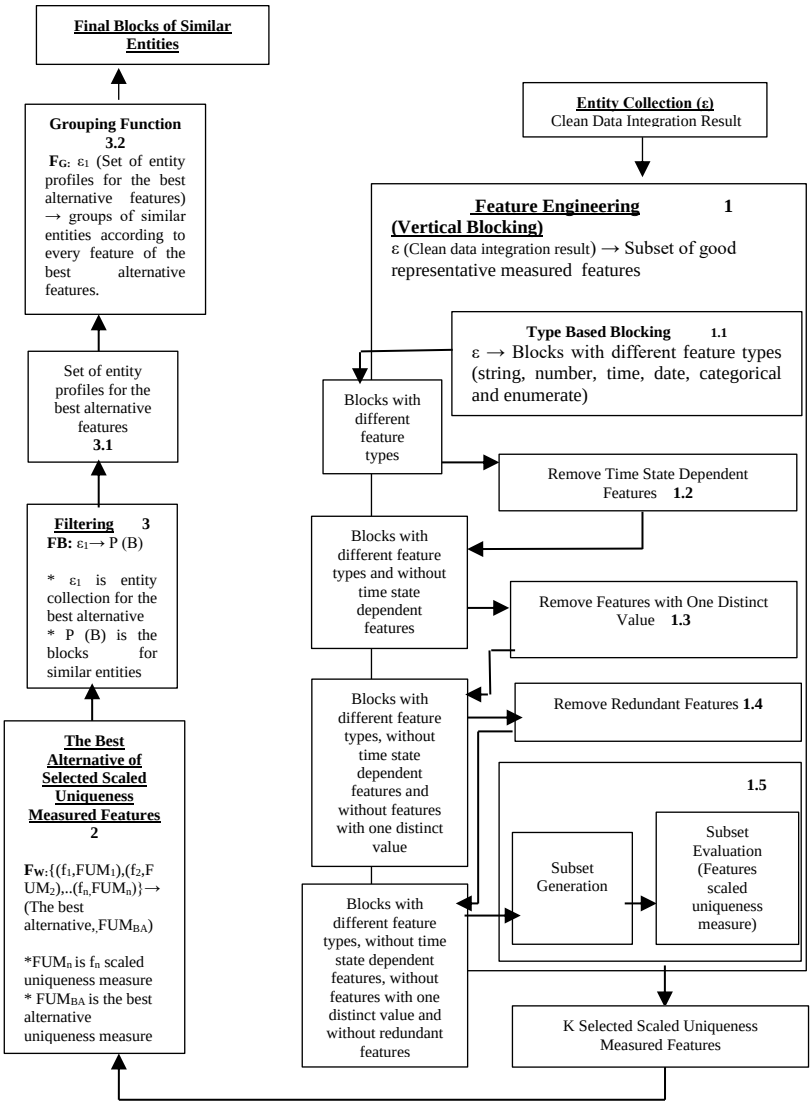

Fig. 1: TSE-HDR components

3.1. Feature engineering

Nowadays digital information increases rapidly, which is beneficial in scientific, engineering, and web and internet technology for making accurate decisions and predictions, etc. Methods for big data analysis are important since data mining approaches cannot deal with this big data. Big data has large and complex characteristics which are research areas now a day. Big data has large high dimensions that need the use of novel or adjusted feature engineering approaches.

This section focuses on the first component in our approach which is feature engineering which could help identify irrelevant features, remove them, and improve the model performance. (K.Janabi and K.Rusul 2018) (Honest 2020)(Muttakin et al. 2022)(Hussein 2022)(Haque et al. 2022)

The feature is "a variable in the dataset"; it is most often indicated as a column in a dataset. Feature engineering, also known as feature selection, is "a data mining technique that emphasizes choosing the best subset of features from the whole feature set". It accomplishes the best performance because it decreases over fitting, increases prediction accuracy, and decreases processing information computation time in terms of predetermined criteria. (Asim et al. 2020)

We use an open-source feature selection repository to assist and support the research on this topic. The most popular feature selection algorithms are found in this repository. (http://featureselection.asu.edu/).

Feature engineering (selection) does not create any new features as it uses the input features themselves to decrease their number. Feature selection produces more readable and accessible models while preserving the physical meanings of the original features. As a result, feature selection is favored in numerous applications, including genetic analysis and text mining.

The feature engineering process in this paper consists of four main tasks, type-based blocking, removing time state-dependent features, removing features with one distinct value, removing redundant features, and evaluating features' uniqueness measures.

3.1.1 Type-based blocking

In this paper, we focus on the preprocessing steps, which is the key part of EM to reduce time and increase scalability. Without them, EM has a quadratic time complexity, O (n^2), as we have to compare all entity profiles together. Decreasing this computational cost is the aim of several techniques from two main frameworks: Blocking and Filtering.

Blocking groups potentially match entities in the same blocks and compare entity profiles that exist in the same block. Blocking achieves a little lower effectiveness for higher efficiency. Its objective is to decrease the number of executed comparisons while omitting as few matches as possible. (Papadakis et al. 2019)

The first phase in the feature engineering process in this paper is distributing the dataset into non-overlapping blocks of the following feature value types: numeric-type block, string-type block, date-type block, categorical-type block, time-type block, and enumeration-type block. The splitting features into different block types algorithm presented in (Zhu et al. 2018) is used for this with some modifications like adding time and categorical types blocks. These modifications are presented in TSE-HDR type-based blocking algorithm in this paper.

An important step in resolving the EM problem is comparing feature values. Researchers developed a variety of methods for that (C.W. Cohen et al. 2003), which usually depend on string comparison techniques. However, there is still more work to be done about many categories of data, such as numerical, enumeration, time, category, and date, which could exist in the dataset (Q.Tan and F.Pivot 2015).

Splitting features into various blocks aims to encompass a range of adaptable similarity metrics to detect redundant features. As an enumeration type, the feature gender assists as an example, and the similarity of the enumeration type should be calculated using equality, as an alternative to the usual techniques used for string-type data. TSE-HDR type-based blocking algorithm presented below describes the functionality of splitting features into different blocks.

```
Input: (ε) entity collection;
       (δ) the threshold for the number of possible values of an enumeration
       feature;
       (λ) the number of values which are randomly selected from ε;

Output: Map < BT (block type), list of feature names > BT∈{NUME, DATE,
        STRING, TIME, ENUM, CATEG}

(1) Map < feature, List < v1, v2, … vλ>>mediateData ← ε;// Using Map to store
feature and its values.
(2) blockMap← new HashMap < String, List>; // blockMap is used to store the
return value;
(3) For each feature in mediateData Do{
(4)      valuesNoRep← Remove duplicate elements from List < v1, v2,…,vλ>;
(5)      n← valuesNoRep.size();
(6)      If ((double) n/λ < δ) then {//the type of this feature is enumeration
(7)         List enumFeatures ← blockMap.get("ENUM");
(8)         If (enumFeatures == null) then {enumFeatures← new ArrayList;
(9)            blockMap.put ("ENUM", enumFeatures);}
(10)           EnumFeatures ← enumFeatures.add(feature name);
(11)     Else if (the elements of valuesNoRep conform to the date type rules) then{
(12)           List dateFeatures ← blockMap.get("DATE");
(13)        If (dateFeatures == null) then {dateFeatures ← new ArrayList;
(14)           blockMap.put("DATE", dateFeatures);}
(15)        dateFeatures ← dateFeatures.add(feature name);
(16)     Else if (the elements of valuesNoRep conform to the time type rules) then {
(17)        List timeFeatures ← blockMap.get("TIME");
(18)        If (timeFeatures == null) then {timeFeatures ← new ArrayList;
(19)           blockMap.put("TIME", timeFeatures);}
(20)        timeFeatures ← timeFeatures.add(feature name);
(21)     Else if (The elements of listWithoutDu conform to the numerical
         type rules) then {
(22)        List numericFeatures ← blockMap.get("NUME");
(23)        If (numericFeatures == null) then {numericFeatures ← new ArrayList;
(24)           blockMap.put("NUME", numericFeatures);}
(25)        numericFeatures ← numericFeatures.add(feature name);
(26)     Else if (the elements of valuesNoRep conform to the categorical
         type rules) then {
(27)        List CATEGFeatures ← blockMap.get("CATEG");
(28)        If (CATEG Features == null) then { CATEG Features ← new
            ArrayList;
(29)           blockMap.put("CATEG", CATEG Features);}
(30)        CATEG Features ← CATEG Features.add(feature name);
(31)     Else {//other features will be treated as string type
(32)        List stringFeatures ← blockMap.get("STRING");
(33)        If (stringFeatures == null) then {stringFeatures ← new ArrayList;
(34)           blockMap.put("STRING", stringFeatures);}
(35)        stringFeatures ← stringFeatures.add(feature name);
(36)}End For each feature in mediateData
(37) return blockMap;
(Note: the δ and λ should be adjusted according to the size of dataset)
```

Algorithm 1: TSE-HDR type-based blocking

Using the following Covid19 example, we explain our approach.

Motivated example: Consider the twelve people's records shown in the Covid19 example that are to be resolved. We would like to detect records that refer to the same person. The column named record isn't the primary key for records it is just the identification number to which the record we refer. In this paper, we suppose the absence of a primary key, e_i should take only one type of vaccination and dose, and the "Null" value means that the feature value doesn't exist, for example, null in the VAX-NAME feature, means that e_i doesn't take any vaccine.

Table 1: Covid19 applied example to TSE-HDR on clean data integration result (ε)

VaxDate	VaxName	CoronaDate	CoronaResult	Hospital	LossOfSmell	LossOfTaste	DifficultyBreathing	HeadAch	SoreThroat	Fever	Cough	Phone	BloodGroup	WorkPlace	DateBirth	Name	Record
4/28/21	Sinopharm	NULL	Negative	AlAmal	0	1	0	0	0	0	0	0101	A+	ElectricCompany	6/19/88	AhmedTarek	e_1
4/30/21	Sinopharm	NULL	Negative	SuezCanal	0	1	0	0	0	0	1	0102	A+	Retired	2/1/48	KhaledAli	e_2
NULL	NULL	NULL	Negative	ElAgoza	0	1	0	0	0	1	0	0103	B+	GizaShcool	5/6/98	TarekNabil	e_3
5/23/21	Sinopharm	NULL	Negative	ElAgoza	1	1	1	0	0	0	1	0121	A-	GizaShcool	2/2/73	WardAli	e_4
5/22/21	Sinopharm	NULL	Negative	SuezCanal	1	1	0	1	0	0	1	0122	O+	SuezPort	4/3/77	NaderMohamed	e_5
5/26/21	AstraZeneca	5/11/2020	Positive	SuadiAlmani	0	0	0	0	0	0	1	0123	O+	Retired	5/20/61	SaraTarek	e_6
6/2/21	AstraZeneca	NULL	Negative	AlAmal	1	1	0	0	0	0	1	0104	AB+	PetrolCompany	11/23/84	ReemSamir	e_7
6/4/21	AstraZeneca	10/5/2020	Positive	ElAgoza	1	1	1	1	0	0	1	0105	AB+	FertilizerCompany	4/6/79	MarwanAnowar	e_8
4/28/21	Sinopharm	2/1/2021	Positive	ElAgoza	0	0	1	1	1	0	1	0106	A+	PetrolCompany	12/15/88	AhmedTarek	e_9
5/26/21	AstraZeneca	2/1/2021	Positive	ElAgoza	1	1	0	0	0	0	1	0107	O+	Retired	5/20/61	SaraTarek	e_{10}
6/4/21	AstraZeneca	10/5/2020	Positive	ElAgoza	1	1	1	0	0	0	1	0105	AB+	FertilizerCompany	4/6/79	MarwanAnowar	e_{11}
NULL	NULL	4/4/2021	Positive	SuadiAlmani	1	1	0	0	0	1	0	0108	B-	SuezPort	11/22/87	TarekNabil	e_{12}

By applying TSE-HDR type-based blocking algorithm to ε (selected dataset / clean data integration result's entity collection) in the Covid19 example, given δ=0.2 (δ is the threshold for the number of possible values of an enumeration feature), λ= 12 (λ is the number of values that are randomly selected from ε), δ and λ should be adjusted according to the size of the dataset, the type-based blocks are created as follows

Following we give an example of one feature from each block type:
STRING-Type
Map < NAME, List < AhmedTarek, KhaledAli, TarekNabil, WardAli, NaderMohamed, SaraTarek, ReemSamir, MarwanAnowar, AhmedTarek, SaraTarek, MarwanAnowar, TarekNabil>>
DATE-Type
Map <DATE_BIRTH, List <6/19/88, 2/1/48, 5/6/98, 2/2/63, 4/3/47, 5/20/61, 11/23/84, 4/6/79, 12/15/88, 5/20/61, 4/6/79, 11/22/87>>
ENUM-Type
Map < COUGH, List <0,1,0,1,1,1,1,1,1,1,1,0>>
CATEG-Type
Map < BLOOD_GROUP, List <A+, A+, B+, A-, O+, O+, AB+, AB+, A+, O+, AB+, B->>

By removing duplicate elements from List < v_1, v_2,…,v_λ>
valuesNoRepNAME ← List < AhmedTarek, KhaledAli, TarekNabil, WardAli, NaderMohamed, SaraTarek, ReemSamir, MarwanAnowar>
valuesNoRepDATE_BIRTH ← List <6/19/88, 2/1/48, 5/6/98, 2/2/63, 4/3/47, 5/20/61, 11/23/84, 4/6/79, 12/15/88, 11/22/87>
valuesNoRepCOUGH ← List <0, 1>
valuesNoRepBLOOD_GROUP ← List <A+, B+, A-, O+, AB+, B->

The numbers of not repeated values for a sample of features are
n_NAME← 8
n_DATE_BIRTH← 10
n_COUGH← 2
n_BLOOD_GROUP← 6

Determining enumeration features
NAME = 8/12 = 0.67 not enumeration feature
DATE_BIRTH = 10/12= 0.83 not enumeration feature
COUGH = 2/12 = 0.16 enumeration feature
BLOOD_GROUP = 6/2 = 0.5 not enumeration feature
enumFeatures← {LOSE_OF_SMELL, LOSE_OF_TASTE, DIFFICULTY_BREATHING, SORE_THROAT, HEAD_ACHE, FEVER, COUGH, CORONA_RESULT}

Determining date features
dateFeatures ← {VAX_DATE, CORONA_DATE, DATE_BIRTH}
Determining categorical features
categoricalFeatures ← {VAX_NAME, BLOOD_GROUP}
Other features are treated as string type
stringFeatures← {NAME, PHONE, HOSPITAL, WORK_PLACE}

Map <ENUM, {LOSE_OF_SMELL, LOSE_OF_TASTE, DIFFICULTY_BREATHING, SORE_THROAT, HEAD_ACHE, FEVER, COUGH, CORONA_RESULT}>
Map <TIME, {}>
Map <DATE, {VAX_DATE, CORONA_DATE, DATE_BIRTH}>
Map <NUME, {}>
Map <STRING, {NAME, PHONE, HOSPITAL, WORK_PLACE}>
Map <CATEG, {BLOOD_GROUP, VAX_NAME}>

3.1.2. Remove redundant, time state-dependent features, features with only one distinct value, and evaluate features' uniqueness measure

The dimensions of data increase every day and this gets new challenges to the efficiency and effectiveness of many existing EM methods. In this paper, we present a feature selection method to remove time-state-dependent features, features with one distinct value, and redundant features, and evaluate the features' uniqueness measure.

Time state-dependent features (features that change their values with time) (A.Z.ElQutaany et al. 2018) are determined from Meta Fusion Repository (MFR) and are removed. MFR contains features state, time state dependent feature or time state independent feature, and fusion policies to identify, match and fuse the entities.

Features with only one distinct value are not distinguishable and hence not be considered.

For redundant features, we can't find redundant features in the input of the data integration result as they are removed through the mapping task (A.Z.ElQutaany et al. 2010) before the integration task but for other inputs of data, redundant features should be removed. To remove redundant features, we divide features inside the same type of block into non-overlapping blocks according to the similarity or correlation of the feature values, which is based on the concept of redundancy. Removing redundant features is presented in the attribute clustering algorithm (Zhu et al. 2018). In this paper, we update the attribute clustering algorithm by eliminating more unnecessary features as time-state-dependent features and features with one distinct value and evaluating the uniqueness measure of selected features. The TSE-HDR feature engineering algorithm presents these modifications.

Finally, the features' uniqueness measure (FUM) evaluation consists of two basic steps, subset generation, and subset evaluation. The subset generation objective is to generate a subset of candidate features. The evaluation of a candidate subset is according to a certain evaluation criterion. In TSE-HDR, the feature evaluation criterion or feature uniqueness measure represents the number of distinct values to their total number of values. Feature uniqueness measure is then scaled to the total uniqueness measures of selected features. The scaled feature uniqueness measure represents the minimum accuracy of the resulting similar entities obtained by the feature alone. Accuracy is increased by adding more features so the best alternative from features is created.

$$\text{Feature\_Uniqueness\_Measure } (f_i) = \frac{\text{Number of distinct values } (f_i)}{\text{Total number of values } (f_i)} \quad (1)$$

$$\text{Scaled Feature\_Uniqueness\_Measure } (f_i) = \frac{\text{Feature\_Uniqueness\_Measure (FUM) } (f_i)}{\text{Scaled ratio}} \quad (2)$$

$$\text{Where Scaled ratio} = \frac{1}{\sum_{i=1}^{F} \text{Feature\_Uniqueness\_Measure } (f_i)} \quad (3)$$

The following is a modified attribute clustering algorithm by removing time state-dependent features, features with one distinct value, and redundant features, and adding features' uniqueness measure evaluation.

This modification decreases the number of selected features and hence decreases the number of compared values, computational cost, and time to detect similar entities.

```
Input: {b1, b2,....bi} ∈ B //B is set for all type based blocks;
       S← Map< feature name, List v 1, v 2,..., v n >//this is for features in
       same type block;
       (δ) the threshold for the features similarity;

Output: features with their scaled uniqueness measure;

(1) For each bi ∈ B do{
//Removing time state dependent features
(2)    using MFR find time state dependent features; //MFR is meta
       fusion repository
(3)    noOfFeatures ← S.size();
(4)    For (int i = 0; i < noOfFeatures; i++) do {
(5)        if (fi is time state dependent feature)
(6)            S.remove (fi);}
(7)    return S1←S;
//Removing features with one distinct value
(8)    noOfFeatures ← S1.size();
(9)    For (int i = 0; i < noOfFeatures; i++) do{
(10)       valuesNoRep← Remove duplicate elements from fi values;
(11)       n← valuesNoRep.size();
(12)       If ((n==1)
(13)          S1. remove (fi);}
(14)   return S2← S1;
//Removing redundant features
(15)   noOfFeatures ← S2.size();
(16)   For (int i = 0; i < noOfFeatures; i++) do{
(17)       For (int j = i + 1; j < noOfFeatures; j++) do{
(18)           Sim i, j← feature[i].getSimilarFeatures (feature
               [j]);//refer to the formula (1)~(4) in [1]
(19)           If (Sim i, j > δ)
(20)           S2.remove (fi);}}
(21)   return S3←S2;}
(22) For each bi ∈ B do{
(23)   If (bi.size() == 0) then B.remove(bi);}
(24) Return B; // the features' type blocks {b1, b2,....bi}without time state
dependent features, features with one distinct value and redundant features}
// Evaluating features uniqueness measure (FUM)
(25) For each bi ∈ B do{
(26)   noOfFeatures ← bi.size();
(27)   For (int i = 0; i < noOfFeatures; i++) do{
(28)       FUM (fi) ← fi.getFUM;// refer to the formula (1)~(3)
(29)       return list scaledFUM<(feature name, scaledFUM (fi) )> sorted
           decinding with scaledFUM (fi) //list for all features' name and their
           uniqueness measure from all type blocks}}
```

Algorithm 2: TSE-HDR feature engineering

By applying the TSE-HDR feature engineering algorithm to the Covid19 example, and by checking MFR, the time state-dependent features are {HOSPITAL, WORK_PLACE, PHONE, CORONA_DATE, LOSE_OF_SMELL, LOSE_OF_TASTE, DIFFICULTY_BREATHING, HEAD_ACHE, SORE_THROAT, FEVER, COUGH, CORONA_RESULT}. They should be removed for changing their values over time.

By measuring the distinct values for each feature in each block type; STRING_Type Block ← {NAME}, DATE_Type Block ← {VAX_DATE, DATE_BIRTH}, and CATEG_Type Block ← {BLOOD_GROUP, VAX_NAME}; there isn't any feature with one distinct value, so no feature is removed.

For removing redundant features, we haven't any redundant features as our approach TSE-HDR input is a clean data integration result.

For features' uniqueness measure presented in Table 2 and Table 3, we use equations from (1)~(3), and the generated feature set { NAME, VAX_DATE, DATE_BIRTH, BLOOD_GROUP, VAX_NAME}

Table 2: Uniqueness measure evaluation for selected features

Feature Name	Feature_Uniqueness_Measure	Scaled Feature _Uniqueness_Measure
NAME	0.67	0.229
DATE_BIRTH	0.83	0.284
VAX_DATE	0.67	0.229
BLOOD_GROUP	0.5	0.171
VAX_NAME	0.25	0.087

Table 3: k ranked set of scaled uniqueness measured features

Feature Name	Feature_Uniqueness_Measure	Scaled Feature _Uniqueness_Measure
DATE_BIRTH	0.83	0.284
VAX_DATE	0.67	0.229
NAME	0.67	0.229
BLOOD_GROUP	0.5	0.171
VAX_NAME	0.25	0.087

3.1.3. Create the best alternative from selected features

The alternative is one or more selected features and its scaled uniqueness measure can be measured using equation (4). Alternative scaled uniqueness measure represents the minimum accuracy of the resulting similar entities obtained by the alternative. The best alternative is determined by the administrator according to his system requirements and according to the specified accuracy threshold.

$$\text{Scaled\_Uniqueness\_Measure}\ (\text{Alternative}_i) = \sum_{i=1}^{n} \text{Scaled Feature\_Uniqueness\_Measure}\ (f_i) \qquad (4)$$

Where n is the number of shared features that form the alternative

Input: scaledUniquenessMeasureSelectedFeatures < (feature name, scaledUniquenessMeasure (f_i)) >// list for all features' name and their uniqueness measure from all type block sorted descending with scaledUniquenessMeasure (f_i)
(δ) minimum accuracy of the resulting similar entities obtained by the alternative

Output: theBestAlternative and its uniqueness measure

(1) theBestAlternative ← {(f_0, scaledUniquenessMeasure (f_0)}; // f_0 is the feature with the highest uniqueness measure
(2) noOfFeatures← scaledUniquenessMeasureSelectedFeatures.size();
(3) For (int i = 1; i <noOfFeatures; i++) do{
(4) if (theBestAlternative.scaledUniquenessMeasure ≥ δ) {
(5) return theBestAlternative;
(6) return theBestAlternativeScaledUniquenessMeasure;}
(7) else {
(8) theBestAlternative ← theBestAlternative + f_i ;
(9) theBestAlternativeScaledUniquenessMeasure← Alternative.getAlternativeScaledUniquenessMeasure; //refer to the formula (8)}}

Algorithm 3**:** The best alternative of selected features and its scaled uniqueness measure evaluation

In more detail, the best alternative of selected features and its scaled uniqueness measure evaluation algorithm proceeds as follows. We take as inputs the selected features' names and their uniqueness measure; these features are ranked descending with their uniqueness measure, and the minimum accuracy threshold of the resulting similar entities. The output is the best alternative and its uniqueness measure Line 1, initializes the best alternative with the first feature in the list of selected features, and line 4 checks for the best alternative uniqueness measure if it is greater than or equal to the minimum accuracy threshold, we return the best alternative otherwise, we iterate using for loop on the remaining features and adding the uniqueness measure of new feature to the uniqueness measure for the last best alternative as presented in line 8. We check for the best alternative uniqueness measure after adding each new feature. We stop to iterate on the list of selected features when we reach the end of the list or when the best alternative uniqueness measure is greater than or equal to the minimum accuracy threshold.

By applying the best alternative of selected features and its scaled uniqueness measure evaluation algorithm for the Covid19 example, the best alternative of selected features to get similar entities according to the specified accuracy threshold equals 0.6 are presented in Table 4.

Table 4: The best alternative of selected features and its uniqueness measure

The Best Alternative Name	The Best Alternative Features	The Best Alternative_ Uniqueness_Measure
The Best Alternative	DATE_BIRTH, VAX_DATE, NAME	0.742

3.1.4. Filtering to get similar entities

Filtering methods take an entity collection as input and output pairs of similar entities that satisfy a predefined criterion. The fundamental steps to obtaining blocks of similar entities are described below.

Entity profiles

Entity profiles describe a real entity in the form of (name, value) pairs. It is the core of entity matching (EM) and entity resolution (ER). An entity profile is formally defined as follows, given infinite sets of feature names N, feature values V, and unique identifiers I (Papadakis et al. 2019):

Definition 1 (Entity Profile). An entity profile e_{id} is a tuple (id, Bid), where id ϵ ID is a unique identifier, and Bid is a set of name-value pairs (n, val), with n ϵ N and val ϵ (V $\cup$ ID). A group of entity profiles ε is named entity collection. In this section entity profiles or entity collection (ε_1) are entity profiles after the feature selection process. (Papadakis et al. 2019)

Two entity profiles e_i and e_j match, $e_i \equiv e_j$, if they denote the same real-world entity. Finding all matching entities in an entity collection or between two or more entity collections is the task of Entity Matching (EM).

From Table 4, the best set of features to detect similar entities is {NAME, DATE_BIRTH, VAX_DATE}, so we form a set of entity profiles each profile represents an entity from the data set in terms of these best representative features.

In this paper, we find matched entities using the best alternative of selected features entity profiles mentioned below in Fig. 2.

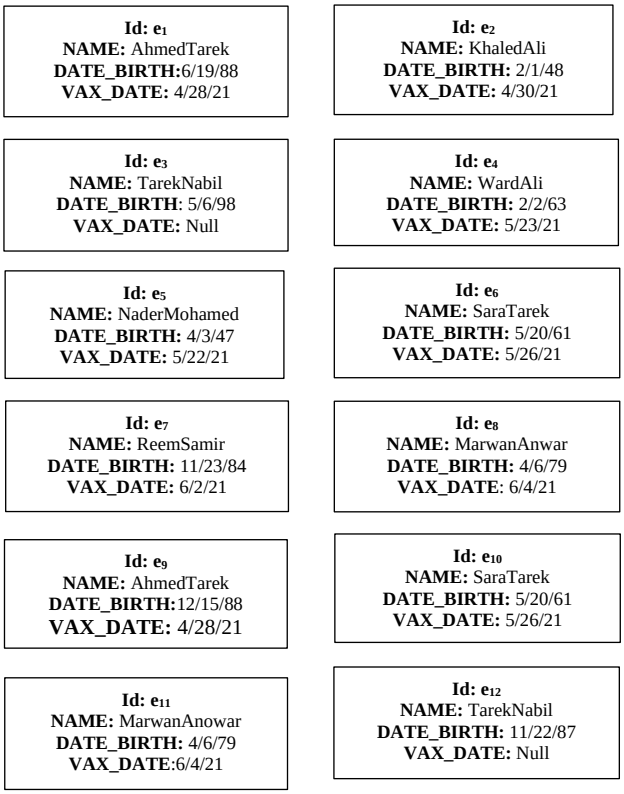

Fig. 2: Entity profiles

Grouping function

This function is used to group entities in iterations according to the values of every selected feature in the best alternative. The groups of similar entities generated from the highest uniqueness measured feature in the best alternative are passed to the next iteration to be grouped by the next selected feature and so on until the lowest uniqueness measured selected feature is in the best alternative. The final blocks of similar entities are produced after pruning all true negatives and false positives entities through iterations. Following is algorithm 4 to define blocks of similar entities.

```
Input: list< featuresOf TheBestAlternative>
       db: datasetOfTheBestAlternative

Output: blocksOfSimilarEntities

(1) list initialBlocks← datasetOfTheBestAlternative;
(2)  FSize← featuresOf TheBestAlternative.size();
(3)       For (int i = 0; i < FSize; i++) do{
(4)            list initiallBlocks←group (initiallBlocks based on fi
               values);
(5)             For (int k = 0; k < initialBlocks.size(); k++) do{
(6)                  If (initiallBlocks[k].size() ==1)
(7)                    initiallBlocks. remove (initialBlocks1[k]);
(8)                    else
(9)                       Continue; }
(10)            return initialBlocks ;}
(11) return list finalBlocksOfSimilarEntities←initialBlocks;
```

Algorithm 4: Defining blocks of similar entities

In more detail, the defining blocks of similar entities algorithm proceeds as follows. We take as inputs features of the best alternative and their data set. The output is the blocks of similar entities. Line 1, initializes blocks with the best alternative data set. We iterate on the best alternative features and form groups of entities according to each feature value of the best alternative and that is presented in line 3 and line 4. Line 7, removes the group of one entity and the groups with more than one entity are filtered using the remaining best alternative features. The final blocks of similar

entities are returned in line 11.

Using the Covid19 example, defining blocks of similar entities algorithm, the best alternative presented in Table 4, we find similar entities that satisfy the determined accuracy threshold.

Table 5: Dataset for the best alternative

DATE_BIRTH	VAX_DATE	NAME	Entity
6/19/88	4/28/21	AhmedTarek	e_1
2/1/48	4/30/21	KhaledAli	e_2
5/6/98	NULL	TarekNabil	e_3
2/2/73	5/23/21	WardAli	e_4
4/3/77	5/22/21	NaderMohamed	e_5
5/20/61	5/26/21	SaraTarek	e_6
11/23/84	6/2/21	ReemSamir	e_3
4/6/79	6/4/21	MarwanAnowar	e_8
12/15/88	4/28/21	AhmedTarek	e_9
5/20/61	5/26/21	SaraTarek	e_{10}
4/6/79	6/4/21	MarwanAnowar	e_{11}
11/22/87	NULL	TarekNabil	e_{12}

Table 6: Entities are grouped according to the DATE_BIRTH feature

DATE_BIRTH	VAX_DATE	NAME	Entity
2/1/48	4/30/21	KhaledAli	e_2
5/20/61	5/26/21	SaraTarek	e_6
5/20/61	5/26/21	SaraTarek	e_{10}
2/2/73	5/23/21	WardAli	e_4
4/3/77	5/22/21	NaderMohamed	e_5
4/6/79	6/4/21	MarwanAnowar	e_8
4/6/79	6/4/21	MarwanAnowar	e_{11}
5/6/98	NULL	ReemSamir	e_3
11/22/87	NULL	TarekNabil	e_{12}
12/15/88	4/28/21	AhmedTarek	e_9
6/19/88	4/28/21	AhmedTarek	e_1

Table 7: Initial blocks1 for similar entities

DATE_BIRTH	VAX_DATE	NAME	Entity	Block
2/1/48	4/30/21	KhaledAli	e_2	
5/20/61	5/26/21	SaraTarek	e_6	IB_1
5/20/61	5/26/21	SaraTarek	e_{10}	
2/2/73	5/23/21	WardAli	e_4	
4/3/77	5/22/21	NaderMohamed	e_5	
4/6/79	6/4/21	MarwanAnowar	e_8	IB_2
4/6/79	6/4/21	MarwanAnowar	e_{11}	
5/6/98	NULL	ReemSamir	e_3	
11/22/87	NULL	TarekNabil	e_{12}	
12/15/88	4/28/21	AhmedTarek	e_9	
6/19/88	4/28/21	AhmedTarek	e_1	

From Table 7, the set of not matched entities is $\{e_2, e_4, e_5, e_3, e_{12}, e_9, e_1\}$, initial blocks list = $\{\{e_6, e_{10}\}, \{e_8, e_{11}\}\}$

Table 8: Entities are grouped according to the VAX_DATE feature

DATE_BIRTH	VAX_DATE	NAME	Entity
5/20/61	5/26/21	SaraTarek	e_6
5/20/61	5/26/21	SaraTarek	e_{10}
4/6/79	6/4/21	MarwanAnowar	e_8
4/6/79	6/4/21	MarwanAnowar	e_{11}

Table 9: Initial blocks2 for similar entities

DATE_BIRTH	VAX_DATE	NAME	Entity	Block
5/20/61	5/26/21	SaraTarek	e_6	IB_1
5/20/61	5/26/21	SaraTarek	e_{10}	
4/6/79	6/4/21	MarwanAnowar	e_8	IB_2
4/6/79	6/4/21	MarwanAnowar	e_{11}	

From Table 9, the set of not matched entities is {e_2, e_4, e_5, e_3, e_{12}, e_9, e_1}, initial blocks list = {{e_6, e_{10}}, {e_8, e_{11}}}

Table 10: Entities are grouped according to the NAME feature

DATE_BIRTH	VAX_DATE	NAME	Entity
5/20/61	5/26/21	SaraTarek	e_6
5/20/61	5/26/21	SaraTarek	e_{10}
4/6/79	6/4/21	MarwanAnowar	e_8
4/6/79	6/4/21	MarwanAnowar	e_{11}

Table 11: Initial blocks3 for similar entities

DATE_BIRTH	VAX_DATE	NAME	Entity	Block
5/20/61	5/26/21	SaraTarek	e_6	IB_1
5/20/61	5/26/21	SaraTarek	e_{10}	
4/6/79	6/4/21	MarwanAnowar	e_8	IB_2
4/6/79	6/4/21	MarwanAnowar	e_{11}	

From Table 11, the set of not matched entities is {e_2, e_4, e_5, e_3, e_{12}, e_9, e_1}, initial blocks list = {{e_6, e_{10}}, {e_8, e_{11}}}

After filtering all entities according to all features of the best alternative, the final blocks of similar entities according to the specified accuracy threshold are {{e_6, e_{10}}, {e_8, e_{11}}}

Table 12: The final blocks of similar entities

Entity Name	Block Name
e_6	B_1
e_{10}	
e_8	B_2
e_{11}	

4. Experiments

We perform many experiments to evaluate the proposed approach in terms of its effectiveness and efficiency versus the literature approaches ICBS (Zhu et al. 2018), SUFFBLOCK (A.Allam et al. 2018), and SAP(A.Allam et al. 2018). ICBS, SUFFBLOCK, and SAP approaches as described in the related work section are some of the most entity-matching recent algorithms using the blocking technique. Experiments are conducted on covid19 data gathered from a variety of sources. All algorithms are implemented using Python 3.9 (64-bit) environment. In addition, all experiments are executed on a laptop with an Intel(R) Core(TM) i5-7300U CPU @ 2.60GHz 2.71 GHz 16.0 GB. The laptop works with Windows 10 Pro. Microsoft SQL Server 2014, and PyCharm Community Edition 2021.2.1. Each time efficiency experiment was run 3 times and the average running time is reported.

4.1 Covid19 Dataset

In our experiment, we use covid19 data collected and integrated from different data sources as input. Kaggle-test data is the first data source, Kaggle-symptoms is the second data source (https://www.kaggle.com/datasets?search=covid19), GitHub-weather data is the third data source, Github-Egypt vaccine is the fourth data source and GitHub-Egypt vaccine type is the fifth data source (https://github.com/owid/covid-19-data/tree/master/public/data). We make some modifications and integration processes on data to be ready to use in our experiment.

Table 13: Covid19 database information

Database Name	Number of Records	Total Number of Features
Covid19	32191	50

4.1.1. Query parameters

There are a set of parameters, ε entity collection which is the data integration result, $\delta = 0.00009$ which is the threshold for the possible number of values for an enumeration feature, $\lambda = 32{,}191$ which is the number of values that are randomly selected from ε (we work on all values of ε), $\delta 1 = 0.9$ which is the threshold for the similarity of the features, $\delta 2 = 0.7$ which is the accuracy threshold of the resulting similar entities obtained by the best alternative, and minLCP =1 and maxAPP = 50. The minLCP determines the minimum number of subsequent starting common words between the suffixes of a block and maxAPP determines the maximum allowed number of appearances of the common subsequences in the entire dataset.

4.1.2. Experimental results

In the evaluation, the proposed approach (TSE-HDR) is evaluated against the improved common block scheme (ICBS), suffix-based blocking (SUFFBLOCK), and (SAB). TSE-HDR gains some advantages over ICBS, SUFFBLOCK, and SAB. Table 14 presents these advantages.

Table 14: Comparison between TSE-HDR, ICBS, SUFFBLOCK, and SAB

		TSE-HDR	ICB	SUFFBLOCK	SAB
Feature Engineering	Type-Based Blocking	√	√	X	X
	Remove Redundant Features	√	√	X	X
	Remove Time State Dependent Features	√	X	X	X
	Evaluate Features Uniqueness Measure	√	X	X	X
	Create The Best Alternative For Selected Features	√	X	X	X
Filtering	Comparison Between Entities Depends on Window Size	X	√	√	X

We use stored procedures to implement feature selection queries that run by the feature selection component. Those stored procedures have the equations used to assess the final feature set for both approaches ICBS, and TSE-HDR, execute according to the scheduled job created by the SQL server, and re-execute as soon as data is updated.

Using the Covid19 dataset, by applying splitting features into different block types algorithms and the attribute clustering algorithm on the ICBS approach, and TSE-HDR feature engineering algorithms and the best alternative of selected features and its uniqueness measure evaluation algorithm on TSE-HDR, The number of selected features, their scaled uniqueness measure, the best alternative, and its scaled uniqueness measure are presented in Tables 15 and 16.

Table 15: TSE-HDR selected features and their uniqueness measure

Feature Name	Feature_Uniqueness_ Measure	Scaled Feature_Uniqueness_ Measure
Name	0.0065	0.492
Vaccination_Date	0.0062	0.470
Vaccination_Type	0.0002	0.015
Gender	0.0001	0.008
Blood_Group	0.0002	0.015

Table 16: The best alternative for TSE-HDR

Alternative Name	Alternative Features	Alternative_Uniqueness_ Measure
The Best Alternative	Name, Vaccination_Date	0.962

From Table 16, the best alternative uses only 2 features Name and Vaccination_Date to determine similar entities.

Number of selected features

In intelligent contexts, feature selection (FS) is critical. It's used to get a set of representative features by evaluating their importance. In our experiment, table 17 presents the comparison between our approach TSE-HDR and ICBS, SUFFBLOCK, and SAP approaches according to the number of used features in determining similar entities.

Table 17: Number of selected features for TSE-HDR versus ICBS, SUFFBLOCK, and SAP

	TSE-HDR	ICBS	SUFFBLOCK	SAP
Number of Selected Shared Features	2	50	50	50

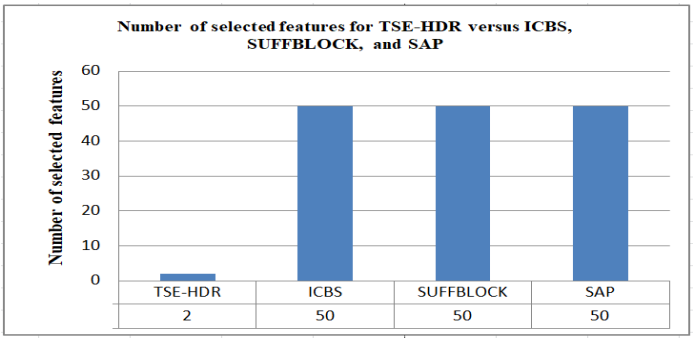

Fig. 3: Number of selected features for TSE-HDR Versus ICBS, SUFFBLOCK, and SAP

Table 17 and Fig. 3; show that the difference between TSE-HDR and ICBS, SUFFBLOCK, and SAP approaches in the number of selected features to specify similar entities is very large, and that reflects the number of processes used.

Sensitivity (SN) (Recall or True Positive Rate)

$$\text{Sensitivity} = \frac{TP}{TP+FN} \tag{5}$$

It is measured using the percentage of the number of true positive predictions to the total number of positives. Recall (REC) or true positive rate (TPR) is another name for sensitivity. The highest sensitivity percentage is 100%, while the lowest percentage is 0%. (Muttakin et al. 2022)

Number of selected features and sensitivity

The number of scaled uniqueness measured features shared in determining similar entities is a direct proportion of sensitivity. Table 18 and Fig. 4 show the difference in sensitivity between our approach TSE-HDR and the ICBS, SUFFBLOCK, and SAP approaches in relation to the number of selected scaled uniqueness measured features. In Table 18 and Fig. 4, 1st feature means the first selected feature presented in Table 15, 1st 2 features mean the first two selected features presented in Table 15, and so on.

Table 18: The difference in sensitivity between the TSE-HDR and ICBS, SUFFBLOCK, and SAP in relation to the number of selected scaled uniqueness measured features

Number of Selected Shared Features	Name of Selected Shared Features	Sensitivity in TSE-HDR	Sensitivity in ICBS	Sensitivity in SUFFBLOCK	Sensitivity in SAP
1st feature	Name	92%	83%	78%	78%
1st 2 features	Name, Vaccination_Date	99.3%	85%	80%	80%
1st 3 features	Name, Vaccination_Date, Vaccination_Type	99.5%	87%	83%	83%
1st 4 features	Name, Vaccination_Date, Vaccination_Type, Gender	99.5%	90%	85%	85%
All 5 features	Name, Vaccination_Date, Vaccination_Type, Gender, Blood_Group	99.8%	92%	90%	90%

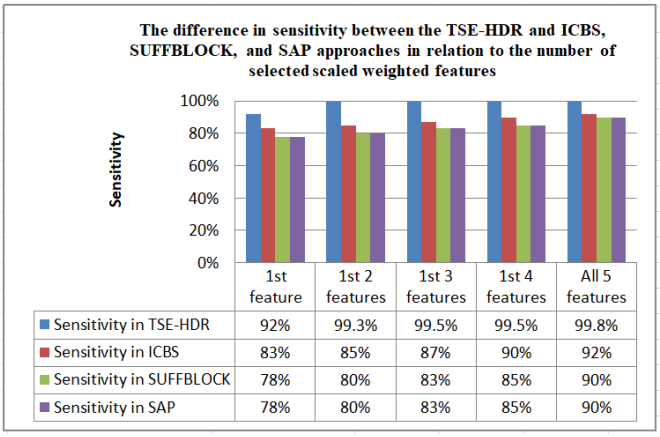

Fig. 4: The difference in sensitivity between the TSE-HDR and ICBS, SUFFBLOCK, and SAP in relation to the number of selected scaled uniqueness measured features

Table 18 and Fig. 4; show that TSE-HDR outperforms ICBS, SUFFBLOCK, and SAP in sensitivity for all used numbers of selected features.

False positive rate

$$FPR = \frac{FP}{FP+TN} \quad (6)$$

It is a result that indicates a given condition exists when it does not. The highest false positive percentage is 0%, while the lowest percentage is 100%.(Muttakin et al. 2022)

Number of selected features and false positive rate

The number of scaled uniqueness measured features shared in determining similar entities is inversely proportional to the false positive rate. Table 19 and Fig. 5 show the difference in false positive rate between our approach TSE-HDR and the ICBS, SUFFBLOCK, and SAP approaches in relation to the number of selected scaled uniqueness measured features. In Table 19 and Fig. 5, 1st feature means the first selected feature presented in Table 15, 1st 2 features mean the first two selected features presented in Table 15, and so on.

Table 19: The difference in false positive rate between the TSE-HDR and ICBS, SUFFBLOCK, and SAP approaches in relation to the number of selected scaled uniqueness measured features

Number of Selected Shared Features	Name of Selected Shared Features	False Positive Rate in TSE-HDR	False Positive Rate in ICBS	False Positive Rate in SUFFBLOCK	False Positive Rate in SAP
1st feature	Name	18%	15%	15%	15%
1st 2 features	Name, Vaccination_Date	7%	10%	14%	14%
1st 3 features	Name, Vaccination_Date, Vaccination_Type	5%	10%	12%	12%
1st 4 features	Name, Vaccination_Date, Vaccination_Type, Gender	4%	10%	11%	11%
All 5 features	Name, Vaccination_Date, Vaccination_Type, Gender, Blood_Group	1%	10%	11%	11%

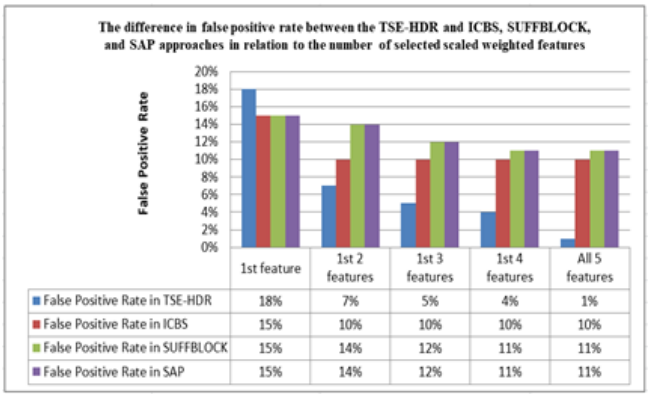

Fig. 5: The difference in false positive rate between the TSE-HDR and ICBS, SUFFBLOCK, and SAP approaches in relation to the number of selected scaled uniqueness measured features

Table 19 and Fig. 5; show that the TSE-HDR false positive rate decreases with a very good rate in relation to the number of selected features while it decreases and then stable in ICBS, and decreases in SUFFBLOCK, and SAP approaches but is still high compared with TSE-HDR and ICBS. In ICBS, SUFFBLOCK, and SAP, the false positive rate varies in relation to the number of selected features. The false positive rate varies also in ICBS and SUFFBLOCK according to specified window size. SUFFBLOCK and SAP approaches are also affected by minLCP and maxAPP. The minLCP determines the minimum number of subsequent starting common words between the suffixes of a block and maxAPP determines the maximum allowed number of appearances of the common subsequences in the entire dataset.

Response time to get similar entities

It is the time between the query submission and receiving complete query answers.

We use stored procedures to implement feature selection queries that run by the feature selection component. Those stored procedures have the equations used to assess the final feature set, execute according to the scheduled job created by the SQL server, and re-execute as soon as data is updated.

Table 20: Response time to get similar entities for TSE-HDR versus ICBS, SUFFBLOCK, and SAP approaches (times are shown in seconds)

	TSE-HDR	ICBS	SUFFBLOCK	SAP
Response Time	0.933	195.3	90.5	400

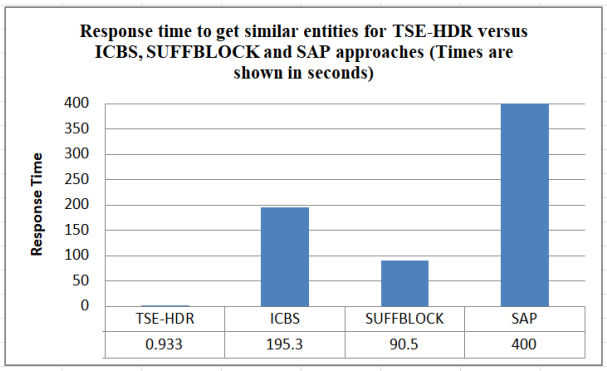

Fig. 6: Response time to get similar entities for TSE-HDR versus ICBS, SUFFBLOCK, and SAP approaches (times are shown in seconds)

Table 20 and Fig. 6; show that TSE-HDR is more efficient to get similar entities than ICBS, SUFFBLOCK, and SAP approaches.

4.1.3. Result discussion

In practice, for an EM system to be successful, it should be easy to apply the EM system to different domains, and there needs to be a good blocking method alongside the EM system. Furthermore, both blocking and EM systems must be easy to deploy.

For big data, the successful EM system should find a solution to a massive amount of features to perform efficiently.

In the evaluation, the proposed approach TSE-HDR is evaluated against the ICBS, SUFFBLOCK, and SAP approaches using four performance metrics, number of selected features, sensitivity, false positive rate, and response time to get similar entities. The experimental evaluation of section 4.1.2 indicates TSE-HDR outperforms ICBS, SUFFBLOCK, and SAB by more than an order of magnitude.

For the number of selected features, TSE-HDR uses 25 times less number of selected features than ICBS, SUFFBLOCK, and SAP to detect similar entities. Fig. 3 and Table 17 show that TSE-HDR uses 2 features to select similar entities where the similarity threshold equals 0.7 while ICBS, SUFFBLOCK, and SAP use 50 features. TSE-HDR feature engineering process put aside time state-dependent features, dependent features, and features with one distinct value and evaluates the scaled uniqueness measure for the remaining features to rank them and choose the best alternative to detect similar entities. ICBS put aside redundant features only to detect similar entities. SUFFBLOCK and SAP use all available features to detect similar entities. Using less number of representative selected features means less number of comparisons and better performance.

For sensitivity or true positive rate, TSE-HDR has better sensitivity for all used numbers of selected features than ICBS, SUFFBLOCK, and SAP as shown in Fig. 4 and Table 18. TSE-HDR sensitivity increases in relation to the number of selected features because it uses a filtering technique in which accuracy for choosing similar entities is increased by adding more features. ICBS and SUFFBLOCK use a dynamic window size to detect similar entities. The sensitivity of ICBS and SUFFBLOCK depends on the chosen window size so some similar entities are missed as a result of the chosen not suitable window size. SUFFBLOCK and SAP also use min_{LCP} and max_{APP} which are changing parameters that affect the sensitivity.

For the false positive rate, ICBS, SUFFBLOCK, and SAP are better than TSE-HDR when using the first selected feature NAME as shown in Fig. 5 and Table 19. ICBS and SUFFBLOCK missing at first fewer correct similar entities than TSE-HDR as the result of using a dynamic window size so it has a better false positive rate using the NAME feature. From using the first two selected features to

the five features in determining similar entities, TSE-HDR has a better false positive rate than ICBS, SUFFBLOCK, and SAP. TSE-HDR accuracy for choosing similar entities is increased by adding more features. The accuracy of ICBS and SUFFBLOCK in choosing similar entities depends on the chosen window size and the number of selected features. The false positive rate for ICBS and SUFFBLOCK varies or stables with changing window size, and the number of selected features. SAP and SUFFBLOCK are also affected by $\min_{LCP}$ and $\max_{APP}$ parameters.

TSE-HDR is an order of magnitude faster than ICBS, SUFFBLOCK, and SAP. TSE-HDR uses only two features to detect similar entities where the similarity threshold equals 0.7 while ICBS, SUFFBLOCK, and SAP use 50 features to detect similar entities. The number of selected features used reflects the number of comparisons, processes, and hence response time. SUFFBLOCK outperforms SAP as SAP requires several queries to the inverted index that stores all suffixes, which suffer from poor locality. Also, SUFFBLOCK is faster than SAB because it uses a dynamic sliding window which is very cache efficient. On the contrary, SAB conducts binary searches to locate blocks of similar suffixes, which suffer from poor locality. Also, SUFFBLOCK avoids iterating over all the prefixes of a certain suffix as SAB does.

From the previously mentioned experiment results, TSE-HDR is an effective and efficient EM system compared to the recent state-of-the-art approaches. It can be applied to different domains, it uses effective blocking and filtering techniques, and it uses a feature engineering component that makes a massive reduction in the number of features. It can be used in different computer science fields like data mining, data integration, data analysis, and information retrieval. It also can be used in other fields and applications like the decennial census (O.Binette and Steorts 2022) which considers one of the important and timely topics, inventor and author disambiguation (O.Binette and Steorts 2022), and estimation of voters in North Carolina (O.Binette and Steorts 2022). TSE-HDR can perform effectively and efficiently in different applications and fields.

5. Conclusion and Future Work

The problem of entity matching or record linkage arises in several contexts and several important applications. A related research area arises with large datasets, which deals with avoiding the quadratic cost comparisons of all record pairs using the specified similarity function. Blocking methods are concerned with minimizing the number of record comparisons without excluding correctly matched records. Since the purpose of blocking is to avoid quadratic cost comparisons, time efficiency is an essential factor of blocking methods. In all blocking techniques, modifying the associated parameters shows a trade-off between excluding more comparisons and including more correctly matched records.

In this paper, The Tracking Similar Entities in Heterogeneous Data Records (TSE-HDR) entity matching approach for the high dimensional, large, distributed, and heterogeneous dataset is presented. It takes as input clean heterogeneous data integration results and finds blocks of similar entities. Entities in the same block refer to the same entity in the same time state and entities in different blocks refer to different entities. TSE-HDR performs using main components, feature engineering, blocking, evaluating the uniqueness measure of the best alternative of selected features, and filtering to find similar entities. The approach is evaluated using a real and large dataset versus the literature ICBS, SUFFBLOCK, and SAP approaches. Experimental results indicated that TSE-HDR outperforms the ICBS, SUFFBLOCK, and SAP approaches. In future work, it's possible to extend our approach to take as input dirty temporal data (data that change its values with time) integration results, matches the same entities in different time states in the present and the absence of the time stamp, detect data anomalies and provide multiple truths, and explain output truths and data fusion decisions for casual users, advanced users, and data fusion administrators.

Funding

Open Access funding enabled and organized by Science Technology & Innovation Funding Authority (STDF) in cooperation with Egyptian Knowledge Bank (EKB)

References

A.Aizawa, and K.Oyama. 2005. "A Fast Linkage Detection Scheme for Multi-Source Information Integration." *WIRI* 30–39. doi: http://dx.doi.org/10.1109/WIRI.2005.2.

A.Allam et al. 2018. "Improved Suffix Blocking for Record Linkage and Entity Resolution." *Data and Knowledge Engineering* 117(March):98–113. doi: https://doi.org/10.1016/j.datak.2018.07.005.

A.Arasu et al. 2006. "Efficient Exact Set-Similarity Joins." *Proceedings of the 32nd International Conference on Very Large Data Bases* 918–29.

A.McCallum et al. 2000. "Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching." *Proceeding of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 169–78. doi: https://doi.org/10.1145/347090.347123.

A.Z.ElQutaany et al. 2010. "A Technique for Mutual Inconsistencies Detection and Resolution in Virtual Data Integration Environment." *Informatics and Systems (INFOS), 2010 The 7th International Conference*.

A.Z.ElQutaany et al. 2018. "V-DIF : Virtual Data Integration Framework." *International Journal of Computer Systems* 05(05):26–32.

Al., A. Dahiya et. 2021. "Data Mining Methods and Techniques for Online Customer Review Analysis: A Literature Review." *Journal of System and Management Sciences* 11(3):1–26. doi: 10.33168/JSMS.2021.0301.

Al., G. Papadakis et. 2013. "A Blocking Framework for Entity Resolution in Highly Heterogeneous Information Spaces." *IEEE Transactions on Knowledge and Data Engineering* 25(12):2665–82. doi: 10.1109/TKDE.2012.150.

Asim, Syed, Ali Shah, Hafiz Muhammad Shabbir, Saif Ur Rehman, and Muhammad Waqas. 2020. "A Comparative Study of Feature Selection Approaches: 2016-2020." *International Journal of Scientific & Engineering Research* 11(February):2016–20.

B.Kenig, and A.Gal. 2013. "MFIBlocks: An Effective Blocking Algorithm for Entity Resolution." *Information Systems* 38(6):908–26. doi: https://doi.org/10.1016/j.is.2012.11.008.

C.W.Cohen et al. 2003. "A Comparison of String Distance Metrics for Matching Names and Records." *Proceedings of the IJCAI-2003 Workshop On* 73–78.

Deng, Dong, Guoliang Li, He Wen, and Jianhua Feng. 2016. "An Efficient Partition Based Method for Exact Set Similarity Joins." *Proceedings of the VLDB Endowment* 9(4):360–71. doi: https://doi.org/10.14778/2856318.2856330.

G.Canalle et al. 2017. "A Strategy for Selecting Relevant Attributes for Entity Resolution Ins Data Integration Systems." *ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems* 1(Iceis):80–88. doi: http://dx.doi.org/10.5220/0006316100800088.

G.Li, and J.Wang. 2011. "Pass-Join: A Partition-Based Method for Similarity Joins." *PVLDB* (2):253–264. doi: https://doi.org/10.48550/arXiv.1111.7171.

Gu, L., and R. Baxter. 2003. "Record Linkage: Current Practice and Future Directions." *Journal of System and Management Sciences* 03/83.

Haque et al., R. 2022. "Geospatial Features Influencing the Formation of COVID-19 Clusters." *Journal of System and Management Sciences* 12(5):1–20. doi: 10.33168/JSMS.2022.0501.

Honest, Nirali. 2020. “A Survey on Feature Selection Techniques.” *GIS SCIENCE JOURNAL* 7(6).

Hussein, A. 2022. “Feature Weighting Based Food Recognition System.” *Journal of Logistics, Informatics and Service Science* 9(3):191–207. doi: 10.33168/LISS.2022.0314.

I.P.Fellegi, and B.Suntib. 1969. “A Theory for Record Linkage.” *Journal of the American Statistical Association* 64(328):1183–1210. doi: https://doi.org/10.1080/01621459.1969.10501049.

J.Chen et al. 2012. “A Learning Method for Entity Matching.” *Proceedings of International Workshop on Quality in Databases, East China Normal University, China.*

J.Qin, and C.Xiao. 2018. “Pigeonring: A Principle for Faster Thresholded Similarity Search.” *Proceedings of the VLDB Endowment* 12(1):28–42. doi: https://doi.org/10.48550/arXiv.1804.01614.

K.Janabi, and K.Rusul. 2018. “Data Reduction Techniques : A Comparative Study for Attribute Selection Methods.” *International Journal of Advanced Computer Science and Technology* 8(1):1–13.

L.Jin et al. 2003. “Efficient Record Linkage in Large Data Sets.” *Proceedings - 8th International Conference on Database Systems for Advanced Applications, DASFAA 2003* 137–46. doi: 10.1109/DASFAA.2003.1192377.

L.Kolb et al. 2012. “Multi-Pass Sorted Neighborhood Blocking with MapReduce.” *Computer Science - Research and Development* 27(1):45–63. doi: https://doi.org/10.1007/s00450-011-0177-x.

M.A. Hernández, S. J. Stolfo. 1995. “The Merge/Purge Problem for Large Databases.” *SIGMOD* 127–138. doi: https://doi.org/10.1145/568271.223807.

M.Bilenko et al. 2006. “Adaptive Blocking: Learning to Scale Up Record Linkage Mikhail.” *ICDM* 29(3):384–473. doi: https://doi.org/10.1109/ICDM.2006.13.

M.Hadjieleftheriou et al. 2009. “Incremental Maintenance of Length Normalized Indexes for Approximate String Matching.” *SIGMOD-PODS’09 - Proceedings of the International Conference on Management of Data and 28th Symposium on Principles of Database Systems* 429–40. doi: https://doi.org/10.1145/1559845.1559891.

Muttakin, Fitriani, Jui Tang Wang, Mulyanto Mulyanto, and Jenq Shiou Leu. 2022. “Evaluation of Feature Selection Methods on Psychosocial Education Data Using Additive Ratio Assessment.” *Electronics (Switzerland)* 11(1). doi: https://doi.org/10.3390/electronics11010114.

N. Polyzotis et al. 2008. “Meshing Streaming Updates with Persistent Data in an Active Data Warehouse.” *Trans. Knowl. Data Eng.* 976–991.

O.Benjelloun et al. 2009. “Swoosh: A Generic Approach to Entity Resolution.” *The VLDB Journal*. doi: https://doi.org/10.1007/s00778-008-0098-x.

O.Binette, and R. Steorts. 2022. “(Almost) All of Entity Resolution.” *Science Advances* 8(12). doi: 10.1126/sciadv.abi8021.

Papadakis, George, Ekaterini Ioannou, Themis Palpanas, Claudia Niederee, and Wolfgang Nejdl. 2013. “A Blocking Framework for Entity Resolution in Highly Heterogeneous Information Spaces.” *IEEE Transactions on Knowledge and Data Engineering* 25(12):2665–82. doi: http://dx.doi.org/10.1109/TKDE.2012.150.

Papadakis, George, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2019. “Blocking and Filtering Techniques for Entity Resolution : A Survey.” *ACM Comput. Surv* 1(1):1–37. doi: https://doi.org/10.1145/3377455.

Q.Tan, and F.Pivot. 2015. “Big Data Privacy: Changing Perception of Privacy.” *Proceedings - 2015*

IEEE International Conference on Smart City, SmartCity 2015, Held Jointly with 8th IEEE International Conference on Social Computing and Networking, SocialCom 2015, 5th IEEE International Conference on Sustainable Computing and Communic (October 2017):860–65. doi: https://doi.org/10.1109/SmartCity.2015.176.

S.Krawczyk et al. 2012. "Citation-Based Bootstrapping for Large-Scale Author Disambiguation." *Journal of the American Society for Information Science and Technology* 64(July):1030–1047. doi: https://doi.org/10.1002/asi.22621.

W.Su et al. 2010. "Record Matching over Query Results from Multiple Web Databases." *IEEE Transactions on Knowledge and Data Engineering* 22(4):578–89. doi: https://doi.org/10.1109/TKDE.2009.90.

Wang, Jiannan, Tim Kraska, Michael J. Franklin, and Jianhua Feng. 2012. "CrowdER: Crowdsourcing Entity Resolution." *Proceedings of the VLDB Endowment* 5(11):1483–94. doi: https://doi.org/10.14778/2350229.2350263.

Z.Amrapali et al. 2018. "CrowdED: Guideline for Optimal Crowdsourcing Experimental Design." Pp. 1109–16 in *The Web Conference*.

Zhu, Hui Juan, Zheng Wei Zhu, Tong Hai Jiang, Li Cheng, Wei Lei Shi, Xi Zhou, Fan Zhao, and Bo Ma. 2018. "A Type-Based Blocking Technique for Efficient Entity Resolution over Large-Scale Data." *Journal of Sensors*. doi: https://doi.org/10.1155/2018/2094696.