La borsa di dottorato è stata cofinanziata con risorse del
Programma Operativo Nazionale Ricerca e Innovazione 2014-2020 (CCI 2014IT16M2OP005)
Fondo Sociale Europeo, Azione I.1 “Dottorati Innovativi con caratterizzazione Industriale”

UNIVERSITA’ DELLA CALABRIA

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

Dottorato di Ricerca in

Tecnologie dell'informazione e della comunicazione
Con il contributo di

PON R&I 2014-2020

XXXVII CICLO

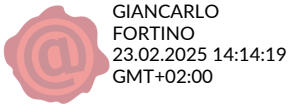
TITOLO TESI

A Sentence Transformer Based Technique for Entity Resolution within Incomplete Data

Settore Scientifico Disciplinare **IINF-05/A**

Coordinatore: Ch.mo Prof. Giancarlo Fortino

Firma



Supervisore/Tutor: Ch.mo Prof. Sergio Greco

Firma

Co-tutor: Dott. Gianvincenzo Alfano

Firma

Dottorando: Dott. Tariq Mahmood

Firma

to my family,
for their love...

Contents

Introduction

Databases are organized collections of data stored electronically, designed to efficiently manage, retrieve, and update information [74, 6, 56]. Missing or incomplete data refers to the absence of values in a database, which can arise from various causes such as human or machine errors, malfunction of equipments, non-response in surveys, or data corruption during transmission or storage [28, 29, 102].

Despite their robust capabilities and widespread use, databases face challenges that can undermine their effectiveness, with managing missing values being one of the most significant issues. This is particularly evident in real-world datasets, where missing information can significantly impact the quality and reliability of statistical analyses and machine learning models, introducing bias, reducing statistical power, and compromising the interpretability of results [69, 160].

Data imputation is the process of replacing missing data with estimated values. It is essential for preserving dataset integrity, enabling full data usage, and improving model performance by reducing distortions from incomplete information [128, 84, 41, 136, 129, 15, 125, 16]. Clearly, the higher the imputation accuracy the better the data quality. These considerations are more evident in safety-critical applications (e.g., healthcare), where missing values can hinder decision-making and lead to inaccurate predictions. Existing data imputation approaches face challenges such as high computational complexity, dependency on specific datasets, and difficulty in imputing rare values, highlighting the need for more advanced techniques. To fill this gap, we propose a novel and innovative data imputation technique called *SENtence Transformer-based Imputation* (SENT-I), that combines advanced indexing techniques to perform quick and accurate similarity searches (called *vector databases*) with sentence transformer models [115]. Sentence Transformers (ST) emerged as a specialized extension of the transformer architecture, particularly in its encoder part, designed to generate fixed-size embeddings that can represent the semantic meaning of entire sentence, thus making them particularly suitable for sentence-level tasks including semantic similarity, clustering, and search [140]. SENT-I starts with converting the (possibly large, and textual) input incomplete dataset into high-dimensional embeddings (according to a sentence transformer model), and storing them in a vector database. Then, missing values in tu-

	Name	Surname	BirthDate	State	Salary	A.Code
t_1	$\perp$	Deere	Second Jan	Un. State	15k	100
t_2	Allisa	Jia	10th Jan	America	$\perp$	400
t_3	David	$\perp$	12th Feb	France	10000	300
t_4	John	Iser	$\perp$	Austria	9000	500
t_5	David	Rao	February 12th	France	10k	300
t_6	Alex	Deere	2nd Jan	$\perp$	15k	$\perp$

	Name	Surname	BirthDate	State	Salary	A.Code
t_1	Alex	Deere	Second Jan	Un. State	15k	100
t_2	Allisa	Jia	10th Jan	America	15k	400
t_3	David	Rao	12th Feb	France	10000	300
t_4	John	Iser	10th Jan	Austria	9000	500
t_5	David	Rao	February 12th	France	10k	300
t_6	Alex	Deere	2nd Jan	Un. State	15k	100

Fig. 1: An example of an incomplete database (top), where null values have been denoted with the symbol $\perp$, and one of its possible imputed versions (bottom).

ples are imputed with values obtained from most (semantically) similar tuples. The experiments revealed that our technique outperforms the state-of-the-art approach up to 40% in imputation accuracy. Performing an initial step of data imputation, under the assumption that the quality of the imputation is high, can significantly impact other well-known problems in database theory, such as *Entity Resolution*, *Data Query and Exchange*, and *Fraudulent Transaction Detection*. Entity Resolution (ER) also known as record linkage or data deduplication, is the process of identification and establishment of links between tuples (belonging to possibly different relations) that correspond to the same real-world entity (e.g., same book, same person, etc.) [38, 104, 149, 78]. ER must be precise and efficient to ensure matched pair generation. The presence of missing data impedes the ER process and adversely affects quality and efficiency. Typically, most ER systems exhibit lower accuracy in the presence of missing data. To overcome this limitation, we propose *Sentence transformer-based solver for Entity Resolution*, (SolvER), a hybrid algorithm that combines SENT-I with any existing ER system. SolvER, before calling an existing ER solver on the input incomplete database, uses SENT-I to convert it into a complete database (i.e., imputing it).

Another well-known problem where the presence of missing information could lead to a decrease in performance is the *Data Query and Exchange*, which consists of transferring data from one database to another (potentially with a different schema), respectively called source and target schema. Within the transfer process, several missing information is typically produced. Thus, a data imputation step could be invoked to ensure the integrity and completeness of the target databases, so that queries could be more precise and reliable.

Financial databases usually contain large amounts of data, in which the possibility of missing values is extremely high [100, 40]. The presence of missing data leads to a decrease in the performance of *Fraudulent Transaction Detection* sys-

tems [130, 118], which identify unauthorized or suspicious transactions in financial systems by analyzing patterns and anomalies in transactional data. It is anticipated that effective data imputation techniques could help restore dataset completeness and potentially enhance the performance of *Fraudulent Transaction Detection* systems.

Plan of the Thesis.

This thesis consists of six chapters, that have been organized as follows.[1] Chapter 1 provides a brief introduction to relational databases (Section 1.1) and incomplete databases (Section 1.2). Section 1.3 focuses on data imputation, discussing existing approaches using machine learning, deep learning, and graph-based techniques. Entity resolution, including recent advancements and techniques in the field are discussed in Section 1.4. Finally, Section 1.5 explores machine learning architectures (graph neural networks and transformers). In Chapter 2 we first discuss the state-of-the-art data imputation approach (Section 2.1), and then present our approach SENTI (Section 2.2). The results of the experimental analysis are discussed in Section 2.3. In Chapter 3 the state-of-the-art entity resolution approach is discussed (Section 3.1) before presenting our approach SolvER (Section 3.2). In Section 3.3 we discuss the results of the experimental analysis. Chapter 4 starts with a discussion of the data query and exchange problem (Section 4.1) followed by the introduction of the proposed approach (Section 4.2). In Section 4.3 we present the complexity analysis and results. Chapter 5 consists of a brief discussion of the fraudulent transaction detection problem (Section 5.1), and a novel technique for solving the problem (Section 5.3). Finally, the conclusion of the thesis including potential directions for future work has been outlined in Chapter 6.

[1] Code and data have been made available online (link).

1

Preliminaries

1.1 Relational Databases

A database model represents a theoretical framework or specification that outlines the structure and usage of a database. It describes the organization of data structures, constrains datasets within those structures, and defines how the data is manipulated. The *relational model* is the most commonly used database model among different types of database models (e.g. hierarchical, network-based, object oriented, graph-based). The relational model was introduced by E. F. Codd in 1970 (see [22]), aimed to make Database Management Systems (DBMSs) independent of specific applications.

Relational Model

We assume the presence of the following pairwise disjoint sets: a countably infinite set *Consts* of *constants*, a countably infinite set $\mathcal{A}$ of *attributes*, and a countably infinite set $\mathcal{R}$ of *relation names*, disjoint from the previous ones. *Consts* is also called *database domain*. Each attribute $A_i \in \mathcal{A}$ is associated with a set of constants called *attribute domain* and denoted as *dom*(A_i). Each relation name $R \in \mathcal{R}$ is associated with a finite sequence of attributes $A_1, \ldots, A_m$, where $m = ar(R)$ is the *arity* of R, which is a non-negative integer. Each attribute A_j is associated with a domain $dom(A_j) \subseteq$ *Consts*.

We say that $R(A_1, \ldots, A_m)$ is a *relation schema*; such a relation schema will also be referred to as $R(U)$, where $U = \{A_1, \ldots, A_m\}$. A *relation* (instance) r over $R(A_1, \ldots, A_m)$ is a finite subset of *dom*$(A_1) \times \cdots \times$ *dom*(A_m). We also say that r is a relation of R. Each element t of r is called a *tuple*. A *position* is an expression of the form $t[A_i]$ or simply $t[i]$ (resp., $r[A_i]$) used to denote the A_i-component of t (resp., the A_i column of r). Likewise, for a set of attributes $X \subseteq \{A_1, \ldots, A_m\}$, $t[X]$ (resp., $r[X]$) denotes the restriction of t (resp., r) to X. A *database schema* is a non-empty finite set $\mathbf{R} = \{R_1(U_1), \ldots, R_m(U_m)\}$ of relation schemas. A *database instance* (or simply database) D over $\mathbf{R}$ is a finite set of relations $\{r_1, \ldots, r_m\}$, where

Symbol	Meaning
Consts	(countably infinite) set of *constants*
A or A_i	specific attribute
$X \subseteq \{A_1 \ldots, A_m\}$	subset of attributes
$\mathcal{A}$	(countably infinite) set of *attributes*
$\mathcal{R}$	(countably infinite) set of *relation names*
R or R_i	specific relation name
$R(A_1, \ldots, A_m)$ or $R(U)$	specific relation schema
r	relation over $R(A_1, \ldots, A_m)$, briefly a relation of R
t or t_i	tuple of r or s
$t[A_i]$	A_i-component of t
$t[X]$	restriction of t over attributes in X
$r[A_i]$	A_i-column of r
$r[X]$	restriction of r over attributes in X
$\mathbf{R} = \{R_1(U_1), \ldots, R_m(U_m)\}$	*database schema*
$D = \{r_1, \ldots, r_m\}$	*database instance* (or simply database) over $\mathbf{R}$
Q	query
$\mathcal{I}$	incomplete database
$\eta/\perp$	labelled/unlabelled null
v	valuation
$v(t)$, $v(R)$, $v(D)$	valuation over incomplete tuple, relation or database
$pw(\mathcal{I})$	set of all possible worlds of $\mathcal{I}$
$\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t \rangle$	data exchange setting
S/T	generic or source schema in data exchange setting
T	target schema in data exchange setting
I, J	generic database instances
ρ	A tuple-generating dependency (TGD)
ζ	An equality-generating dependency (EGD)
Σ_{st}	finite set of TGDs
Σ_t	finite set of TGDs and EGDs over $\mathbf{T}$

Table 1.1: Glossary of (database theory) terms and symbols used through the thesis.

each r_i is a relation over $R_i(U_i)$. To denote a tuple t belonging to a relation r_i we will use the notation $r_i(t)$ and call $r_i(t)$ a (ground) *atom*, or simply a *fact*; hence a database can be viewed as a finite set of (ground) atoms, or facts.

Functional dependencies, keys and foreign keys.

Data dependencies or *integrity constraints* express semantic information about data, i.e. relationships that should hold among data. To give an example, consider a database schema consisting of the following two relation schemas:
$Employee(E\#, EName, Proj)$ and $Project(P\#, PName)$. In this case, we may want to enforce the condition that no two distinct tuples in any relation based on the first relation schema can have the same id (i.e. the same value on $E\#$). Likewise, every relation over the second relation schema cannot contain two different tuples with

the same id (i.e. the same value on P#). It would be sensible also to insist that every department id which occurs in the employee relation also occurs in the department relation. The first two constraints mentioned above are examples of *key constraints* – we say that attribute E# is a key of $Employee$ and P# is a key of $Project$. The last constraint mentioned above is an example of *foreign key constraint* – we say that attribute $Proj$ of $Employee$ is a foreign key (referring to attribute P# of $Project$). Key and foreign key constraints are special types of *functional* and *inclusion dependencies*.

Example 1.1. Consider the following database:

E#	Ename	Proj
e1	John	d1
e2	Peter	d2
e3	Craig	d3

P#	Pname
d1	Physics
d2	Chemistry

Attribute E# is a key of the first relation, whereas attribute P# is a key of the second one (this is illustrated by underlining E# and P#). The foreign key constraint stating that each department appearing in the employee relation must appear in the department relation can be denoted with the expression $Employee[Proj] \subseteq Project[P\#]$. The above database satisfies the key constraints, it is easy to see that the foreign key constraint is not satisfied because there is no tuple in the department relation having $d3$ as P#-value.

Given a relation schema $R(U)$, a *functional dependency* fd over $R(U)$ is an expression of the form $X \rightarrow Y$, where $X, Y \subseteq U$. If Y is a single attribute, then fd is said to be in *standard form* whereas if $Y \subseteq X$, then the functional dependency is *trivial*. A relation r over $R(U)$ *satisfies* fd, denoted as $r \models fd$, iff $\forall t_1, t_2 \in r,\ t_1[X] = t_2[X]$ implies $t_1[Y] = t_2[Y]$ (we also say that r is *consistent* with respect to fd). Moreover, r satisfies (or is consistent w.r.t.) a set FD of functional dependencies over $R(U)$, denoted as $r \models FD$, iff r satisfies every functional dependency in FD. We say that FD *logically implies* a functional dependency fd, denoted $FD \models fd$, iff for every relation r over $R(U)$, if r satisfies FD, then r satisfies fd. A *superkey dependency* is a functional dependency of the form $X \rightarrow U$. Given a set FD of functional dependencies, a *key* of R is a minimal (under set inclusion) set K of attributes of R such that FD logically implies $K \rightarrow U$. Each attribute in K is called *key attribute*. A *primary key* of R is a designated key of R. Given two relation schemas $R(U)$ and $S(V)$, a *foreign key constraint* fk is an expression of the form $R(W) \subseteq S(Z)$, where $W \subseteq U, Z \subseteq V, |W| = |Z|$ and Z is a key of S (if Z is the primary key of S we call fk a *primary foreign key constraint*). Two relations r and s over $R(U)$ and $S(V)$, respectively, *satisfy* fk iff for each tuple $t_1 \in r$ there is a tuple $t_2 \in s$ such that $t_1[W] = t_2[Z]$ (we also say that r and s are *consistent* with respect to fk).

1.2 Incomplete Databases

The problem of incomplete data has been studied since the development of the relational data model. A database is said to be incomplete if some of its attribute values are missing. Whenever a value is missing we also say that is is a *null value*. Semantically, an *incomplete database* represents a collection of complete databases, often referred to as *possible worlds*. Thus, instead of completely specifying one state of the world, an incomplete database describes a range of alternative potential states. Different formalisms, known as *representation systems*, have been developed to represent incomplete databases. The focus is primarily on representation systems that utilize *null values*, though alternative methods not relying on null values have been proposed for representing incomplete databases. Over the years, different types of null values have been explored: an *unknown* null indicates that a value exists but is currently unknown, an *inapplicable* (or *non-existing*) null specifies that a value does not exist, a *no-information* null indicates uncertainty about whether a value exists at all.

We use $\mathcal{I}$ and $\mathcal{J}$ to denote incomplete database instances (also called incomplete databases or incomplete instances). Moreover, for any incomplete database $\mathcal{I}$ (represented according to a given representation system RS), we use $pw(\mathcal{I})$ to denote the set of possible worlds D of $\mathcal{I}$.

Definition 1.2. *Given a query Q and an incomplete database $\mathcal{I}$ (represented according to a given representation system RS), the result of evaluating Q over $\mathcal{I}$ is $Q(\mathcal{I}) = \{Q(D) \mid D \in pw(\mathcal{I})\}$.*

Thus, $Q(\mathcal{I})$ includes a set of query answers corresponding to each possible world of $\mathcal{I}$. However, certain tuples may be answers to Q regardless of which possible world represents the true state of the real world. Conversely, there may be tuples that serve as answers to Q in relation to some possible worlds, but not necessarily in all of them. These concepts give rise to the notions of *certain* and *possible* query answers.

A tuple is considered a certain answer to a query Q with respect to an incomplete database $\mathcal{I}$ if it satisfies Q in every possible world represented by $\mathcal{I}$.

Definition 1.3. *The set of* certain answers *to a query Q with respect to an incomplete database $\mathcal{I}$ is defined as follows:*

$$certain(Q,\mathcal{I}) = \bigcap_{D \in pw(\mathcal{I})} Q(D)$$

A tuple is a *possible* answer to Q with respect to $\mathcal{I}$ if it is an answer to Q in some possible world of $\mathcal{I}$.

Definition 1.4. *The set of* possible answers *to a query Q with respect to an incomplete database $\mathcal{I}$ is defined as follows:*

$$possible(Q,\mathcal{I}) = \bigcup_{D \in pw(\mathcal{I})} Q(D)$$

For any incomplete relation r, *certain(r)* denotes the set of tuples in r not containing null values. Moreover, for an incomplete database $\mathcal{I}$, $certain(\mathcal{I}) = \{certain(r) \mid r \in \mathcal{I}\}$.

Representation Systems

To represent incomplete databases effectively, various *representation systems* have been proposed. These systems focus on compactly encoding the information within databases where some values are missing or unknown. Representation systems that extend the relational model introduce labeled nulls into tuples, enhancing the model's ability to handle incomplete information. These labeled nulls act as placeholders, representing unknown values while maintaining their distinct identities. The key idea behind these systems is that the set of possible worlds (or complete databases) is defined as the collection of all databases that can be obtained by substituting the labeled nulls with constants. Representation systems for incomplete databases are often classified as *strong* and *weak*, and give a picture of how the different representation systems behave with respect to such properties.

We start by introducing some notation and terminology that will be used in the following. We assume the existence of a countably infinite set $\mathit{Nulls} = \{\eta_i \mid i \in \mathbb{N}\}$ of *labeled nulls*.

Recall that *Consts* denotes the database domain. A *valuation* is a mapping $v : \mathit{Nulls} \cup \mathit{Consts} \to \mathit{Consts}$, such that $v(c) = c$ for every $c \in \mathit{Consts}$. Hence, a valuation maps every constant to itself and every labeled null to either a constant or a different labelled null. The result of applying a valuation v over a tuple t (resp. relation r, database D) possibly containing labeled nulls is denoted by $v(t)$ (resp. $v(R)$, $v(D)$) and is defined in the natural fashion (e.g. $v(t)$ is the tuple obtained from t by replacing every occurrence of a labeled null η_i with $v(\eta_i)$).

Given a query Q, we would like to be able to find a representation of the answers to Q over the incomplete database $\mathcal{I}$, using the same representation system used for $\mathcal{I}$. More precisely, for each query Q and incomplete database $\mathcal{I}$, we would like to compute an incomplete database $\mathcal{J}$ (from $\mathcal{I}$ and Q) such that $pw(\mathcal{J}) = Q(pw(\mathcal{I}))$. If a representation system has this property for a query language $\mathcal{L}$ then it is said to be a *strong* representation system for $\mathcal{L}$. The following commutative diagram illustrates this point.

$$\begin{array}{ccc} \mathcal{I} & \xrightarrow{pw} & pw(\mathcal{I}) \\ {\scriptstyle Q}\downarrow & & \downarrow{\scriptstyle Q} \\ \mathcal{J} & \xrightarrow[pw]{} & Q(pw(\mathcal{I})) \end{array}$$

We now present the notion of a *weak* representation system by relaxing the requirements of a strong representation system.

Given a query language $\mathcal{L}$, we say that two incomplete databases $\mathcal{I}$ and $\mathcal{J}$ are *$\mathcal{L}$-equivalent*, denoted $\mathcal{I} \equiv_{\mathcal{L}} \mathcal{J}$, if $certain(Q, \mathcal{I}) = certain(Q, \mathcal{J})$ for each query

Q of $\mathcal{L}$. A representation system is *weak* for a query language $\mathcal{L}$ if for each incomplete database $\mathcal{I}$ and query Q of $\mathcal{L}$ there exists a incomplete database $\mathcal{J}$ such that $pw(\mathcal{J}) \equiv_{\mathcal{L}} Q(pw(\mathcal{I}))$. As opposed to a strong representation system, a weak representation system is not required to be able to represent $Q(pw(\mathcal{I}))$ for every query Q and incomplete database $\mathcal{I}$; however, a weak representation system must be able to provide an incomplete database $\mathcal{J}$ s.t. $pw(\mathcal{J})$ is $\mathcal{L}$-equivalent to $Q(pw(\mathcal{I}))$ – this means that $pw(\mathcal{J})$ and $Q(pw(\mathcal{I}))$ are indistinguishable as long as we are interested only in the certain answers to queries in $\mathcal{L}$.

In the following, different representation systems are presented. To simplify the presentation, we restrict our discussion to unirelational databases, and assume that every attribute domain coincides with the database domain *Consts*. Generalization is straightforward.

Codd tables.

A *Codd table* is a relation possibly containing labeled nulls from *Nulls*, where each labeled null can occur at most once. The incomplete database represented by a Codd table $\mathcal{I}$ is defined as follows:

$$pw(\mathcal{I}) = \{v(\mathcal{I}) \mid v \text{ is a valuation}\}$$

Thus, the possible worlds represented by $\mathcal{I}$ are the complete databases that can be derived from $\mathcal{I}$ by replacing every labeled null in $\mathcal{I}$ with a constant. It is important to highlight that the previous definition of $pw(\mathcal{I})$ assumes the Closed World Assumption (CWA) because each tuple in a possible world of $pw(\mathcal{I})$ must be derived from a tuple of $\mathcal{I}$. If the Open World Assumption (OWA) is made, then the possible worlds represented by $\mathcal{I}$ include $pw(\mathcal{I})$ and any other complete database that contains a database in $pw(\mathcal{I})$.

Example 1.5. Assume $\eta_1, \eta_2, \eta_3, \eta_4$ are labeled nulls in *Nulls*. The following is a Codd table:

A	B	C
0	1	η_1
η_2	η_3	1
2	0	η_4

The following relations are some of the possible worlds represented by the previous Codd table:

A	B	C
0	1	1
1	1	1
2	0	2

A	B	C
0	1	2
1	2	1
2	0	1

A	B	C
0	1	2
4	2	1
2	0	0

As an example, the first relation above is obtained from the Codd table by means of a valuation v s.t. $v(\eta_1) = 1$, $v(\eta_2) = 1$, $v(\eta_3) = 1$, $v(\eta_4) = 2$. Under the OWA, the following is also one of the possible worlds (because it is a superset of the first possible world reported above):

A	B	C
0	1	1
1	1	1
2	0	2
1	1	0

As illustrated in the following example, Codd tables are not a strong representation system even for very restricted subsets of relational algebra.

Example 1.6. To give an idea of why Codd tables fail to be a strong representation system for different subsets of relational algebra, consider the Codd table of Example 1.5, call it $\mathcal{I}$, and the simple query Q defined as $\sigma_{A=4}(\mathcal{I})$. Clearly, $Q(pw(\mathcal{I}))$ contains an empty relation (this is obtained, for instance, by evaluating Q over the first possible world reported in Example 1.5) and at least one non-empty relation (e.g. the one obtained by evaluating Q over the third possible world reported in Example 1.5). It can be easily verified that there is no Codd table $\mathcal{I}'$ whose possible worlds contain the two aforementioned relations.

Codd Tables form a weak representation system for the subset of relational algebra consisting only of selection (involving equalities and inequalities) and projection. If we consider a language that allows also join or union, then Codd tables are no longer a weak representation system for such a language.

Naive tables.

One of the limitations of Codd tables is that a labeled null can occur at most once. *Naive tables* remove this limitation and are defined like Codd tables except that labeled nulls are allowed to occur more than once.[1]

The set of possible worlds represented by a naive table $\mathcal{I}$ is defined in the same way as done for Codd tables, namely:

$$pw(\mathcal{I}) = \{v(\mathcal{I}) \mid v \text{ is a valuation}\}$$

Notice that if a naive table contains multiple occurrences of the same labeled null, then possible worlds are obtained by replacing the different occurrences of the same labeled null with the same constant.

Example 1.7. Assume η_1, η_2, η_3 are labeled nulls in *Nulls*. The following is a naive table (but not a Codd table because of the two occurrences of η_1):

[1] Naive tables have also been called *V-tables* and *e-tables* [1, 50, 64].

A	B	C
0	1	η_1
η_2	η_3	1
2	0	η_1

Notice that the naive table above says that even if the C-values of the first and third tuples are unknown, we know that they are the same.

The following relations are some of the possible worlds represented by the previous naive table:

A	B	C
0	1	1
1	1	1
2	0	1

A	B	C
0	1	2
1	2	1
2	0	2

A	B	C
0	1	2
3	2	1
2	0	2

As an example, the first relation above is obtained from the naive table by means of a valuation v s.t. $v(\eta_1) = 1$, $v(\eta_2) = 1$, $v(\eta_3) = 1$. We remark again that for each possible world of the previous naive table the C-value of the first tuple is equal to the C-value of the third tuple.

Naive tables are a weak representation system for relational algebra queries using selection (where only equalities are allowed), projection, join and union. For a query Q in this class, the certain answers to Q with respect to the incomplete database represented by a naive table $\mathcal{I}$ can be computed as follows: first, Q is evaluated over $\mathcal{I}$ in the standard way by treating labeled nulls as new constants different from any constant in the database domain; then, tuples in the result containing labeled nulls are discarded and the remaining tuples are the certain answers.

Conditional tables.

So far we have seen that neither Codd nor naive tables are a strong representation system for full relational algebra. We now present a much more powerful representation system, namely *conditional tables*, that forms a strong representation system for relational algebra.

A *condition* is a conjunction of atoms of the form $\eta_i = \eta_j$, $\eta_i = c$, $\eta_i \neq \eta_j$, or $\eta_i \neq c$, where η_i and η_j are labeled nulls and c is a constant. A valuation v *satisfies* a condition ϕ iff by replacing every occurrence of labeled null η_i in ϕ with $v(\eta_i)$, the resulting formula is true. A *conditional table* (*c-table* for short) is a triple $\langle \mathcal{I}, \Phi, \Psi \rangle$ where $\mathcal{I}$ is a naive table, Φ is a condition (called *global condition*) and Ψ is a function mapping every tuple of $\mathcal{I}$ to a condition (conditions associated with tuples via function Ψ are called *local conditions*). Global and local conditions can contain labeled nulls not appearing in $\mathcal{I}$.

The set of possible worlds represented by a conditional table $\langle \mathcal{I}, \Phi, \Psi \rangle$ is defined as follows:

$$pw(\langle \mathcal{I}, \Phi, \Psi \rangle) = \{r \mid \text{there exists a valuation } v \text{ such that } v \text{ satisfies } \Phi \text{ and } r = \{v(t) \mid t \in \mathcal{I} \text{ and } v \text{ satisfies } \Psi(t)\}\}$$

Notice that the previous definition of pw adopts the closed world assumption.

Example 1.8. Suppose we know that Sally is taking math or computer science (CS), but not both, and another course that is not known. Alice takes biology if Sally takes math, and math or physics, but not both, if Sally takes physics. This can be represented by the following c-table:

$\eta_1 \neq math \wedge \eta_1 \neq CS$

Student	Course	
Sally	math	$\eta_2 = 0$
Sally	CS	$\eta_2 \neq 0$
Sally	η_1	
Alice	biology	$\eta_2 = 0$
Alice	math	$\eta_1 = physics \wedge \eta_3 = 0$
Alice	physics	$\eta_1 = physics \wedge \eta_3 \neq 0$

In the previous c-table η_1, η_2, and η_3 are labeled nulls. The global condition Φ is $\eta_1 \neq math \wedge \eta_1 \neq CS$. For each tuple the condition associated by Ψ is reported in the last column (a missing condition for a tuple t means that $\Psi(t) = true$).

The following relations are some of the possible worlds represented by the previous c-table:

Student	Course
Sally	math
Sally	physics
Alice	biology
Alice	math

Student	Course
Sally	math
Sally	biology
Alice	biology

Student	Course
Sally	CS
Sally	biology

As an example, the first relation above is obtained from the c-table by means of a valuation v s.t. $v(\eta_1) = physics, v(\eta_2) = 0, v(\eta_3) = 0$.

A valuation v such that $v(\eta_1) = math$, $v(\eta_2) = 1$ would lead to the following relation:

Student	Course
Sally	CS
Sally	math

However this relation is not a possible world because v does not satisfy the global condition.

C-tables form a strong representation for relational algebra.

Nulls in SQL

In the literature, the problem of nulls has been investigated for a long time, but there is no accepted solution for its use and meaning. Two main approaches have been considered based on using i) a single null value, adopted in the SQL language, and

ii) multiple null values (called labeled nulls). We consider first the approach that SQL adopted and based on using a unique null value. Although manuals and books recommend to avoid or limit nulls, in practical applications such as data integration, data exchange, database repairing, etc., they arise very often. Null values can be introduced not only when databases are updated, but also in the execution of queries as outer join operations introduce null values to complete tuples.

The SQL standard provides one single constant NULL to represent a missing value. Generally, the full behavior of the NULL value in SQL is not described in detail, as the SQL rules surrounding NULL can be ambiguous, often not intuitive, and in some cases surprising. How NULLs should be handled in SQL, in all circumstances, is not clear from the standard documents.

In SQL, NULL indicates that the value is unknown. Notice that a NULL occurrence is different from the value zero, the empty string, and even from others NULL occurrences, that is two occurrences of NULL are not equal. Indeed, any comparison between NULL and any other value – a constant or another NULL – yields the *unknown* truth value because the value of NULL is unknown. Thus, in the presence of NULL, SQL considers a three-valued logic where the truth values are $false$, $unknown$, and $true$. As an example, given two relations $r_1 = \{(a, \text{NULL}), (b, 1), (c, 2)\}$ and $r_2 = \{(a, \text{NULL})\}$ with schemas $R_1(A, B)$ and $R_2(C, D)$, the join of r_1 and r_2 with join condition $B = D$ gives in output an empty relation, while the union of r_1 and r_2 is equal to $\{(a, \text{NULL}), (b, 1), (c, 2)\}$. Moreover, the selection of the tuples of r_1 satisfying the condition $B = 1$ gives the relation $\{(b, 1)\}$, whereas if the condition is $B \neq 1$ we get the relation $\{(c, 2)\}$, as NULL cannot be assumed to be equal to 1, but cannot even be assumed to be different from 1, hence both comparisons yield the truth value *unknown* and the first tuple of r_1 is not included in the result. In fact, only tuples for which the comparison yields *true* are included in the result.

Under the linear ordering $false < unknown < true$ defined over the truth values, the meaning of the logical operators $\wedge$ and $\vee$ does not change as $A \wedge B = min\{A, B\}$ and $A \vee B = max\{A, B\}$; the meaning of the negation operators must be extended assuming that $\neg unknown = unknown$ is true. Moreover, arithmetic operators involving NULL values give as result NULL. Although it is reasonable that NULL $+ 5 =$ NULL, surprisingly NULL $\times 0 =$ NULL. More surprisingly is the fact that in SQL if we count the tuples in the above relation r_1 the result is 3 (`SELECT COUNT(*) FROM R1`), but if we count the tuples in the relation obtained by projecting r_1 over attribute B the result is 2 (`SELECT COUNT(B) FROM R1`) and if we sum the values in the second column of r_1 the result is 3 (`SELECT SUM(B) FROM R1`).

1.3 Data Imputation

Real-world databases are frequently characterized by the occurrence of missing values, that stems from system errors, failure, human mistakes, etc [28, 29, 102], and can affect the quality, interpretability, and reliability of analytical results. Intuitively, the Data Imputation (DI) is a process that transform any incomplete dataset (i.e., a

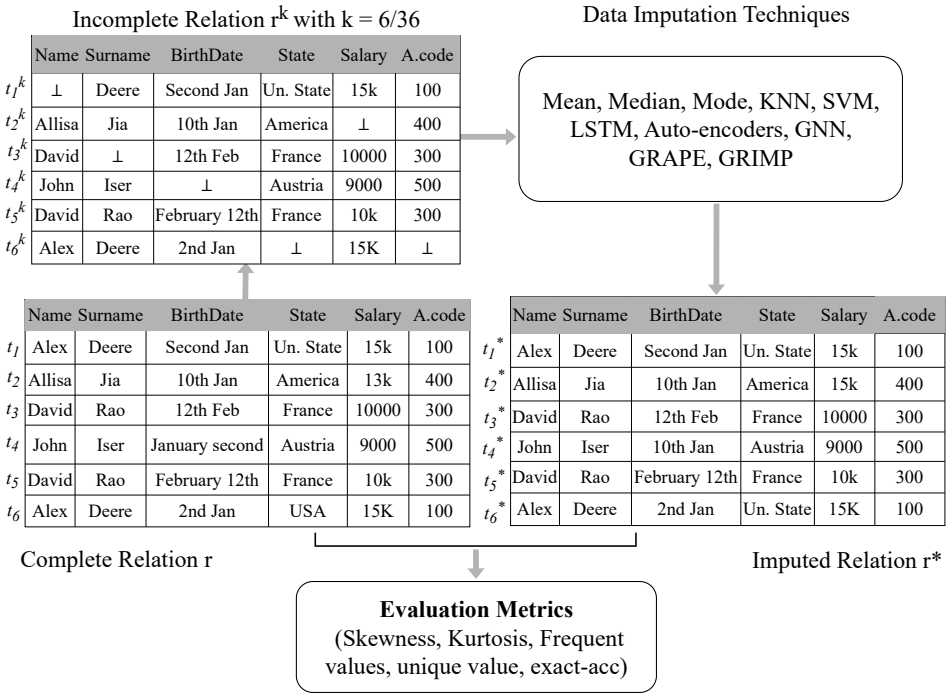


Fig. 1.1: An example of the data imputation problem.

dataset containing missing values), into a complete dataset. The goal of any imputation algorithm is to (i) detect cells containing missing values and (ii) replace them with some 'concrete' values. Numeric data imputation is generally easier than textual data imputation, as text data involves semantics, context, and structure, requiring sophisticated models like transformer-based approaches (e.g., BERT) to generate meaningful imputations.

Definition 1.9 (General similarity between relations). Given a relation $r = \{t_1, ..., t_n\}$ defined over $R(A_1, ..., A_m)$ and a relation $r^* = \{t_1^*, ..., t_n^*\}$ obtained from r by replacing some attribute values $t_i[A_j]$ with a value in $dom(A_j)$, the similarity between r and r^* is defined as $\delta(r, r^*) = \frac{1}{n}\sum_{i=1}^{n}\delta(t_i, t_i^*)$.

For any r and rational number $k \in [0, 100]$, r^k denotes a relation derived from r by replacing uniformly at random $k\%$ of its $(n \times m)$ values with the null value $\perp$.

Example 1.10. Considering the relation $r = \{t_1, \ldots, t_6\}$ shown in Figure 1.1 (bottom-left) where A_1=Name, A_2=Surname, A_3=BirthDate, A_4=State, A_5=Salary, and A_6= A.Code. The number of positions in r is $n \times m = 6 \times 6 = 36$. An example of relation r^k, with $k = \frac{6\times100}{36}$, obtained from r by replacing 6 of the 36 values with nulls is shown in Figure 1.1 (top-left). □

Given an incomplete relation $r^k = \{t_1, ..., t_n\}$, a data imputation function $\gamma : i \times j \rightarrow dom(A_j)$ such that $\gamma(i, j) = t_i[A_j]$ if $t_i[A_j]$ is not null, otherwise $\gamma(i, j) \in dom(A_j)$. Thus, γ is a function replacing nulls with constants.

We next state the problem formally.

Definition 1.11 (Data Imputation Problem). Let r be a complete relation, k a natural number in $[0, 100]$, δ a similarity function, τ a real number in $[0, 1]$, and r^k an incomplete relation obtained from r. Data Imputation is the problem of determining a function γ such that $\delta(r, r^* = \gamma(r^k)) \geq \tau$.

The threshold τ should be chosen based on the application's tolerance for imputation error and the type of data. For sensitive applications, a higher threshold is preferred to ensure high imputation accuracy, while for less critical applications, a lower threshold may be acceptable. An inappropriate choice of τ can either leave too much data unfilled (if τ is too high) or lead to inaccurate imputations (if τ is too low). In most of the ML approaches γ is learned from the (training) data. It is worth noting that, as γ is learned, the threshold τ it is only used for checking the feasibility of γ. Observe also that although the data imputation problem has been defined for incomplete relations r^k obtained from a complete relation r, it can be easily extended to incomplete databases $D^k = \{r_1^k, ..., r_i^k\}$ that is a set of incomplete relations $r_1^k, ..., r_i^k$ obtained from complete database $D = \{r_1, ..., r_i\}$.

Regarding the similarity function δ, it should be chosen based on the nature of the data (i.e., categorical/numerical). As it will be discussed in Chapter 2 in more detail, we use the cosine similarity between the embedded relations r and s to effectively capturing semantic similarity of the two relations. So that, the accuracy of data imputation is insensible with respect to the type of data, as both numerical and categorical data are represented in the latent space through embeddings.

Furthermore, the results of any data imputation system might depend on the specific function δ used. As it will be discussed in Example 2.2, imputation based on exact similarity might fail in accurately imputing data, whereas using semantic similarity does not.

We next discuss some basic, standard instances of functions δ that are able to evaluate the effectiveness of DI, and postpone the presentation of the more advanced technique based on the semantic similarity in the next chapter.

Evaluation Metrics

To evaluate the effectiveness of the imputation, i.e. how close r^* and r are, several metrics for emulating the computation of the similarity between r^* and r have been proposed so far, also capturing both statistical properties and accuracy.

Exact Accuracy.

Exact Accuracy is a metric used to evaluate how well an imputation method reconstructs missing values by comparing the imputed dataset with the original complete dataset. To better understand, let $r = \{t_1, ..., t_n\}$ is a complete relation defined

over the schema $R(A_1, ..., A_m)$, r^k is an incomplete relation obtained from r and $r^* = \gamma(r^k) = \{t_1^*, ..., t_n^*\}$ is an imputed relation obtained from r^k,

$$\textsf{exact-acc}(r, r^k, r^*) = \frac{1}{k \times (n \times m)} \sum_{i=1}^{n} \sum_{j=1}^{m} ord(t_i[A_j] = t_i^*[A_j])$$

where ord is the standard ordinal function.

Example 1.12. Considering the scenario of Figure 1.1, we have that $\textsf{exact-acc}(r, r^k, r^*) = \frac{3}{6} = 0.5$. □

The exact accuracy metric is strict because it requires an exact character-by-character match between the original and imputed values. This is problematic for categorical data, where different representations of the same entity (e.g., USA vs. United States) are semantically equivalent but would still be considered incorrect under exact accuracy.

However, expecting an exact match between r and r^* is unrealistic and has limited practical value. Because of this, we also rely on other statistical tests, i.e., skewness, kurtosis, frequent and unique values to emulate the computation of the similarity between r^* and r.

Average Skewness.

Skewness is a statistical measure that represents the asymmetry or deviation from the symmetry of a dataset's distribution. It indicates whether the data points are more concentrated on one side of the mean (average) value. Clearly, skewness makes sense for numerical attributes only.

The formula for computing the average skewness (denoted as $\textsf{avg-skew}(r)$) across all columns in r is typically written as follows:[32, 126]:

$$\textsf{avg-skew}(r) = \frac{1}{m} \sum_{i=1}^{m} \textsf{skew}(r[A_i])$$

The skewness of r w.r.t. an attribute A_i, $\textsf{skew}(r[A_i])$ is computed as:

$$\textsf{skew}(r[A_i]) = \frac{\frac{1}{n} \sum_{j=1}^{n} (t_j[A_i] - \mu)^3}{\left(\frac{1}{n} \sum_{j=1}^{n} (t_j[A_i] - \mu)^2\right)^{3/2}}$$

where $n = |r[A_i]|$ is the number of tuples in $r[A_i]$, and μ is the mean of these values, defined as follows:

$$\mu = \frac{1}{n} \sum_{i=1}^{n} t_i[A_i].$$

Thus, the distance between r and r^* according to the average skeweness can be computed as $\delta(r, r^*) = |\textsf{avg-skew}(r) - \textsf{avg-skew}(r^*)|$. Clearly, whenever r and r^* coincide (i.e., whenever the imputation is correct) we have that $\delta(r, r^*) = 0$.

Average Kurtosis.

Kurtosis is another statistical measure used for the data distribution in terms of tailedness of the distribution. Mathematically, one can represent average kurtosis denoted as (avg-kurt(r)) across all columns in r of a given dataset with the following formula [11]:

$$\mathsf{avg\text{-}kurt}(r) = \frac{1}{m}\sum_{i=1}^{m}\mathsf{kurt}(r[A_i])$$

The kurtosis of r w.r.t. an attribute A_i, $\mathsf{kurt}(r[A_i])$ is computed as:

$$\mathsf{kurt}(r[A_i]) = \frac{\frac{1}{n}\sum_{j=1}^{n}(t_j[A_i]-\mu)^4}{\left(\frac{1}{n}\sum_{j=1}^{n}(t_j[A_i]-\mu)^2\right)^2} - 3$$

Thus, the distance between r and r^* according to the average kurtosis can be computed as $\delta(r, r^*) = |\mathsf{avg\text{-}kurt}(r) - \mathsf{avg\text{-}kurt}(r^*)|$. Clearly, whenever r and r^* coincide (i.e., whenever the imputation is correct) we have that $\delta(r, r^*) = 0$.

Average Number of Frequent Values.

The following metric is based on frequent values. A value is considered frequent if its number of occurrences is larger than the 90% quartile of all occurrence frequencies in the same attribute [16]. Formally,

$$\mathsf{avg\text{-}freq}(r) = \frac{1}{m}\sum_{i=1}^{m}\mathsf{F}(r[A_i])$$

where $\mathsf{F}(r[A_i])$ denotes the number of $r[A_i]$'s rows that contain frequent values. Thus, the distance between r and r^* according to the average frequent values can be computed as $\delta(r, r^*) = |\mathsf{avg\text{-}freq}(r) - \mathsf{avg\text{-}freq}(r^*)|$. Clearly, whenever r and r^* coincide (i.e., whenever the imputation is correct) we have that $\delta(r, r^*) = 0$.

Average Number of Unique Values.

This metric averages the unique values across all columns [16]. A value $t_i[A]$ is said to be unique in column A if there is no $t_j \in r$ s.t. $t_j \neq t_i$ and $t_j[A] = t_i[A]$. Formally,

$$\mathsf{avg\text{-}uni}(r) = \frac{1}{m}\sum_{i=1}^{m}\mathsf{N}(r[A_i])$$

where $\mathsf{N}(r[A_i])$ is the number of unique values in $r[A_i]$. The distance between r and r^* according to the average frequent values can be computed as $\delta(r, r^*) = |\mathsf{avg\text{-}uni}(r) - \mathsf{avg\text{-}uni}(r^*)|$. Clearly, whenever r and r^* coincide (i.e., whenever the imputation is correct) we have that $\delta(r, r^*) = 0$.

Data Imputation Methods		References
Traditional/ statistical	Mean	[33]
	Median	[68]
	Mode	[158]
Machine Learning	K-Nearest Neighbors (KNN)	[71], [21], [70], [162], [88]
	Support Vector Vachine (SVM)	[7], [128]
	Decision Trees (DT)	[139],[61],[95]
	Gaussian Process Regression (GPR)	[63], [117]
	Expectation Maximiumization	[101], [80].
Deep learning	Auto-encoder (AE)	[129], [111], [25], [46], [135], [15]
	Long Short-Term Memory (LSTM)	[94], [83], [156], [84], [41]
	Generative Adversarial Network (GAN)	[125]
	GNN	[16], [132], [154], [47]

Table 1.2: Overview of the different approaches proposed to solve the Data Imputation problem.

Intuitively, avg-freq and avg-uni metrics give an idea of "how hard" is to impute values in a relation. Frequent values are in general easier to impute correctly, therefore the larger the avg-freq the greater the number of rows containing "easier" values. Dually, avg-uni measures how many distinct "frequent" values are present on average; in this case, the larger avg-uni values, the greater the number of values to be considered 'frequent'.

After defining the data imputation and evaluation metrics, the focus now turns to data imputation approaches. Thus, we next briefly explore some existing machine learning, deep learning, and graph-based approaches for data imputation [33, 158, 4, 69, 68, 160, 71, 128, 84, 41, 15, 125, 16], that encompass traditional methods (e.g. mean, median, and mode [33], [158], [68]) in terms of imputation accuracy. An overview of the different approaches proposed to solve the DI problem is shown in Table 1.2 and discussed next.

Statistical Data Imputation Methods

Statistical imputation methods are widely used due to their simplicity and efficiency in handling missing data. These methods assume that the missing values follow the same distribution as the observed data, making them easy to implement in various applications. Mean, median, and mode imputation are common statistical imputation methods used to handle missing data. These techniques are based on summary statistics and provide a straightforward way to fill missing values in datasets. Mean imputation fill missing values in a dataset with the arithmetic mean of the available data [33], median imputation fills missing data with the middle value when the data is sorted, making it robust against outliers [68]. In contrast, mode imputation fills the missing data with the most occurring value in the dataset [158]. Traditional imputation methods often rely on simplistic approaches that may fail to capture the

underlying data patterns, leading to biased estimations [4, 69, 160]. The main difference between machine learning and statistical algorithms is that the former follows an optimization process.

Machine Learning Based Data Imputation Methods

K-Nearest-Neighbor.

The k-Nearest Neighbours (KNN) algorithm is frequently utilized for data imputation tasks despite its primary use as a supervised learning technique. To impute the missing values, one can use the mean (or often the weighted mean) of the top closest samples. The basic method is first to calculate the distance between a sample with missing data and other accessible samples. Euclidean distance is typically utilized to compare sample similarity. Several KNN-based imputation algorithms exist, such as CKNN [71], IKNN [121], SKNN [143], and ICKNN [71]. Techniques for calculating distance, weighting approaches, and ways for figuring out the ideal value of k vary throughout these approaches. As such, their computational complexity and performance levels differ. In certain situations, certain algorithms may perform better than others. For example, CKNN has been demonstrated to perform worse than ICKNN when there is a high percentage of missing values in the data. Due to the requirement for extensive distance calculations and the process of determining the k-nearest samples, KNN-based imputation algorithms can be computationally costly, especially for big datasets. This is one of their limitations.

Support Vector Machine.

SVM is mostly used for tasks related to classification and regression, but it may also be customized for the purpose of data imputation, providing a distinct method of managing missing values. Treating missing values as a prediction problem, SVM for imputation takes advantage of the algorithm's capacity to identify intricate patterns in the data. Using the existing data, SVM models can be trained, treating the missing values as target variables that need to be predicted. SVM is capable of modeling nonlinear relationships and produces accurate predictions. It can be better appropriate for smaller or medium-sized datasets where capturing complex data structures is essential. SVM offers a strong and adaptable approach to imputing missing values, especially when other techniques might not be sufficient [7], [128].

Decision Trees.

A Decision Tree (DT) divides the input dataset into sample groups in order to produce a hierarchical structure. A decision-making path from the root node to the leaves, which correspond to distinct sample groups, represents each group. In data imputation, samples associated with the same leaf as the incomplete sample are taken into account for estimating missing values. DT comes in several varieties, including ID3, C4.5, CRAT, CHAID, and QUEST [61, 95], each of which constructs the

tree using a unique set of techniques. Certain trees, such as CRAT and C4.5, simply go through the growing phase, but others, like CHAID, and QUEST, go through two phases: a growing phase to develop the tree and a pruning phase to simplify it [139]. Since decision trees are non-parametric algorithms, they don't make any presumptions about the distribution of the data before they begin training. Employing a collection of decision trees can enhance performance, as a single DT might not be able to extract sufficient information from the data. The decision trees are often built using bootstrapping techniques.

Gaussian Process Regression.

Gaussian Process Regression (GPR) is a probabilistic, non-linear machine learning technique that evaluates the uncertainty or variance of its predictions in addition to predicting outcomes. When used for data imputation, GPR-based techniques estimate missing values using a posterior distribution that is established by presumptions. GPR provides a more accurate forecast than deterministic approaches as it provides a range of such values rather than a point estimate. The radial basis functions are rational quadratic, Exp-Sine-Squared, and others, whereas linear kernels are some of the kernel functions that are used in GPR among the kernel functions. Each of these kernels are unique, all of them can be used to detect any patterns in the data. The linear kernel is used when the data exhibits linear relationship, while the Exp-Sine-Square kernel is used when the data is cyclic in nature. When data show more than one trait, different kernel combinations can be used to capture both. When the data exhibits both periodic and linear trends, for example, an Exp-Sine-Squared and linear kernel combination can be helpful. For a variety of applications, many GPR-based imputation techniques have been developed, including NGPR [63] and MTGPR [60]. Nevertheless, a significant drawback of GPR is its high computational cost, which renders it unsuitable for usage with big datasets.

Expectation Maximization.

When a statistical model's local maximum likelihood parameters cannot be determined directly, the Expectation Maximization (EM) procedure is used to estimate them. Observed data, unknown parameters, and hidden variables are frequently present in these models. The missing values of a dataset might be regarded as latent variables when it comes to data imputation. Derivatives of the likelihood function with respect to all values, latent variables, and unknown parameters are needed to determine the highest probability. For most statistical models, it is also usually not feasible to solve these concurrently. The expectation and maximization iterative numerical processes –used by the EM algorithm to estimate the unknown values. It is known to have a slow convergence rate, but it assures an increase in likelihood with each iteration [101, 80].

Deep Learning Based Data Imputation Methods

Auto-Encoder.

One class of deep learning algorithms, Auto-Encoders (AEs) can learn, without human supervision, to encode input data. Input, hidden, and output layers are the standard construction of an AE [129, 135]. The purpose of using the same data for both the input and output layers is to map the input to the hidden layer and then use the encoded hidden vector to reconstruct the original input. Data with missing values can be inputted into the input layer and rebuilt using the hidden layer once training is finished. By randomly masking input features during training, missing values are simulated so that the model may learn to deal with partial data. Consequently, the AE determines how to fill in the blanks by piecing together the incomplete input. Auto-Encoders (AEs) come in a variety of flavours, with Denoising Auto-Encoders (DAEs), Sparse Auto-Encoders (SAEs), Stacked Denoising Auto-Encoders (SDAEs) being the most popular and widely used. Generative models like VAEs can generate new samples under typical conditions, which makes them great for filling in missing data, and DAEs excel at dealing with input data noise, [111, 25, 46, 15].

Long Short-Term Memory.

Problems where data points are dependent on temporal correlations are well-suited to the strong deep learning technique known as Long Short-Term Memory (LSTM). LSTM is a member of the RNN family, which is well-known for its capacity to handle sequential data by retaining a memory of prior inputs [155]. When it comes to learning from longer sequences, however, LSTM overcomes the constraints of regular RNNs and goes beyond them. LSTM is superior to other architectures because it can detect both immediate and distant relationships in time-series data. The “cells”, which are specialized memory units, accomplish this by controlling the flow of information through gates, namely the “input”, “output”, and delete gates. By utilizing these gates, the model is able to selectively retain, update, or discard data, allowing it to recall significant patterns over extended durations while disregarding insignificant particulars. Since patterns in time-series data may extend over different time intervals, LSTM is particularly good at obtaining useful information from this type of data. Although LSTM has been a useful tool for time-series analysis, comparing its performance for missing data imputing to other machine learning algorithms is not as easy [94], [83], [156], [84], [41]. This is due to the fact that its imputation capabilities are typically tested in the setting of sequential data, where it excels. In time-series datasets, LSTM may be employed for imputation to fill in missing values by utilizing both historical and future data points. It learns temporal patterns during training and uses them to anticipate what the missing information should be. Making broad comparisons with imputation methods used in non-sequential data is problematic due to the fact that LSTM’s performance in this domain is highly context-dependent.

Generative Adversarial Networks.

Generative Adversarial Networks (GANs)– Since the initial introduction by Goodfellow et al. in 2014, GANs have found widespread use in several domains such as image processing, speech recognition, video prediction, fingerprint localization, and most significantly in data imputation. Two primary parts comprise a standard GAN: the Generator (G) and the Discriminator (D). The generator discover a means of converting random noise into data that cannot be distinguished from actual data. In the meantime, the generator generates data, and the discriminator is trained to differentiate between the two data types. The ultimate goal of the Generator is to generate data that is indistinguishable from real data by the discriminator. GANs have vast research opportunities for the future, but they are also full of weaknesses like instability, a vanishing gradient, and an inability to produce labeled data. In order to solve these problems, different models have been designed. For instance, conditional GAN enriches the ability to produce labeled data; on the other hand, wasserstein GAN alleviates the instability issues [98, 125].

Graph Neural Networks.

The problem of incomplete data impedes data analysis, as absent values often result in biased outcomes and inferior performance. This issue is particularly evident in relational datasets containing continuous and categorical variables.

In recent years graph neural networks have been employed for missing data imputation. GNN formulate missing data as denoising auto-encoder task, where graph edges represent similarity between patterns. A GNN encoder constructs an intermediate representation (i.e., embeddings) for each data point by combining classical projection layers with local information aggregation from similar nodes, while a decoding phase of GNN use the output of GNN encoder (embeddings) to reconstruct the imputed dataset [132]. GRAPE, a graph-based framework for feature imputation and label prediction. In GRAPE, observations and features are treated as nodes of bipartite graph, with edges denoting the observed feature values. Both tasks are addressed using graph neural networks (GNN) [154].

Temporal setting imputation using graph neural network (TSI-GNN), introduces a framework that captures sequence information for use within an aggregation function of a graph neural network. This was the first approach to employ a joint bipartite graph that integrates sequence information to address data imputation [47]. Another advanced technique, GRIMP (Graph embeddings for Relational data IMPutations), is a unique imputation technique specifically designed for large-scale relational datasets [16]. GRIMP represents relational data as a heterogeneous graph illustrating the interactions of individual cell values, attributes, and tuples. This graph-based representation is crucial for effective imputation since it enables the model to understand the underlying structure of the data and precisely infer missing values. GRIMP uses a whole range of clever tricks to make imputation more effective. It encodes relational data as a graph for integrating the data from related tuples and characteristics. This method ensures the context information for imputation is optimal. Moreover, the method of graph representation learning is used for message

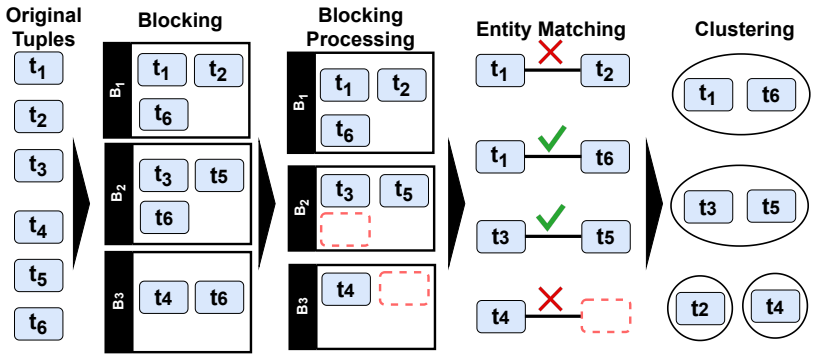


Fig. 1.2: An overview of the Entity Resolution Problem. Tuples t_i with $i \in [1, 6]$ are described in Figure 1.4.

passing and neighborhood information will be incorporated to enhance imputation. Ultimately, the model can train on datasets devoid of explicit imputation labels due to GRIMP's implementation of a self-supervised multi-task learning framework. Consequently, GRIMP exhibits remarkable flexibility in scenarios characterized by significant data absence, offering a dependable approach for data imputation in relational databases.

1.4 Entity Resolution

Entity Resolution (ER), also known as record linkage or data deduplication, is a complex and indispensable process in data management and analysis. ER's primary purpose is to identify and establish connections between tuples (belonging to possibly different relations) that pertain to the same real-world entity (e.g., same book, same person, etc.) [38], [104], [149], [78] as visually depicted in Figure 1.2. The entity resolution problem is crucial in both knowledge graphs (KGs) and multimodal knowledge graphs (MKGs), as it plays a key role in ensuring the accurate identification and integration of entities from diverse sources [157]. In KGs, this problem is fundamental for linking different data points and maintaining consistency across large datasets. In the context of multimodal knowledge graphs, entity resolution is particularly similar to entity alignment, as both involve aligning and matching entities that appear across different modalities. Entity resolution in multimodal graphs becomes even more critical, as it requires handling and aligning data from various modalities to provide a unified and comprehensive representation of knowledge [161].

Figure 1.3 represents two entities, where images associated with "Unical, Italy" indicate that it represents a university, but aligning "University of Calabria" in KG1 with "Unical, Italy" in KG2 remains challenging when relying exclusively on images or textual information. Knowledge graphs efficiently structure, manage, and retrieve

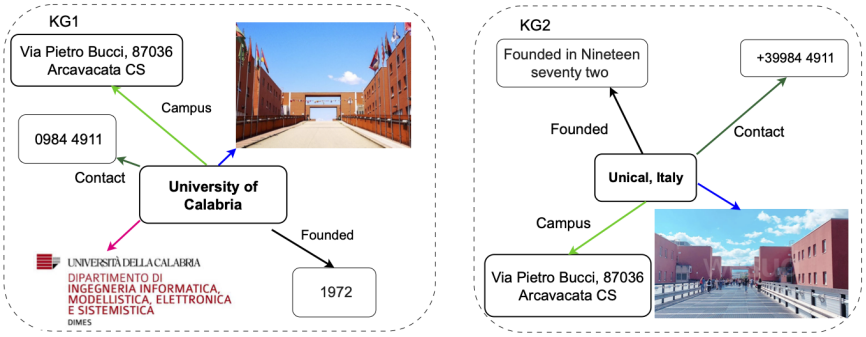


Fig. 1.3: Illustration of entity alignment between multi-modal knowledge graphs.

vast amounts of information, typically represented as triplets [161]. Two key challenges in KG research include link prediction, which involves completing missing triplets by identifying either the head or the tail, and entity matching, which focuses on identifying similar entities across two KGs and linking them via the same as predicate.

In the contemporary data landscape, characterized by an exponential growth in data across various domains, ER has emerged as a critical and increasingly vital technique. Its applications span across a multitude of fields, including healthcare, finance, and marketing [52, 65, 42, 12]. ER acts as a bridge that connects disparate data points (tuples), enabling the merging of information attributed to the same real-world entities. Doing so substantially enhances data quality, a foundational cornerstone in any data-driven endeavor. High-quality, integrated data is pivotal for informed decision-making, enabling organizations to derive meaningful insights and make more accurate predictions. Moreover, the significance of ER extends beyond data quality alone. It plays a pivotal role in enhancing the overall data analysis process. Merging data from multiple sources empowers analysts and data scientists to perform more comprehensive statistical analyses. This leads to a deeper understanding of trends, patterns, and correlations [8, 124].

We next introduce the ER problem formally.

Definition 1.13 (Entity Resolution Problem). Let r and s be two (possibly incomplete) relations, δ a similarity function, and τ a threshold value. The Entity Resolution problem consists of determining all the (matched) pairs $(t_i, t_j) \in r \times s$ such that $\delta(t_i, t_j) \geq \tau$.

Clearly, r and s might not be distinct. Moreover, we assume that r (resp. s) does not contain tuples denoting the same entity. Whenever such an assumption does not hold, we can consider the problem where the input relations are $r \cup s$ and $r \cup s$. In such a case, trivial matchings (t_i is similar to t_i) are filtered out. It is worth noting that, the ER problem can also be formulated as a learning problem where function δ can be learned from (training) data.

	Name	Surname	BirthDate	State	Salary	A.Code
t_1^k	$\perp$	Deere	Second Jan	Un. State	15k	100
t_2^k	Allisa	Jia	10th Jan	America	$\perp$	400
t_3^k	David	$\perp$	12th Feb	France	10000	300
t_4^k	John	Iser	$\perp$	Austria	9000	500
t_5^k	David	Rao	February 12th	France	10k	300
t_6^k	Alex	Deere	2nd Jan	$\perp$	15k	$\perp$

Fig. 1.4: Incomplete relation r of Example 1.14.

Hereafter we will denote all the matched pairs w.r.t. r, s, δ, and τ with $\mathcal{M}(r, s, \delta, \tau)$, or simply $\mathcal{M}$ whenever r, s, δ, and τ are understood.

Example 1.14. Consider the incomplete relation $r = \{t_1, t_2, t_3, t_4, t_5, t_6\}$ shown in Figure 1.4. Assume δ is s.t. $\delta(t_i, t_j) = 1$ if t_i and t_j share at least 3 non-null values and could have the same state, 0 otherwise. The ER process over $r \times r$ would return the set $\mathcal{M} = \{(t_1, t_6), (t_3, t_5)\}$ of matched pairs. □

It is worth noting that, the ER task could require a quadratic number (i.e.,$|r| \times |s|$) of possible comparison of tuples, although the use of indexes allows to really compare small subsets of tuples. Considering Example 1.14, we could avoid comparing tuple t_3 with tuple t_1, t_2, t_4, and tuple t_4 with tuples t_1, t_2, t_3 and t_5 because they have a different *State*, thus it is very unlikely to describe the same entity (i.e., person). This intuition is captured by the so-called *blocking phase*. During the blocking phase, tuples in $\mathcal{I}$ are partitioned in blocks by applying very simple rules. Considering again Example 1.14, there are three blocks $B_1 = \{t_1, t_2, t_6\}$, $B_2 = \{t_3, t_5, t_6\}$ and $B_3 = \{t_4, t_6\}$ (see also Figure 1.2) that groups tuples having *State* in common. Tuples with missing values in the *State* are included in all blocks (e.g., t_6).

Then, blocks could be further processed so that some tuples could be removed according to some specific simple criteria. As an example, tuple t_6 has been removed from block B_2 (resp., B_3), based on the *BirthDate*—tuples with the same *BirthDate* are retained in the block, while those with different *BirthDate* are removed. Then, after concluding the blocking phase, only tuples in the same block are compared to check whether they are similar (and thus called matched pairs). This latter phase is called *entity matching*.

Evaluation Metrics

To assess the effectiveness of the ER task, several metrics have been proposed so far.

Let $\mathcal{G}$ be correct the set of matched tuples, given by an oracle $\mathcal{O}$ (simply referred to as ground truth). As it will be clearer in the next sections, $\mathcal{G}$ is given in the training phase in the ML-based approaches solving the ER process.

- True Positive refers to the number of correct predictions of matched pairs that actually correspond to match entities by a classification model, and is denoted as

$$TP = |\mathcal{G} \cap \mathcal{M}|;$$

- True Negative gives the number of correct non-match pairs that actually correspond to non-match pairs by a classification model, and can be denoted as

$$TN = |((r \times s) \setminus G) \cap ((r \times s) \setminus M)|;$$

- False Positive gives the number of matched pairs that actually correspond to different entities, and can be denoted as

$$FP = |((r \times s) \setminus \mathcal{G}) \cap \mathcal{M}|;$$

- False Negative refers to the number of non-match pairs (wrong prediction) that actually correspond to match pairs, and can be denoted as

$$FN = |\mathcal{G} \cap ((r \times s) \setminus \mathcal{M})|;$$

- Precision measures the proportion of correctly predicted match pairs, that is

$$PR = \frac{TP}{TP + FP};$$

- Recall measures the proportion of predicted (total) match pairs, that is

$$RE = \frac{TP}{TP + FN};$$

- F1-score is the harmonic mean of precision and recall, that is

$$F1 = 2 * \frac{PR * RE}{PR + RE}.$$

Example 1.15. Consider again the incomplete relation $r = \{t_1, t_2, t_3, t_4, t_5, t_6\}$ shown in Figure 1.4. Assume $\mathcal{G} = \{(t_1, t_2), (t_3, t_5)\}$, and an ER system producing $\mathcal{M} = \{(t_1, t_2), (t_2, t_5), (t_1, t_6)\} \subseteq (r \times r)$ as set of matched pairs. Then, we have that $TP = |\{(t_1, t_2)\}| = 1$, $TN = 32$, $FP = 2$, $FN = 1$, and thus $PR = \frac{1}{3}$, $RE = \frac{1}{2}$, and $F1 = 2 * \frac{\frac{1}{3} * \frac{1}{2}}{\frac{1}{3} \frac{1}{2}} = 2$. □

The F1-score is a general metric often used to summarize the overall performance of an ER task. As it contains both precision and recall, the F1-score also captures the trade-offs between recall and precision. The F1-score is thereby especially useful when high accuracy and completeness are important. The F1-score is the harmonic mean of precision and recall. Only when both are strong, the F1-score attains high values, thus appropriately reflecting the accuracy and completeness of the ER method in solving entities. That also makes it a useful single measure for assessing the success of ER algorithms.

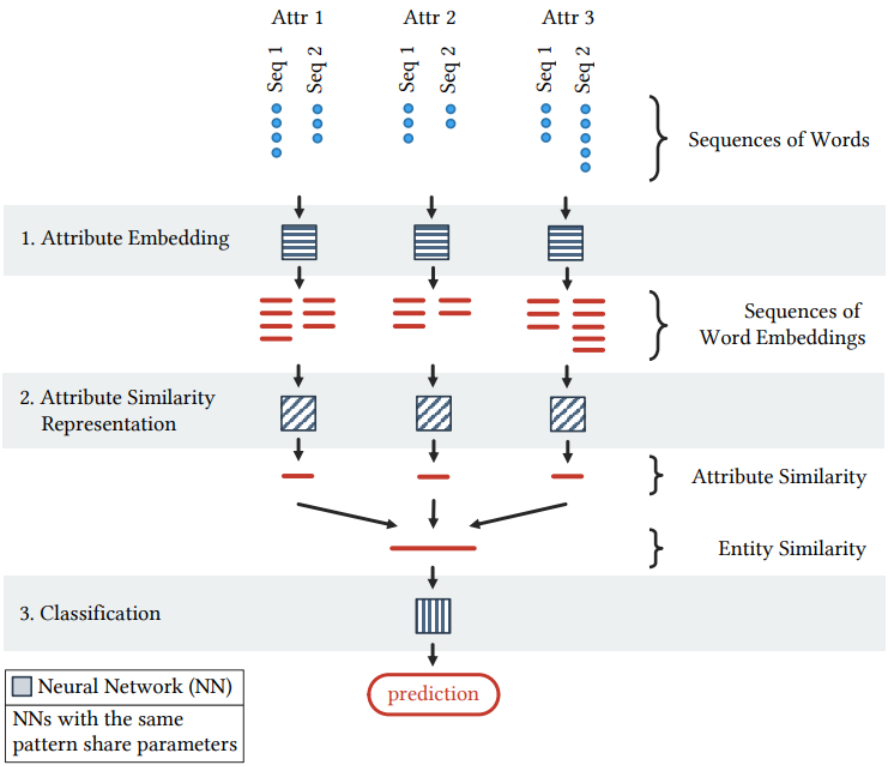


Fig. 1.5: Architecture of DeepMatcher [99].

Machine and Deep Learning Based Methods

Entity Resolution has developed strategies of over three ages: traditional, ML/DL, and now, LLM-based techniques. For a long time, the rule-based system and heuristics have been used to match and merge records in the traditional generation, which has some disadvantages, like being able to cope with complex datasets well but not at scale. ML/DL algorithms are an innovation that brought patterns in data for more intelligent matching and processing, thus allowing faster responses with larger volumes of data. Entity resolution has addressed the matching problem through various approaches, including rule-based methods [27, 36, 127, 147], crowd-sourcing [146, 45, 97], and machine and deep learning based techniques [45, 10, 24, 45, 76, 122]. Some of the deep learning-based approaches for entity resolution are given below.

DeepMatcher & DeepER.

DeepMatcher [99] explores the application of Deep Learning (DL) in ER. It introduces a design framework for DL-based solutions, emphasizing their potential to

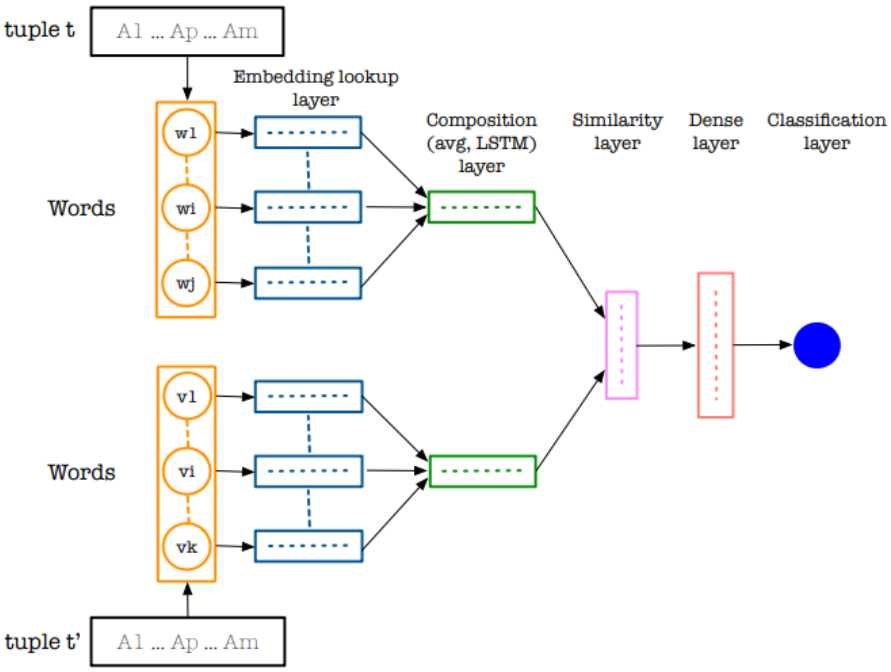


Fig. 1.6: Architecture of DeepER [35].

address challenges in ER—where traditional entity resolution methods may struggle due to limited feature extraction capabilities. The architecture of DeepMatcher [99] is given in Figure 1.5 and focuses on four optimization models: SIF (an aggregation model), recurrent neural network (RNN), that is a sequence-aware model, Attention (A Sequence Alignment Model), and Hybrid (Sequence-aware with Attention), each employing distinct strategies for attribute summarization and comparison. The Smooth Inverse Frequency (SIF) sentence embedding technique serves as a baseline due to its simplicity and effectiveness [5]. Its performance largely depends on the quality of attribute embeddings and the classifier. The attention model (i.e., a sequence alignment model) uses decomposable attention for feature summarization. It is a medium-complexity approach that jointly analyzes input sequences while learning a representation of their similarity [106]. The hybrid model combines decomposable attention with a bidirectional RNN. This hybrid is used for attribute summarization, allowing it to capture contextual information from both directions of the input sequences [148].

Similar to DeepMatcher, DeepER [35] processes two data entries and converts them into vector representations (embeddings) and then uses a feed-forward neural network to perform classification (matching, non-matching) based on the embedded vectors as given in Figure 1.6. The embeddings are generated by averaging the GloVe [108] embeddings for each attribute, which provides a straight-forward rep-

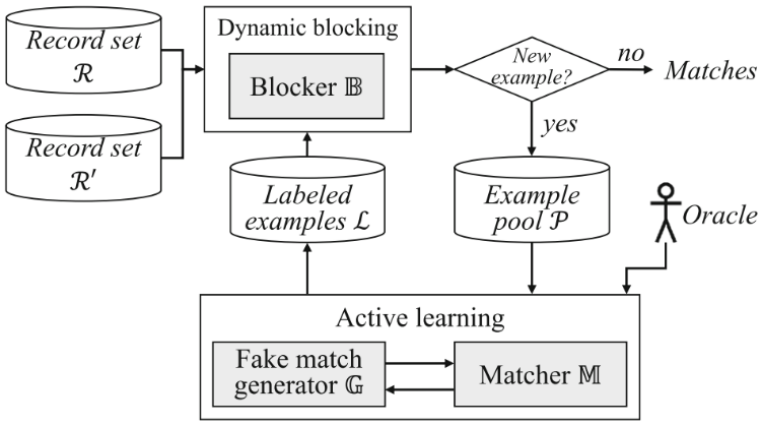


Fig. 1.7: Workflow of Deep Active Entity Matcher [62].

resentation, or using RNN to generate embeddings. Once the vectors are generated, DeepER measures the similarity between them by computing their cosine similarity, a metric that evaluates how closely aligned the vectors are in the embedding space. These similarity scores are then fed into the feed-forward neural network, which classifies whether the two entities represent a match or not.

Deep Active Entity Matcher.

Deep Active Entity Matcher (DAEM) is an approach to entity resolution that combines deep neural networks with adversarial active learning [62]. The general workflow of DAEM is depicted in Figure 1.7. This approach tackles the challenges posed by large-scale noisy data and insufficient labeled examples by introducing a dynamic blocking mechanism. Furthermore, an adversarial active learning framework is also presented that prioritize the collection of high-quality examples to improve model stability, while minimizing labeling costs.

BERT-ER.

BERT-ER addressed the computational challenges in BERT models [81]. BERT-ER introduced a siamese network architecture that independently encodes entities using BERT. By delaying the pair-wise interaction through an enhanced alignment network, the model effectively manages the computational costs associated with deep interactions. The architecture of BERT-ER is depicted in Figure 1.8. The blocking model is used to minimize the number of pair comparisons that reduce the computational costs. Furthermore, the integration of blocking and matching into a multi-task learning framework enables the optimization of both tasks. This approach not only enhances the model performance but also enhances the performance across diverse datasets.

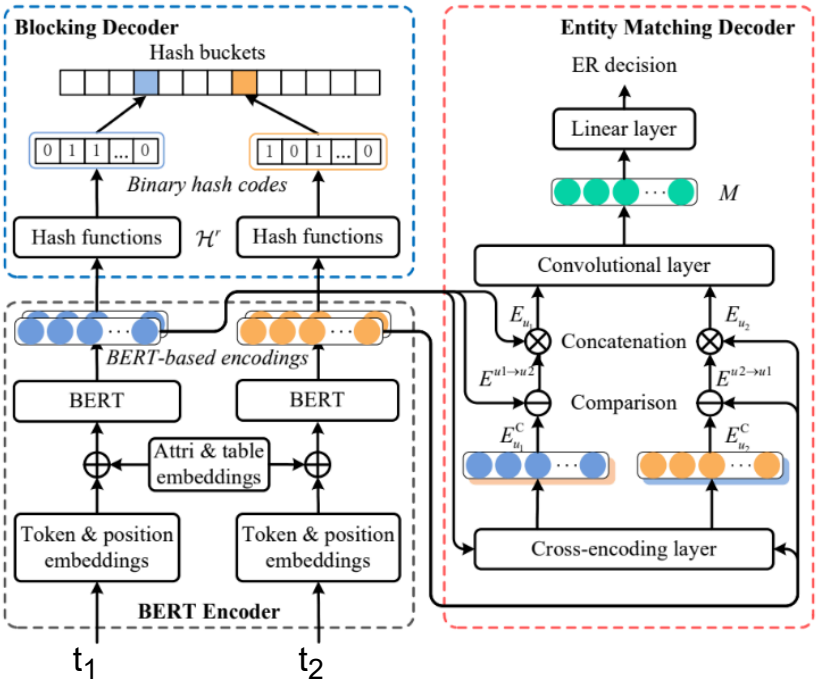


Fig. 1.8: Architecture of BERT-ER [81].

1.5 Machine Learning Architectures

In this section, we provide a brief overview of ML architectures discussed or used throughout the thesis, including Deep Neural Networks (DNN), Graph Neural Networks (GNN), Transformers (along with Sentence Transformers).

Deep Neural Networks

A supervised (neural network) learning model can be defined as a pair $M = \langle N, \Theta \rangle$ where N identifies the neural network and Θ denotes the set of its parameter values. The goal is to build a function $f_N(x;\Theta)$ (or simply $f(x;\Theta)$ whenever the neural network is understood) relating inputs x (also called instances) to outputs $\hat{y} = f(x;\Theta)$ (also called model predictions). The particular relationship between inputs and model predictions is determined by M. To train the model, a loss function $\mathcal{L}(D,\Theta)$ is adopted over a training dataset D[2] consisting in pairs (x,y), where x is an instance and y is its corresponding label (also called ground truth). The loss function quantifies the mismatch between the model prediction $\hat{y} = f(x;\Theta)$ and the ground truth y

[2] With a little abuse of notation we will denote D as both a database or dataset.

over all pairs (x, y) in D. Since the function $f(x; \Theta)$ depends on parameters Θ, the goal is to search for the parameter values that minimize the loss. An important (deep) neural network learning model is Multi-Layer Perceptron (MLP), which appeared as a building block of several learning architectures [109, 9]. An MLP $M = \langle N, \Theta \rangle$, where N has k layers, is defined by a sequence of weighted matrices $\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(k)}$, bias vectors $\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(k)}$, and fixed activation functions $a^{(1)}, \ldots, a^{(k)}$. [3] Given an input instance x, we inductively define $\mathbf{h}^{(i)} = a^{(i)}(\mathbf{h}^{(i-1)}\boldsymbol{\omega}^{(i)} + \mathbf{b}^{(i)})$ with $i \in \{1, \ldots, k\}$, assuming that $\mathbf{h}^{(0)} = x$. The output of M on x is defined as $\mathbf{h}^{(k)}$.

Graph Neural Networks

Graph Neural Networks (GNNs) are deep learning architectures designed to operate on graph-structured data. Graphs are particularly versatile data structures capable of representing complex systems where relationships (edges) occur between entities (nodes), such as social networks (e.g., interactions between individuals) or biological networks (e.g., interactions among proteins). Formally, a graph G can be defined as (V, E, X), where V is the set of nodes (with $|V| = N$ number of nodes), $E \subseteq V \times V$ is the set of edges and $X \in \mathbf{R}^{N \times d}$ is the node feature matrix storing for each node $v_i \in V$ the corresponding feature vector $x_i \in \mathbf{R}^{\mathbf{d}}$.

This general structure can be further specialized to capture more nuanced semantics, such as in the case of directed edges (where edge directionality matters), weighted edges, or encompass multiple (i.e., heterogeneous) types of entities or edges. A typical graph representation consists of the adjacency matrix $A \in \mathbf{R}^{N \times N}$ that encodes the connection between nodes as follows:

$$A_{ij} = \begin{cases} 1 & \text{if an edge between } v_i \text{ and } v_j \text{ exists} \\ 0 & \text{otherwise} \end{cases}$$

GNNs operate on graph-structured data as $G = (V, E, X)$ through iterative message-passing and neighborhood aggregation schemes. Initially, each node $v_i \in V$ has an initial feature vector $h_i^{(0)}$ corresponding to its row in the node feature matrix X. At each step of message passing, nodes exchange information with their neighbors using the connections in the edge-set E. Specifically, at a generic time-step k, each node $v_i \in V$ gathers information from its neighbors as

$$m_i^{(k)} = Aggregate(h_j^{(k-1)} \mid j \in \mathcal{N}(i))$$

where $\mathcal{N}(i)$ denotes the set of neighbors of node v_i, and $h_j^{(k-1)}$ corresponds to the feature vector of neighbor node v_j from the previous layer (or step). It should be noted that aggregation functions must ensure *permutation invariance*, as the order of neighbors does not have to affect the result; these include *sum*, *mean*, and *max* functions.

[3] In MLP, the activation functions are part of the neural network N, while matrices $\boldsymbol{\omega}$ and vectors $\mathbf{b}$ constitute parameters Θ.

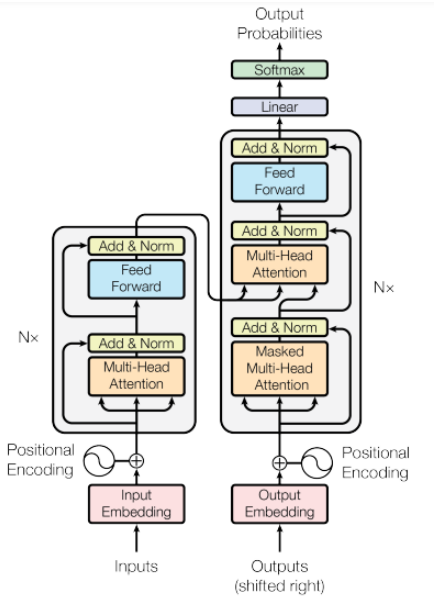


Fig. 1.9: An overview of the Transformer architecture from Vaswani et al. [140].

Following aggregation, each node updates its features using a learnable function (e.g., a feedforward neural network) as

$$h_i^{(k)} = Update(h_i^{(k-1)}, m_i^{(k)})$$

After the k-th iteration, the node features of v_i are updated to $h_i^{(k)}$, and gather information from k-top neighbors, i.e., nodes at distance k from the target node.

Transformers

The Transformer architecture, originally introduced by Vaswani et al., [140], has emerged as a foundation model in Natural Language Processing, revolutionizing a wide range of tasks by achieving state-of-the-art performances, including machine translation, summarization, generation, and language understanding. Transformers model complex dependencies between tokens using *self-attention* mechanisms, replacing traditional recurrence-based approaches more efficiently.

The Transformer architecture follows an encoder-decoder structure, as shown in Figure 1.9, where the former (on the left) maps input sequences into sequences of context-aware dense representations, i.e., *embeddings* capturing semantic relationships within input sentences, that are eventually fed into the decoder (on the right). This predicts outputs in an auto-regressive way, conditioned on the received output from the encoder, as well as the decoder output from the previous step.

The Transformer architecture consists of the following building blocks/phases:

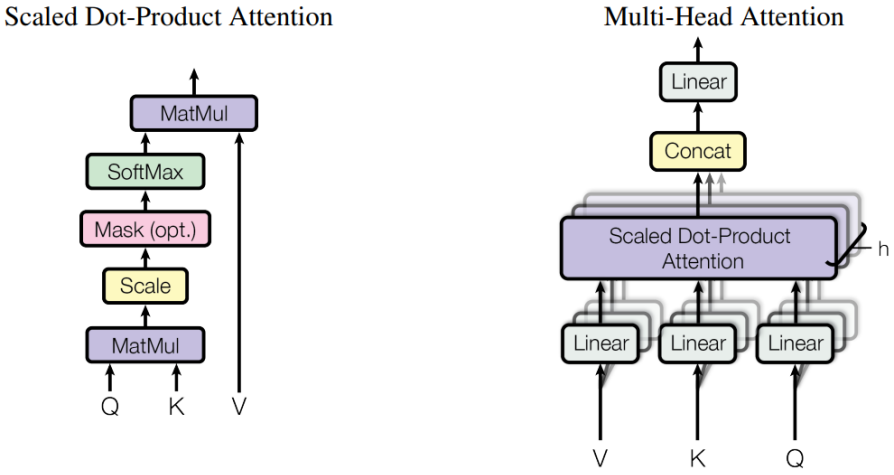


Fig. 1.10: (left) Scaled Dot-Product Attention. (right) Multi-head attention consists of several attention layers running in parallel [141].

- *Tokenization*: aimed at breaking down input sequences into smaller units, dubbed *tokens*, that the model can process. These are produced according to different strategies, ranging from the simplest word-level or character-level tokenization (whereby each word, respectively character, represents a token), to more complex strategies involving sub-words (e.g., Byte-Pair or WordPiece encoding).
- *Input Representation*: here, a tokenized input sequence $T = [\tau_1, \tau_2, ..., \tau_\ell]$ is transformed into a sequence of embeddings $e = [e_1, e_2, ..., e_\ell]$, s.t. each $e_i \in \mathbb{R}^d$ corresponds to the d-dimensional word embedding for token τ_i. Furthermore, positional encodings $P \in \mathbb{R}^{d \times \ell}$ are added to input embeddings to account for the token positions in the input sequence, resulting in a final input H.

- *Self-Attention Mechanism*: it computes pairwise interactions between tokens to capture contextual relationships. For each token, the so-called *queries*, *keys*, and *values* are computed according to learned projection matrices

$$Q = HW^Q, K = HW^K, V = HW^V,$$

where W^Q, W^K, and $W^V \in \mathbf{R}^{d \times d_k}$ are learnable parameters, and d_k is the dimensionality of the query/key space. The attention matrix is then computed as

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V.$$

In this regard, Vaswani et al [141], found it beneficial to linearly project queries, keys, and values h times with different learned parameters, to jointly attend information from different representation spaces at different positions.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O$$
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

where the projections are parameter matrices W_i^Q, W_i^K and $W_i^V \in \mathbf{R}^{d \times d_k}$, and $W^O \in \mathbf{R}^{hd_k \times d}$.

- *Position-wise Feed-Forward Networks*: after attention, each token representation is passed through a position-wise FFN s.t.

$$FFN(x) = ReLU(xW_1 + B_1)W_2 + B_2$$

The Transformer architecture lends itself very versatile and adaptable to diverse NLP tasks, involving understanding, generating, and transforming text sequences. More specifically, *encoder-only* models like BERT [30] only use the encoder part of the architecture and are primarily designed for understanding tasks (e.g., text classification, text similarity), *decoder-only* approaches as GPT [13] discard the encoder to entirely focus on autoregressive sequence generation (i.e., one token at a time, conditioned on the previous generated ones). Finally, *encoder-decoder* approaches like T5 [112] retain the full Transformer structure to operate on tasks where the input sequence needs to be transformed into an output one (e.g., machine translation).

Sentence transformers

Sentence Transformers (ST) also called pre-trained language models (PLM) are specialized adaptations of the Transformer architecture, focusing on the encoder component to produce high-dimensional embeddings that capture the semantic meaning of the sentences. This makes ST particularly well-suited for sentence-level tasks such as semantic similarity, clustering, and search. Indeed, while traditional Transformer-based models like BERT [30] perform well for token-level tasks by producing embeddings for each input token, they are suboptimal in sentence-level tasks, where naive approaches would involve aggregating (i.e., via mean or CLS-token *pooling*) individual token embeddings to derive single representation of the entire input. On the contrary, Sentence Transformers (as in the case of Sentence-BERT [115]) are specifically fine-tuned for sentence-level semantic tasks and can therefore yield more meaningful sentence embeddings, resulting in embeddings closer to a sentence's meaning as a whole [115, 23, 137, 116].

Formally, the process of sentence embedding can be described as follows.

A given input sentence s is passed through the *tokenization* process associated with the PLM, which transforms it into an initial token sequence representation $e = [e_1, ..., e_\ell]$. In our context (i.e., that of Data Imputation), to represent each tuple t as an input sentence, we concatenate the values of all its attributes resulting in a sentence $S(t) = \langle t[A_1], t_i[A_2], ..., t[A_m] \rangle$. The token sequence is then deeply contextualized by mapping each e_i into a dense vector space of dimensionality d, where d depends on the specific PLM used. The resulting output, $PLM(S(t)) \in \mathbb{R}^{d \times \ell}$, represents the *token embeddings* of tuple t. Hereinafter, we will refer to $PLM(S(t))$ simply as $PLM(t)$. To obtain a single embedding vector $\mathbf{h} \in \mathbb{R}^d$ representing the

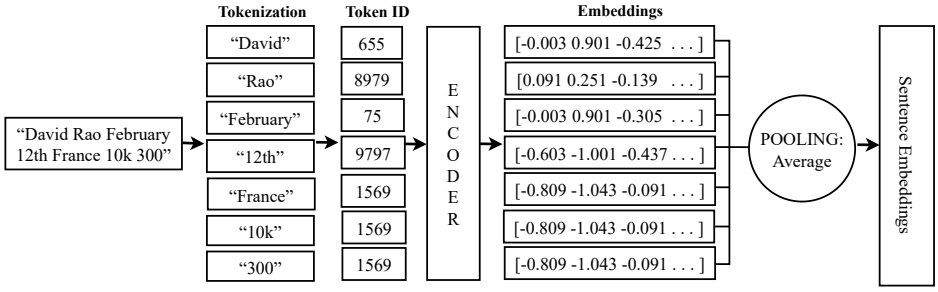


Fig. 1.11: An example of sentence tokenization and embedding.

entire textual representation of tuple t, a pooling function $pooling(\cdot)$ is applied, s.t., $\mathbf{h} = pooling(PLM(t)) \in \mathbb{R}^d$. Following Sentence Transformers, we consider the mean as the pooling function, and the resulting outputs $\mathbf{h}$ are referred to as *sentence embeddings*. During training, STs as Sentence-BERT leverage *siamese* and *triplet network* structures to push the capability of the encoder to produce semantically meaningful and appropriate representations [115].

Figure 1.11 reports an example of sentence embedding production via Sentence Transformers considering the string "David Rao February 12th France 10k 300", its tokenization, pooling and the embedding.

It should be noted that, by producing embeddings for entire sentences in a fixed-length format, these networks enable feeding sentence-level information into other workflows or applications. For instance, these embeddings are particularly suitable for comparing the meaning between different sentences, making them ideal for applications like semantic search, where a user's search query can be matched with similar information in a database, using embedding representations.

Similarity.

Given two embedding vectors (or simply embeddings) E_1 and E_2, a widely used metric to compare E_1 and E_2 is the so-called *cosine similarity*, that is computed as follows [152]:

$$\mathsf{sim}(E_1, E_2) = \frac{E_1 \cdot E_2}{||E_1|| \cdot ||E_2||} \tag{1.1}$$

Intuitively, it captures the cosine of the angle between the vectors, ranging from -1 to 1 (completely different and completely the same). A cosine similarity of 0 means E_1 and E_2 are orthogonal, that is there is no correlation between them.

Name	Chroma	Qdrant	FAISS
Developer	Chroma Inc	Zilliz	Facebook AI Research
Website	trychroma.com	qdrant.tech	faiss.ai
Initial Release	2022	2021	2017
License	Apache-2.0	Apache-2.0	MIT License
Cloud-based Only	No	NO	No
Sharding	Yes	Yes	Yes
Index Types	HNSW	HNSW	Flat, IVF, HNSW, PQ
Programming Languages	Python, TypeScript	Java, Python	Python, C++, with wrappers for others

Table 1.3: Basic properties of Vector databases.

Vector databases

Vector databases are a specialized type of database built to manage high dimensional data, enabling efficient storage, indexing, and retrieval of vector representations [72, 107, 152, 55, 145]. These databases treat sentence embeddings as points in a high-dimensional geometric space, where similar points are kept close together. This structure enables efficient search of similar points based on similarity metrics.

Some of the advanced vector databases include Chroma[4], Qdrant[5], and FAISS[6] (cf. Table 1.3). Each of these databases boasts an automated indexing mechanism, streamlining the process of data organization and retrieval for enhanced efficiency and usability. Chroma is a robust vector database designed to support the needs of language models (LMs). Its functionality goes beyond just storing high-dimensional vectors; it also allows the integration of metadata, enriching the dataset with contextual details. Qdrant, like Chroma, provides a complete solution for storing high-dimensional vectors along with their corresponding metadata payloads. Its architecture is optimized for efficient storage and retrieval, enabling fast search and similarity calculations even with large datasets. In addition to others, FAISS (*Facebook AI Similarity Search*), is a vector database developed by Facebook AI, it enables quick and efficient similarity search in high dimensional vector space, and is commonly used in applications such as information retrieval, recommendation systems, and large-scale clustering. FAISS core functionality is implemented in standard C++ for speed and efficiency, without external dependencies, and provides a wrapper allowing seamless integration with Python [34].

We next discuss the technical detail of a vector search in a vector database.

[4] https://www.trychroma.com/
[5] https://qdrant.tech/
[6] https://faiss.ai/

The vector search is a well-defined, unambiguous operation. In its simplest formulation, given a set of embedded vectors $\{x_i, \mid i = 1, \ldots, N\} \subset \mathbb{R}^d$ and a query vector $q \in \mathbb{R}^d$, it computes:

$$n = \underset{i=1\ldots N}{\text{argmin}} \|q - x_i\|$$

The minimum can be computed with a direct algorithm by iterating over all embedded vectors: this is brute force search [73].

A slightly more general and complex operation is to compute the nearest neighbors of q:

$$(n_1, ..., n_k, *, ..., *) = \underset{i=1..N}{\text{argsort}} \|q - x_i\|$$

Where argsort returns the indices of the array to sort it by increasing distances and $*$ means that an output is ignored. This is what the search method of a FAISS index returns. A related operation is to find all the elements that are within some ε distance to the query:

$$R = \{n = 1..N \text{ s.t. } q - x_n \leq \varepsilon\},$$

In the equations above, the definition of the distance is undefined. The most commonly used distances in FAISS are the L2 distance, the cosine similarity and the inner product similarity (for the latter two, the argmin should be replaced with an argmax). These measures have useful analytical properties: for example, they are invariant under d-dimensional rotations.

2

Data Imputation Using Transformers

Recent advances in data imputation have proposed sophisticated methods using relational structures and context-aware techniques. These approaches uses the relationships between attributes and cell values to enhance the quality of imputation [16]. In this chapter, we present a new data imputation algorithm that makes use of transformer architecture. We compare our algorithm with SOTA data imputation algorithm GRIMP over benchmark datasets.

2.1 State-of-the-art Approach: GRIMP

Graph embeddings for Relational data IMPutation (GRIMP) is an algorithm solving the problem of missing data [16]. GRIMP represents relational data as a heterogeneous graph and capture interactions between individual attributes, tuples, and cell values. This graph-based approach is important because it allows the model to understand the underlying structure of the data, and with this, it can effectively infer a lot of the missing values by using relationships in the data. GRIMP processes missing data in three steps: preprocessing, training, and imputation. The architecture of GRIMP is depicted in Figure 2.1. During the preprocessing step GRIMP construct a heterogeneous graph where each node represent rows and values, and edges represents the relationship between them. Additionally, GRIMP produce training corpus by injecting synthetic missing values in the considered datasets. The generated graph and corpus provide a foundation structure for training. GRIMP uses a graph neural networks to refine node features by aggregating information from closest nodes [54]. A multi-task learning models (sub-models) are then applied. These sub-models are combining hard-parameter sharing for graph neural network parameters across sub-models with attribute-specific tasks to impute data in individual attributes. Each step of GRIMP is briefly discussed below.

Graph Construction.

The first step in GRIMP is the construction of a heterogeneous graph from the considered dirty dataset that contain missing values. The graph is created row by row

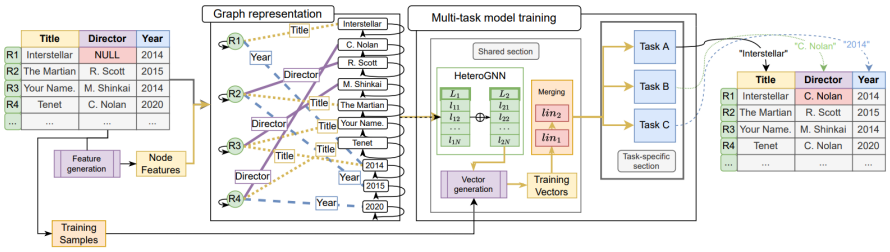


Fig. 2.1: GRIMP architecture [16].

	Title (A)	Country (B)	Year (C)
R1	NULL	*France* (b_1)	2014 (c_1)
R2	*Your Name* (a_1)	*Japan* (b_2)	2014 (c_1)
R3	*Tenet* (a_2)	*USA* (b_3)	2020 (c_2)

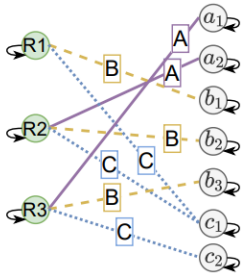


Fig. 2.2: Process of graph construction in GRIMP [16].

iteratively over the entire dataset, whereas for each new tuple a new node is added to the graph. Additionally, the algorithm also check for unique values in the tuples and if found a unique value– create a new node for it, for each tuple node and value node an edge is established. For a missing value in a cell no edge is created and hence each time the cell with missing value is ignored as shown in Figure 2.2. If a same value appear in multiple attributes, each time a new node is created so that ensuring that each corresponds to its respective attribute. GRIMP create embeddings for these graphs and then a missing value in a tuple is replaced with the closest one.

Building the Training Corpus.

GRIMP uses an efficient and noise-robust representation of tuples. In order to train the model, all non-missing values are used once by creating multiple copies of the original tuple. For each copy, a particular value is artificially treated as missing, hence enabling the model to impute it and compute a loss, since the actual value for the injected missing data is known. Each tuple, with a missing value, is copied once for each non-missing attribute to create a separate training sample. To better understand the training phase and the creation of tuple copies, we are considering example of [16], visually represented in Figure 2.3. From the Figure 2.3 is clear that $R1$ tuple generates a training sample by introducing a new null (yellow cell) for each non-null value in the same tuple. This method enable GRIMP to use all possible version of tuples, even those with a large number of missing values. Each training vector is passed through an attribute-specific imputation model.

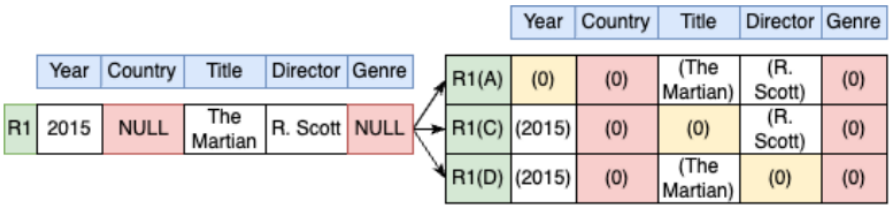


Fig. 2.3: Generation of training samples for GRIMP [16].

Graph Representation Learning.

After constructing graph and the training corpus, next, GRIMP use the constructed graph and trained corpus to learn node representations. Initial node features may be random, pre-trained, or generated using local embedding models. Then GRIMP refines these representations with GNNs, which are designed to capture generic patterns from the training data, such as relations between nodes or their neighborhood structures [49, 151, 150]. By this inductive capability, GRIMP is able to impute unseen tuples for missing values.

Multi-task Learning & Imputation.

The imputation module of GRIMP follows a multi-task learning architecture with task-specific and shared parts [16]. The task-specific part models attributes individually, treating each attribute independently as an imputation task. The shared part uses hard parameter sharing for embedding the input into a common representation for all tasks. Moreover, for the same input vector, the result might be different if the target attribute is different. This approach allow GRIMP to handle those tuples that contain more than one missing value. Imputation is achieved by building a vector representation for every dirty tuple and putting it to the trained model. For categorical values, imputation is done by choosing the value having the highest likelihood, while numerical values are imputed by processing the output of the numeric task.

Limitations of GRIMP

Although GRIMP works well in data imputation for relational databases, it has some limitations that may affect the overall efficiency and generalizability in many usage contexts. These restrictions might result from the features of the information itself, computational complexities, or some related hypothesis the model makes about relationships in data. Understanding these limitations is key to finding opportunities for improvement. In the following sections, we will look into particular drawbacks of GRIMP and how these may impact its widespread applicability in varying data contexts.

Difficulty with Rare Values.

One of the most substantial limitation of GRIMP is, how to handle filling rare (infrequent) values correctly. According to the authors, it is a known difficulty for most imputation methods (like GRIMP itself) when dealing with rare values. Subsequently, this indicates that GRIMP might be a sub-optimal approach for some scenarios with rare values to impute (resembling an issue of missing data points in today's imputation techniques).

Dependence on Data Quality.

The results of GRIMP are likely affected by the quality of the input data, as is the case with any predictive tool. Excessive noise or defects in the dataset can adversely affect imputations. The impact of noise on imputed values (robustness of synthetic complete-data imputation): GRIMP indicates an issue with the robustness of imputation in the presence of noise, suggesting that noise may compromise the veracity or precision of synthetic missing data.

Complexity of Graph Representation.

The use of a heterogeneous graph representation, while beneficial, introduces complexity. Constructing and processing these graphs can be computationally expansive, especially for large datasets. This complexity may limit the scalability of GRIMP in practical applications where quick processing times are essential.

Generalization to Different Data Types.

The paper mainly concerns relational datasets that contain categorical and numerical data. However, it does not detail how generalizable GRIMP is to other database structures or formats. This limitation of GRIMP raises issues on its generalization power to data from other sources: does this property indicate whether the algorithm will work well in datasets out of their friend-list environment?

In response to the limitations of GRIMP, we introduced a more advanced and novel data imputation technique based on sentence transformers.

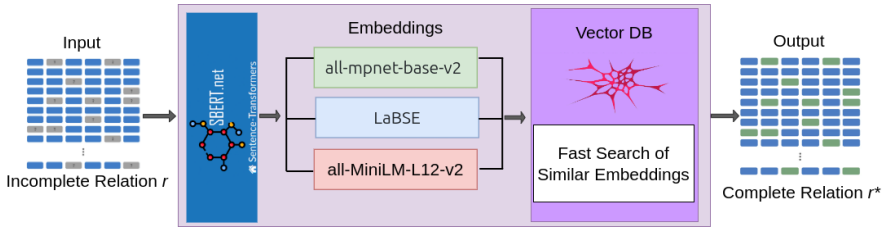


Fig. 2.4: Overview of SENTentence transformer-based data Imputation (SENT-I).

2.2 Proposed Technique for Data Imputation

To fulfill the limitations of GRIMP and to accurately fill in missing values, we introduce a novel approach that combines any well-known vector database with multiple sentence transformer models (Section 1.5), particularly aimed at enhancing accuracy for high textual datasets. Before that, we introduce the *semantic-aware data imputation problem*, discussed next.

Semantic-aware Data Imputation.

Here, we are defining the semantic similarity, that is a specialization of Definition 1.9 measuring of how similar two pieces of information are in meaning, regardless of their syntactic structure.

Definition 2.1 (Semantic similarity). Given a sentence transformer ST, a relation $r = \{t_1, ..., t_n\}$ defined over $R(A_1, ..., A_m)$ and a relation $r^* = \{t_1^*, ..., t_n^*\}$ obtained from r by replacing some attribute values $t_i[A_j]$ with a value in $dom(A_j)$, the semantic similarity between r and r^* is defined as follows:

$$\mathsf{sim}(r, r^*) = \frac{1}{n} \sum_{i=1}^{n} \frac{\mathbf{h}_i \cdot \mathbf{h}_i^*}{||\mathbf{h}_i|| \cdot ||\mathbf{h}_i^*||}$$

where $\mathbf{h}_i$ (resp., $\mathbf{h}_i^*$) denotes the embedding of tuple t_i (resp., t_i^*) produced by ST.

Thus, the semantic similarity between r and r^* is computed as the average of the cosine similarities of their tuples. Intuitively, it captures the (average of the) cosine of the angle between the vectors $\mathbf{h}_i$ and $\mathbf{h}_i^*$ (with $i \in [1, n]$).

We consider the *semantic-aware data imputation problem*, that is an instance of Definition 1.11 where function δ is replaced with the semantic similarity function sim introduced in Definition 2.1. In the following example, we compare the classical data imputation problem, where $\delta(t, t') = 1$ iff $t[i] = t'[i]$ for any $i \in [1, m]$, with our proposed semantic-aware version.

Example 2.2. Consider the following incomplete relation:

	Name	*Surname*	BirthDate
t_1	$\perp$	Deere	Second Jan
t_2	Alex	$\perp$	2nd Jan
t_3	Brad	Carter	4th March

and assume we want to impute the missing name in t_1. Note that the exact similarity between t_1 and either t_2 or t_3 is 0 (as 'Second Jan' differs from both '2nd Jan' and 4th March). Thus, classical data imputation approaches could wrongly impute 'Brad', as either t_2 or t_3 can be selected. Differently, the semantic-aware data imputation is able to recognize that t_2 is 'closer' to t_1. In fact, using any ST, it holds that the semantic similarity between t_1 and t_2 is higher than that between t_1 and t_3.[1] □

Hereinafter, the semantic-aware data imputation will be simply called data imputation. Next, we discuss our proposed data imputation approach, which enhances data imputation accuracy for high-dimensional datasets by combining vector databases with pre-trained sentence transformers.

Our proposed technique for data imputation starts by embedding the input incomplete dataset and storing it into a vector DB. As it will be clearer in what follows, the vector DB plays a key role in handling high-dimensional textual data as it represents an efficient and accurate setup for finding similar entries in large and high-dimensional databases. By employing a range of transformer models from Huggingface[2] (i.e., `all-MiniLM-L6-v2`, `all-mpnet-base-v2`, and `LaBSE`), we capture different aspects of meaning and context, allowing us to handle even the most complex and nuanced data [39]. After creating the embeddings for the input dataset, we organize and store these embeddings in a vector database, so that fast similarity searches to identify records that closely match those with missing values are conducted. When we need to fill in missing information, the system retrieves the most similar entries from the vector DB and uses their data as a reference. This allows us to fill in the nulls with values from entries that are contextually relevant, drawing on the relationships embedded in the text.

The architecture of our proposed model is depicted in Figure 2.4.

SENTence transformer-based Imputation (SENT-I)

We now present *SENTence transformer-based Imputation* (SENT-I) (cf. Algorithm 1), an algorithm specifically designed to solve the imputation problem. The architecture of SENT-I is shown in Figure 2.5. In particular, SENT-I fills the missing values in an incomplete relation r^k using a pre-trained sentence transformer ST. Algorithm 1 starts by creating a vector DB to facilitate similarity-based retrieval (line 1). For each (possibly incomplete) tuple t in r, its embedding $\mathsf{ST}(t)$ is computed using the sentence transformer ST, resulting in the desired vector DB that is a set of embeddings $V_r^{\mathsf{ST}} = \{\mathsf{ST}(t) \mid t \in r\}$. These embeddings can be managed using any of the libraries designed for efficient similarity search and clustering in high-dimensional

[1] For instance, considering the LaBSE sentence-transformer, $\mathsf{sim}(t_1, t_2) = 0.536$, while $\mathsf{sim}(t_1, t_3) = 0.377$.

[2] https://huggingface.co/

Algorithm 1 SENT-I(r, ST)

Input: An incomplete relation r defined over schema $R(A_1, ..., A_m)$; a pre-trained sentence transformer ST.
Output: A complete relation r^* obtained from r by replacing nulls with constants.

```
    // Step 1: Creating the Vector DB
 1: Compute vector DB V_r^ST = {ST(t) | t ∈ r};
 2: Compute distances among all pairs of embeddings in V_r^ST;
    // Step 2: Imputing the Missing Values
 3: for t ∈ r s.t. ⊥ ∈ t do
 4:     Let ⟨t_1, ..., t_n⟩ be the list of tuples in r ordered w.r.t. the distance of ST(t_i) from
        ST(t) (for i ∈ [1, n]);
 5:     for j ∈ [1, m] do
 6:         if t[j] = ⊥ then
 7:             for i ∈ [1, n] do
 8:                 if t_i[j] ≠ ⊥ then
 9:                     t[j] = t_i[j];
10:                     break;
11: return r* = r
```

spaces. This step is critical as it transforms the data into a format where semantic relationships among tuples are preserved and can be effectively compared. The resulting vector database V_r^{ST} forms the foundation for identifying similar tuples, which will be used to impute the missing values in the upcoming second step. Then, the algorithm computes for each pair of distinct embeddings in the vector DB their distance (line 2).

After the two preliminary steps, the algorithm proceed to replace nulls with constants. The underlying idea of the next steps is that missing values in the incomplete relation are filled with the most similar values based on the semantic similarity of tuples. Thus, for each incomplete tuple $t \in r$ (line 3), the sequence $\langle t_1, ..., t_n \rangle$ of tuples in r, ordered w.r.t. the distance of $\mathsf{ST}(t_i)$ from $\mathsf{ST}(t)$ (for $i \in [1, n]$), s.t. the higher the similarity, the closer the distance, is computed (line 4). We recall that we use the cosine as the reference similarity function between embeddings. Then, for each missing value of tuple t in position j (lines 5-10), Algorithm 1 iterates over the list $\langle t_1, ..., t_n \rangle$, and whenever the i-th closest tuple (retrieved from the i-th closest embedding $\mathsf{ST}(t_i)$) contains a constant value in position j (line 8), it replaces the null in $t[j]$ with the non-null value in $t_i[j]$ (line 9). The algorithm ends in line 11 returning r^* that is the complete updated relation r. It is worth noting that, it is not necessary to order at each step the complete set of tuples (line 4), but it is sufficient to consider only the tuples corresponding to the closest embeddings to $\mathsf{ST}(t)$ which have no nulls in correspondence of the positions containing nulls in t.

Example 2.3. Consider as input the incomplete relation r^k shown in Figure 1 (top) containing six tuples $t_1^k, \ldots, t_6^k$, and a given pre-trained sentence-transformer ST.

Algorithm 1 creates the vector DB $V_r^{\mathsf{ST}} = \{\mathsf{ST}(t^k) \mid t^k \in r^k\}$ in line 1, where e.g. $\mathsf{ST}(t_1^k) : [-0.06, -0.003, \ldots]$. Then, at line 3, the first tuple of r^k containing a missing value is $t_1^k = [\perp,$ *'Deere'*, *'Second Jan'*, *'Un. State*, '$15K$', '100'], as

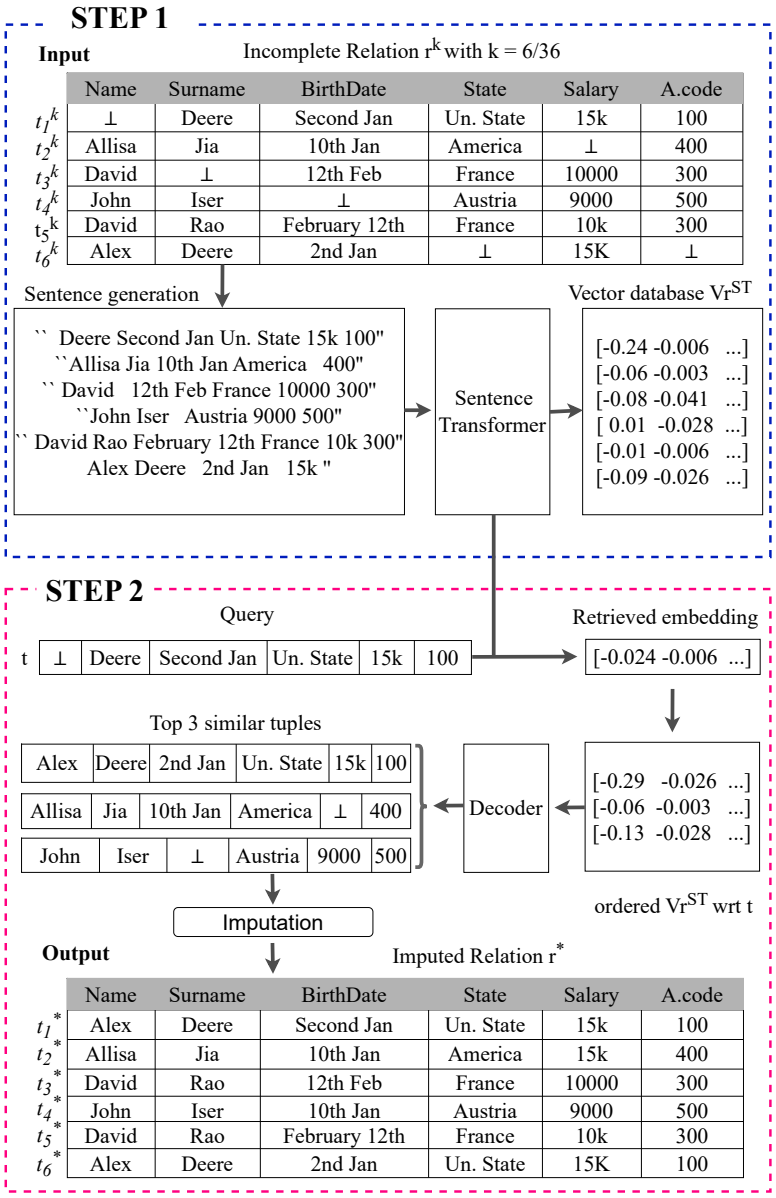


Fig. 2.5: Overview of Step 1 (top, blue dashed box) and Step 2 (bottom, magenta dashed box) of SENT-I (Algorithm 1).

Dataset	Acronym	#Rows	#Columns	#Categ	#Num	#Distinct	# Fun. dep.
IMDB	IM	4529	11	9	2	9829	0
TAX	TX	5000	12	5	7	910	6
Australian	AU	690	15	9	6	957	0
Mammogram	MA	830	6	5	1	93	0
Flare	FL	1066	13	10	3	34	0
Credit	CR	653	16	10	6	918	0
Contraceptive	CO	1473	10	8	2	65	0
Adult	AD	3016	14	9	5	289	2
Fodorszagats	FD	864	6	2	4	435	2
Thoracic	TH	470	15	14	3	355	0

Table 2.1: Statistics for all datasets used in the experimental analysis. For each dataset we report the number of rows, columns, and that of categorical and numerical columns, respectively. Moreover, we report the number of unique values in the dataset (Distinct), and the number of functional dependencies.

$t_1^k[1] = \perp$. The list of tuples $\langle t_1^k, t_6^k, t_3^k, t_2^k, t_5^k, t_4^k \rangle$ in r^k ordered w.r.t. the distance of $\mathsf{ST}(t_i^k)$ from $\mathsf{ST}(t_1^k)$ (for $i \in [2, 6]$); is computed at line 4. Then, Algorithm 1 replaces $t_1^k[1] = \perp$ with $t_6^k[1]$ at line 9. This process is repeated for each incomplete tuple of r^k. Finally, the complete relation r^*, shown in Figure 1 (bottom), is obtained from r^k by replacing nulls with constants and returned at line 11. □

Proposition 2.4. *Assuming that the input relation r does not have any column with only nulls, Algorithm 1 produces a complete relation in time bounded by $O(n^2 \times (d + m))$, where $n = |r|$, $m = ar(r)$ and d is the size of embeddings.*

As usual, we have that $d \gg m$, and we can say that the complexity of Algorithm 1 is $O(n^2 \times d)$.

In the next section, a thorough experimental analysis conducted on 10 real-world mixed datasets (i.e., containing both categorical and numerical values) showed that our technique outperforms GRIMP up to 40% in imputation accuracy.

2.3 Experimental Analysis and Results

We conduct a thorough experimental analysis on real-world datasets, and show that our technique outperforms the current state-of-the-art approach by up to 40% in imputation accuracy. We start discussing the dataset used in the evaluation analysis and then present the methodology adopted.

Datasets.

To evaluate the effectiveness of Algorithm 1, and ensure proper and fair comparison with the current state-of-the-art approach, namely GRIMP, we resorted to the same benchmark datasets used in [16]. It includes 10 diverse datasets, each containing a

single relation, with the number of rows ranging between 470 and 5000, and a number of attributes ranging between 6 and 16; attributes refer to different data types, encompassing both categorical and numerical attributes, whose number ranges in $[2, 10]$ and in $[1, 6]$, respectively. Finally, the number of distinct values ranges from 34 (in Flare DB) to 9829 (in IMDB dataset), while the number of functional dependencies does not exceed 6. All these statistics are resumed in Table 2.1.

Methodology.

We corrupt each relation $r \in \{\text{AD, AU, CO, CR, FD, FL, IM, MA, TH, TX}\}$ by injecting four different percentages $k\%$ with $k \in \{5, 10, 20, 40\}$ of nulls in each column, obtaining four different incomplete datasets r^k. The injected nulls have the type of Missing Completely At Random (MCAR) [120], because we are not considering any specific pattern in injecting nulls. Then, we repeated the process 30 times obtaining, for each complete relation r a number of $4 \times 30 = 120$ distinct incomplete datasets $r_1^5, r_1^{10}, r_1^{20}, r_1^{40}, \ldots r_{30}^5, r_{30}^{10}, r_{30}^{20}, r_{30}^{40}$. To ensure diversity and significance in the results, a different seed for the randomness is used in each of the 30 runs. For each of the 120 runs, we computed the (imputation) accuracy between r_i^k and r by means of Definition 2.1. Although we have used several sentence-transformers (e.g., `LaBSE`, `all-MiniLM-L6-v2`, `all-mpnet-base-v2`, that are publicly available at (https://huggingface.co), we show only the results for `LaBSE` that showed the best performance.

- `all-MiniLM-L6-v2`: This lightweight model is based on the architecture of MiniLM (Miniature Language Model). This model has 6 transformer layers and maps sentences and paragraphs to a 384-dimensional dense vector space and has approximately 22 million parameters, making it lightweight and efficient for deployment in resource-constrained environments. This transformer model is optimized for tasks requiring semantic understanding, such as semantic similarity, semantic search, clustering, and classification. The model is trained on various datasets containing more than one billion sentence pairs. It effectively ensures the efficiency of embeddings without compromising their quality [57, 153].
- `LaBSE`: The full form of `LaBSE` is Language-agnostic BERT Sentence Embedding, this model is specially optimized for generating high-quality sentence embeddings across a variety of languages [39]. It was developed by researchers at Google AI to incorporate the need for robust multi-lingual semantic understanding, enabling various natural language processing (NLP) tasks across multiple languages. This model ensures strong cross-lingual skills, which is important for tasks where datasets are available in multiple languages. This language model supports more than 109 languages, providing embeddings that are language-neutral, which means that sentences with similar meanings in multiple languages will have embeddings that are similar to each other in the high dimensional vector space. `LaBSE` is a large model based on the BERT architecture, with approximately 470 million parameters. `LaBSE` architecture consists of 12 hidden layers, 768 hidden units per layer, and 12 attention heads per layer.

Model	Architecture	Languages Supported	Dimensionality	Use Case	Performance	Key Features
all-MiniLM-L6-v2	Transformer (MiniLM)	Primarily English (effective on multilingual)	384	Embeddings for semantic similarity	Moderate	Lightweight and optimized for efficiency with smaller size.
LaBSE	Transformer (BERT-based)	109+ languages	768	Multilingual sentence embedding, translation, and cross-lingual retrieval	High (cross-lingual tasks)	Designed for bi-directional translation tasks and cross-lingual similarity.
all-mpnet-base-v2	Transformer (MPNet-based)	Primarily English (effective on multilingual)	768	Semantic search, clustering, and classification	High (on English)	Provides high-quality embeddings for semantic similarity.

Table 2.2: Key features and performance comparison of Hugging Face models: `all-MiniLM-L6-v2`, `LaBSE`, and `all-mpnet-base-v2`.

- `all-mpnet-base-v2`: This is a high-dimensional sentence transformer used to convert a sentence to a high-dimensional space. The model is built on the framework of MPNet (Masked and Permuted Pre-training) and leverages both masked language and permuted language modeling techniques [39]. It is optimized for the generation of sentence embeddings that are effective in tasks like semantic search, text similarity, and clustering. The architecture of this model consists of 12 transformer encoder layers, 768 hidden units per layer, and 12 attention heads per layer, making it a powerful model. The model is trained on various datasets including Stanford Natural Language Inference (SNLI), Multi-Genre NLI (MNLI), Quora Question Pairs, and Paraphrase Adversaries from Word Scrambling (PAWS) [159].

Key features and performance comparisons of the three above discussed models have been summarized in Table 2.2. The aim of testing different sentence transformer models is to ensure that the imputed values reflect diverse linguistic and contextual nuances. As vector DB we used FAISS [34], a specialized library designed for Approximate Nearest Neighbor Search (ANNS) (see Section 1.5 for background on FAISS).

For each incomplete relation r_j^k, we compute $r^{j,k}_{\mathsf{GRIMP}}$ and $r^{j,k}_{\mathsf{SENTI}}$, respectively denoting the imputed dataset by calling GRIMP and Algorithm 1 over r_j^k. Then, for each of the two (complete) datasets $r^* \in \{r^{j,k}_{\mathsf{GRIMP}}, r^{j,k}_{\mathsf{SENTI}}\}$, the overall imputation accuracy w.r.t. the ground truth r has been computed based on the average of the cosine similarity between pairs of embeddings. More in detail, for each row index $i \in [1, n = |r|]$, we compute the cosine similarity between embeddings $\mathsf{ST}(t_i)$ and $\mathsf{ST}(t_i^*)$, where t_i (resp., t_i^*) is the i-th tuple of r (resp., r^*). Formally:

$$\mathsf{acc}(r, r^*) = \frac{1}{n} \cdot \sum_{i=1}^{n} \mathsf{sim}(\mathsf{ST}(t_i), \mathsf{ST}(t_i^*)) \tag{2.1}$$

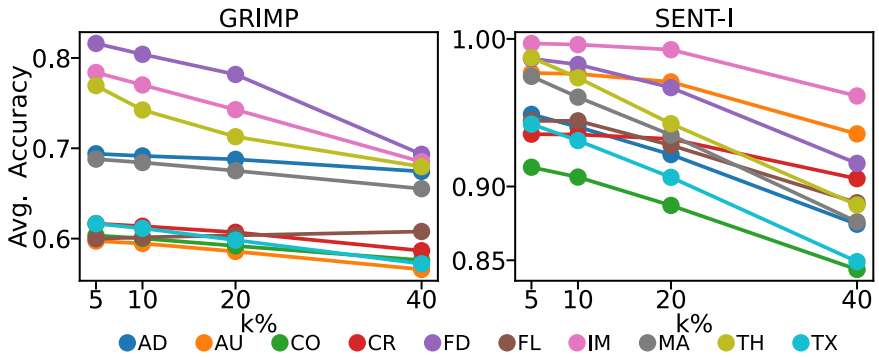


Fig. 2.6: Average accuracy of GRIMP (left) and SENT-I (right) across benchmark datasets (cf. Table 2.1) with varying percentages of nulls $k\%$ with $k \in \{5, 10, 20, 40\}$.

Thus, the higher the value of $\mathsf{acc}(r, r^*)$ the closer r and r^* from a semantic point of view.

As an example, consider the following.

Example 2.5. Consider the following strings:

- S_1 : "Deere Second Jan Un. State 15k 100";
- S_2 : "Allisa Jia 10th Jan America 400"; and
- S_6 : "Alex Deere 2nd Jan 15k";

respectively obtained from tuples t_1, t_2 and t_6 of relation r shown in Figure 2.5 (top).

The embeddings of S_1, S_2, and S_6 w.r.t. the sentence transformer ST are as follows:

- $\mathsf{ST}(t_1) = [-0.136,\ -0.003, \ldots]$;
- $\mathsf{ST}(t_2) = [-0.094,\ -0.015, \ldots]$; and
- $\mathsf{ST}(t_6) = [-0.077,\ -0.093, \ldots]$.

The (cosine) similarity between the embeddings $\mathsf{ST}(t_1), \mathsf{ST}(t_2)$ and $\mathsf{ST}(t_6)$ is as follows:

- $\mathsf{sim}(\mathsf{ST}(t_1), \mathsf{ST}(t_2)) = 0.82$,
- $\mathsf{sim}(\mathsf{ST}(t_1), \mathsf{ST}(t_6)) = 0.88$,
- $\mathsf{sim}(\mathsf{ST}(t_2), \mathsf{ST}(t_6)) = 0.85$.

This states that t_1 and t_6 are semantically similar, tough syntactically different. □

It is worth noting that the accuracy w.r.t. the semantic similarity can not be computed by using the exact accuracy, as evidenced in the next example.

Example 2.6. Consider tuples t_1 and t_6 from the previous example, and assume now that the BirthDate of t_6 is replaced with '2nd January'. Then, according to the exact accuracy we have that the similarity between t_1 and t_6 is 0, while according to the above-proposed semantic similarity measure, it holds that $\mathsf{sim}(\mathsf{ST}(t_1), \mathsf{ST}(t_6)) = 91$ (with ST is `LabSE`). □

Results.

Figure 2.6 reports the imputation accuracy for each considered dataset in the benchmark (denoted with different colors in the figure) for either GRIMP (left) and SENT-I (right). In particular, we report imputation accuracy averaged over 30 runs for each dataset and percentage of nulls under the `LaBSE` model, which was the one achieving the best performance. It is worth noting that, although the accuracy under the other two models is lower than that obtained under `LaBSE`, it does not decrease by more than 10%. This suggests that SENT-I outperforms GRIMP regardless of the specific sentence transformer model used. From the results, it is clear that for datasets such as IM, MA, AD, FD and TH, GRIMP maintains high accuracy (i.e., more than 65%), demonstrating its effectiveness on certain datasets. However, in the other datasets (i.e., TX, AU CR, FL and CO), GRIMP's accuracy is on average 57%. As the missing values grow, its accuracy drops noticeably, highlighting its challenges in dealing with higher levels of missingness. GRIMP does not perform very well with high percentages of nulls. In fact, for $k = 40$, its accuracy is on average 62%.

Compared to GRIMP, SENT-I shows better imputation accuracy across all datasets and nulls percentages. That is, SENT-I's accuracy remains high (i.e., more than 80%) for all the benchmark datasets and null percentages (i.e., $5\%, 10\%, 20\%$, and 40%). In particular, whenever the percentage of nulls is lower than 10%, SENT-I is able to impute values with an accuracy greater than 90%, independently of the dataset. We observe that SENT-I is insensitive to the nature of the data (i.e., categorical or numerical), and the performance improvement is primarily attributed to the use of sentence transformers, that are able to represent data on the basis of their semantics in the so-called latent space.

Thus, our results state that SENT-I should be the preferred method, even when the datasets are highly textual and contain a high level of missing values.

2.4 Summary

We presented *SENTence transformer-based Imputation* (SENT-I), a novel approach designed to overcome the limitations of existing methods like GRIMP by improving the accuracy and reliability of imputing missing values, particularly in datasets with substantial textual content. SENT-I combines the strengths of advanced sentence transformer models with a vector-based search mechanism to efficiently handle incomplete data. Our method embeds the dataset into a semantic space to identify and leverage similar records for accurate imputation. Through evaluation of diverse real-world datasets with varying levels of missing data, SENT-I consistently outperforms

GRIMP, achieving over 40% improvement in accuracy on average. Its performance advantage increases as missingness grows, highlighting its robustness and adaptability. These results position SENT-I as a highly effective solution for imputation tasks, capable of enhancing data quality in complex, text-rich applications.

For future work, we would like to focus on the fine-tuning of sentence transformers on topic-specific datasets to improve embedding quality. We will also focus on attention mechanisms, such as multi-head self-attention, alongside with some optimizations like summarization for long strings and serialization. We believe these enhancements could dynamically refine the imputation process by better capturing relevant contextual information and efficiently handling larger or more complex textual data.

3

Entity Resolution Using Transformers

Data integration tries to provide a single, unified, duplicate-free data representation. The benefits include better accessibility of information and improved decision-making, as the overall quality of the information will be higher as well. One of the main steps in the data integration pipeline is Entity Resolution (ER)–the process of identifying and merging records from multiple data sources that belong to the same entity [38, 104, 149, 78]. Nowadays researchers mainly approach ER tasks with Deep Learning (DL) models [35, 77, 20, 62]. These DL models are trained for binary classification, where 0 represents a non-match, and 1 represents a matched pair. The architecture of a DL model consists of two main parts, an encoder and a decoder. The encoder parts generate a piece of information for matched pairs. In contrast, the decoder classifies the results from the encoder as 1 or 0 (i.e., match or non-match).[1] In this chapter, after discussing the state-of-the-art ER (SOTA) system Ditto [85], we present a technique for the ER task in the presence of missing data that is orthogonal w.r.t. the specific ER system used. Our approach first applies data imputation on the datasets used for the ER task by making use of Algorithm 1, and second applies any existing ER system over the previously obtained datasets. We have conducted an experimental analysis on the well-known *Magellan ER benchmark* and showed that our approach outperform Ditto.

3.1 Overview of the Ditto System for ER

The current state-of-the-art approach, Ditto [85], works in the same way as discussed above. Ditto, a transformer-based architecture (BERT, SBERT, RoBERTa, and DistilBERT), can capture hidden information about the matched and non-matched pairs. These transformers bring a revolution in the task of ER. The architecture of Ditto for ER is given in Figure 3.1. Ditto is one of the first ER processes that leverage pre-trained transformer-based language models to provide deeper language understanding for ER. The process of Ditto starts with the blocker–it is used to reduce the

[1] Tough for a different purpose, the terminology encoder/decoder overlaps that of Transformer architecture

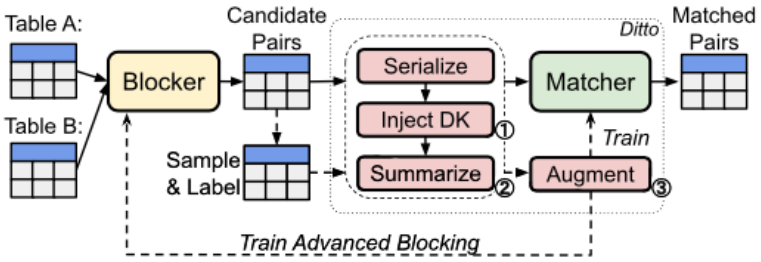


Fig. 3.1: Overview of the Ditto System [85].

number of tuple comparisons, resulting in less computation time. The second step in Ditto is *serialization, Injecting domain-knowledge, and Summarization*, these sub-steps correspond to the training phase of the Ditto model. The following sections briefly explore each step of Figure 3.1.

Blocker.

Without blocking, the process of comparing every tuple with all other tuples in a database can be computationally expansive, especially for large databases [105, 82]. Ditto employs various blocking techniques depending on the database.

Serialization.

Serialization is the process used to convert data into a specific format suitable for a ML (or DL) model to process, and then reconstruct it back to its original format. Ditto uses a special type of serialization to encode the input data. That is, the serialization of a tuple $t = [t[A_1], \ldots, t[A_{ar(t)}]$ is defined as follows:

$$\mathsf{serialize}(t) = \text{[COL]}\ A_1\ \text{[VAL]}\ t[A_1]\ \ldots\ \text{[COL]}\ A_{ar(t)}\ \text{[VAL]}\ t[A_{ar(t)}]$$

where [COL] and [VAL] are special tokens used to represent the start and end of an attribute, respectively.

Example 3.1. The serialization of tuple t_5 = [‘David’, ‘Rao’, ‘February 12th’, ‘France’, ‘10000’, ‘300’] from relation r of Figure 1.4 is the following:

$\mathsf{serialize}(t_5)$ = [COL] Name [VAL] David[COL] Surname [VAL] Rao [COL] BirthDate [VAL] February 12th [COL] State [VAL] France [COL] Salary [VAL] 10000[COL] Code [VAL] 300.

□

It is worth noting that, whenever $t[A_i] = \perp$, a blank space is used to represent $\perp$. Moreover, the serialization of a pair (t, t') of tuples is defined as $\mathsf{serialize}(t, t') =$

serialize(t) serialize(t'), that is the string obtained by concatenating the two serializations.

Thus, the language models used in Ditto (i.e., BERT, SBERT, RoBERTa, or DistilBERT) convert serialize(t, t') into embeddings, rather than the pair (t, t'). Serialized data can enhance overall performance of ER task.

Domain-knowledge.

To enhance the overall performance of ER task, Ditto inject domain-knowledge into the model, to improve the quality of the embeddings, domain-knowledge is used by refining the serialized tuples. That is, special additional tokens are added to serialize(t). Ditto implements two types of domain-knowledge, that are *span typing* and *span normalization* discussed next.

Span Typing.

Span types represent a form of domain knowledge that can be provided to Ditto. Examples include product IDs, street numbers, and publishers. These span types help Ditto reduce mismatches by ensuring that, for instance, a street number is not mistakenly matched with a year or a product ID. Table 3.1 outlines the key span types that human annotators typically consider when matching three types of entities in our benchmark datasets. The span typing adds information about the attributes' meaning (e.g., date, country, etc). After the types are recognized, the original text is replaced by a new text where special tokens are inserted to reflect the spans types.

Example 3.2. The span typing of tuple t_5 = [‘David’, ‘Rao’, ‘February 12th’, ‘France’, ‘10000’, ‘300’] after its serialization could be the following:

[COL] Name [VAL] David[COL] Surname [VAL] Rao
[COL] BirthDate [VAL] [/DATA] February [/DATA][/DATA]12th[/DATA]
[COL] State [VAL] [/COUNTRY]France[/COUNTRY]
[COL] Salary [VAL] [/Salary]10000[/Salary]
[COL] Code [VAL] [/Code]300[/code].

Intuitively, in comparing strings like the previous one, the model could exploit the novel tokens representing the domain-knowledge. □

Span Normalization.

Span normalization is another type of domain knowledge that can be provided to Ditto. Span normalization rewrites syntactically different but semantically equivalent values into a uniform string (e.g. “New York” and “ N. York” are replaced with “N.Y”). This ensures they share identical embeddings, making it easier for Ditto to recognize them as matching spans. Ditto defines a set of rewriting rules to transform spans. This process involves a function that first identifies the relevant spans and then replaces them with their rewritten versions. Ditto includes predefined rules for

Dataset	Types of important spans
Publications, movies, music	Persons (e.g., authors), year, publisher
Organizations, employers	Last 4-digit of phone, street number
Products	Product ID, brand, configurations (num.)

Table 3.1: Main span types for ER used in Ditto [85].

Operator	Explanation
span_del	Removes a randomly selected span of tokens
span_shuffle	Alters the token order within a random span
attr_del	Deletes a randomly chosen attribute and its associated value
attr_shuffle	Changes the order of all attributes
entry_swap	switches the positions of two data entries

Table 3.2: Data augmentation operators used in Ditto [85].

handling numbers, such as rounding floating-point values to two decimal places and removing commas from integers (e.g., "2,020" → "2020"). For abbreviations, developers can provide a dictionary of synonym pairs to standardize equivalent spans.

For each dataset, the specific span type utilized in Ditto is shown in Table 3.1.

Summarization.

It is worth noting that, language models are not performing well in the presence of long strings as they can not pay attention to important pieces of information, leading to a decrease in the model's performance. To overcome this limitation, Ditto selects the most important words from the data by focusing on those that are more meaningful for the ER task. Ditto ignore the tokens generated by span typing in this process and use the list of stop words from scikit-learn library. In this way, Ditto inject only the important tokens to the model.

Data augmentation.

Data augmentation, through one (or more) augmentation operator, is a common technique used to generate more training data from the original dataset. The data augmentation operators used by Ditto are summarized in Table 3.2. Some of these operators might suffer from the drawback of making the ER task harder. For example, the operator *attr_del* may delete a randomly chosen attribute and its associated value. Whenever the ER process heavily depends on a specific attribute and this has been deleted through a data augmentation, the ER system may fail to predict the match. To avoid this situation, Ditto uses a technique called *MixDA*, that computing convex interpolation between the original data and the augmented data [85]. We remand the reader to the original paper for more details.

Matcher.

The last step in Ditto is the *Matcher*, that is devoted to predict the matched pairs across the processed datasets, as also shown in Figure 1.2.

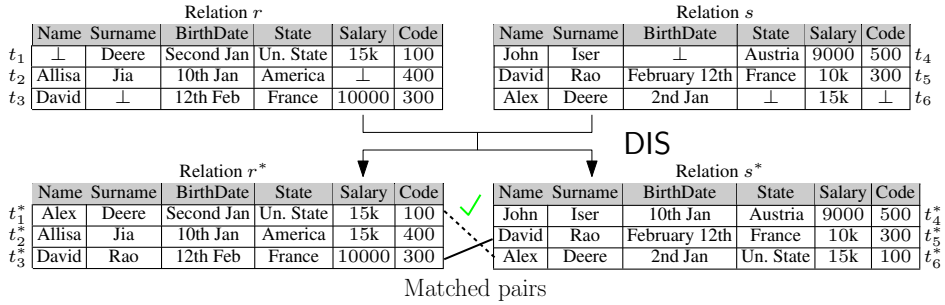


Fig. 3.2: Overview of our ER technique. DIS (resp., ERS) system is any system able to perform the Data Imputation (resp., ER) task.

3.2 Proposed Technique for Entity Resolution

Although Ditto leverages language models, it does not fully exploit their potential when applied to incomplete databases. As demonstrated in the original work, its performance significantly degrades when dealing with inconsistent databases (referred to as *dirty databases*). These dirty databases are generated from the clean version by randomly emptying certain attribute values and filling them with other randomly selected values. Moreover, the percentages of missingness (i.e., the proportion of nulls in the dataset) are not explicitly reported, and no study has been conducted to evaluate the system's efficiency as the degree of incompleteness varies.

To address this limitation, we pose the following research question:

Before performing entity resolution on incomplete databases, is it beneficial to first conduct an imputation phase?

To answer the above question, we propose a novel ER technique (summarized in Figure 3.2) that consists of an initial imputation phase followed by a standard ER phase. Particularly, in the first step, the incomplete databases from which we aim to identify matched pairs are imputed by calling any existing DI system.[2] In the second step, any ER system is called over the previously obtained (complete) databases to return the set of matched pairs.

Example 3.3. Consider tuples t_1 and t_6 respectively belonging to relations r and s shown in Figure 3.2 (left). As t_1 and t_6 shares only 2 (over 6) attribute values, an ER system could fail in classify them as matched pairs.

Consider now the case where a data imputation process has been applied to both relations r and s, obtaining respectively the complete relations r^* and s^*, also shown in Figure 3.2 (right). As t_1^* and t_6^* now shares 5 (over 6) attribute values, any ER system is more likely to classify them as matched pairs than before the imputation. □

[2] Although the ER problem has been initially defined for a single database (cf. Definition 1.13), it can be generalized to multiple databases sharing both the schema and the topic (e.g., Restaurants, Electronic, Music, etc.).

Algorithm 2 SolvER(r, s, DIS, ERS)

Input: A pair (r, s) of incomplete relations over the same schema,
a Data Imputation system DIS,
an Entity Resolution system ERS.
Output: A set $\mathcal{M} \subseteq r \times s$ of matched pairs;
// STEP 1: Performing the imputation
1: Let $u = r \cup s$;
2: Compute $u^* = \mathsf{DIS}(u)$;
// STEP 2: Performing the resolution
3: $(r^*, s^*) = \text{SPLIT}(u^*, r, s)$; // r^* and s^* denotes the complete relations obtained from r and s, respectively;
4: **return** $\mathsf{ERS}(r^*, s^*)$

Naturally, the more accurate the imputation, the higher the accuracy of the ER. It is worth noting that our technique is orthogonal to the specific ER and DI systems adopted in the two steps. Moreover, it is specifically designed to solve ER task in the presence of missing values—it would coincide with the ER system adopted whenever the input datasets contain no missing value.

In the following sections, we first describe an algorithm implementing the proposed technique that focuses on specific DI and ER solvers (cf. Algorithm 2), and then experimentally evaluate it to find a response to the above research question. The evaluation assesses the impact of varying the percentage of incompleteness on the overall ER accuracy. We demonstrate that Algorithm 2 outperforms SOTA ER system Ditto whenever Algorithm 1 is selected as DI system.

Sentence transformer-based solver for Entity Resolution (SolvER)

We now present *Sentence transformer-based solver for Entity Resolution* (SolvER) (cf. Algorithm 2), that is an algorithm specifically designed to solve ER task in terms of data imputation. Algorithm 2 takes as input (i) a pair r and s of incomplete relations, (ii) a Data Imputation system DIS, (iii) an Entity Resolution system ERS, and gives as output a set $\mathcal{M} \subseteq r \times s$ of matched pairs.

Algorithm 2 starts combining the two relations by taking their union $u = r \cup s$ (line 1), so that each of them could be imputed by looking at u (ensuring that the imputation process leverages more information on the same domain-knowledge). Intuitively, as r and s may contain overlapping or duplicate tuples, combining them may allow Algorithm 2 to impute missing data in r not only based on the information within r itself but also considering information from s, and vice versa. Then, (at line 2) a complete relation $u^* = \mathsf{DIS}(u)$ obtained from u is computed by calling any Data Imputation system (e.g. Algorithm 1 introduced in the previous chapter).

After the imputation process has been completed, the (imputed) relation u^* is split back in two complete relations r^* and s^* corresponding to the original incomplete relations r and s, respectively (line 3). The complete relations r^* and s^* are then passed into any ER systems ERS. The result of applying ERS to the complete

relations r^* and s^* is a set of matched and non-matched pairs between them, that is returned at line 4 by Algorithm 2.

Example 3.4. Consider the relations r and s shown in Figure 3.2 (left). Algorithm 2 performs an imputation phase by calling a Data Imputation solver DIS on $u = r \cup s$ (line 2). The result u^*, though function SPLIT(u^*, r, s) called at line 3, is split in two complete relations r^* and s^* (see Figure 3.2 (right)). Finally, Algorithm 2 calls (at line 4) an ER system ERS on r^* and s^* to find the matched pairs, that are denoted with green checkmarks in Figure 3.2. □

Example 3.5. Consider t_1, t_3, t_5, and t_6 from Figure 3.2, the cosine similarity between t_1 and t_6 (w.r.t. ST $=$ `LaBSE`) before imputation is 0.88, while after the imputation the similarity becomes 1, whereas the similarity between t_3 and t_5 is 0.85, and after imputation it similarity enhance to 0.97. This suggests that effective imputation let the ER process more accurate. □

In the next section we present a thorough experimental analysis conducted on three real-world datasets, and show that SolvER (i.e., Algorithm 2) outperforms Ditto.

3.3 Experimental Analysis and Results

In this section, we present a thorough experimental analysis, aimed to show the effectiveness of our approach. After providing a comprehensive description of the datasets used, we give details of the methodology used for analyzing the effectiveness of our approach, and finally discuss the obtained results.

Datasets.

We conducted experiments using dataset provided by the Magellan library, that is the main benchmark for the evaluation of ER systems.[3] Each dataset consists of candidate pairs (of tuples) from two relational tables r and s of entity records defined over the same schema. We have considered datasets whose relations have arity greater than 3 and a reasonable number of candidate pairs (i.e., greater than $10K$). As shown in Table 3.3, the percentage of matched pairs varies from 9.4% (in Walmart-Amazon) to 18.63% (in DBLP-Scholar), whereas the number of candidate pairs ranges from $10.2K$ (in Walmart-Amazon) to $28.7K$ (in DBLP-Scholar). To start with complete datasets, an initial data cleaning step has been applied to remove candidate pairs containing nulls.

[3] http://github.com/anhaidgroup/deepmatcher/

Relations r-s	**#Tuples**	**#Attributes**	**% matched pairs**	**Topic**
DBLP-ACM	12363	4	17.96	Bibliographic
DBLP-GoogleScholar	28707	4	18.63	Bibliographic
Walmart-Amazon	10242	5	9.39	Electronics

Table 3.3: Statistics of paired datasets used in the experimental analysis. We report, for each pair $r - s$ of relations, the total number of rows (i.e., $|r| + |s|$), the number of attributes (recall they coincides in r and s), and the percentage of matched pairs.

Methodology.

Each pair r-s of relations has been corrupted by injecting $k\%$ of nulls in every column of each of the two relations, with $k \in \{10, 20, 30, 40\}$. The process is repeated 30 times (using distinct seeds) to guarantee diversity and meaningful results. Then, we compared the F1-scores obtained by invoking Ditto and SolvER (i.e., Algorithm 2) on r^k-s^k, respectively. Regarding the specific data imputation system, we choose SENT-I (with parameter ST $=$ `LaBSE`) based on the strong performance demonstrated in Section 2.3, whereas for the ER system, we opted for the state-of-the-art approach Ditto [103].

Results.

Table 3.4 presents the average F1-scores for each combinations of dataset and missing percentages, comparing the performance of Ditto (columns 2-6) and SolvER (columns 7-10). Blue-colored values in the last columns represent the (average) percentage of improvement obtained by SolvER w.r.t. Ditto. From these values, we can say that our proposed SolvER approach consistently outperforms Ditto in all the combinations, with an overall percentage of improvement of 1.5% (resp., 1.7%, 2.7%, and 4.2%) with $k = 10$ (resp., 20, 30, and 40). Thus, the higher the percentage of nulls the higher the improvement. It is worth noting that, although the improvement could appear not very large in absolute terms, it is consistent with typical enhancement observed in the ER field. For example, Ditto itself demonstrated an average improvement of 4.8% over DeepMatcher [99, 85].

We also observe that the imputation accuracy achieved by SENT-I consistently remains above 80% as shown in Table 3.5 (Definition 2.1).

Regarding the efficiency, we point out that the imputation phase in SolvER takes on average the 43% of the total time required to solve the whole ER task. Although this could seem a big overhead, we should stress that in the entity resolution problem, the main focus is on effectiveness rather than on efficiency.

Thus, our findings emphasize the pivotal role of DI in enhancing ER performance on highly-textual datasets.

Dataset r - s	Ditto					SolvER			
	0%	10%	20%	30%	40%	10%	20%	30%	40%
DBLP-ACM	97.2	90.6	83.4	73.9	63.4	91.6 (+1.10%)	85.2 (+2.11%)	76.0 (+2.88%)	65.7 (+3.69%)
DBLP-Scholar	97.8	88.1	79.6	69.9	59.1	89.1 (+1.16%)	81.0 (+1.64%)	71.6 (+2.44%)	60.2 (+1.83%)
Walmart-Amazon	58.7	53.7	49.0	43.9	37.1	55.0 (+2.30%)	49.7 (+1.44%)	45.2 (+2.83%)	39.8 (+7.18%)

Table 3.4: Average F1-scores of Ditto and SolvER across pairs of relations r-s with varying percentages of nulls in 10%, 20%, 30%, and 40%. We also report the result for 0%, where SolvER coincides to Ditto. Blue-colored values represent the (average) percentage of improvement obtained by SolvER.

Dataset	Percentages of nulls (k)			
	10%	20%	30%	40%
DBLP-ACM	80.0	80.6	80.4	80.0
DBLP-Scholar	84.1	83.4	83.5	83.0
Walmart-Amazon	84.6	82.6	81.0	80.0

Table 3.5: Imputation accuracy achieved by SENT-I with varying percentages of nulls in 10%, 20%, 30%, and 40% (Definition 2.1).

3.4 Summary

We developed SolvER (Algorithm 2), a novel technique designed to tackle the challenge of ER in datasets with missing values. Instead of allowing incomplete data to hinder the process, SolvER takes a two-step approach: First, SolvER imputes the missing values using SENT-I (Alogrithm 1), that predict and restore the missing information, staying as close as possible to the original data. This step ensures the dataset is more complete and ready for further processing. Once the imputation process completed, SolvER applies any existing ER method to identify and match duplicate entities. By separating the imputation and ER phases, SolvER offers a flexible design that works with a wide range of ER approaches. Without relying on any optimization techniques, SolvER achieves impressive results, outperforming the state-of-the-art approach, Ditto. This demonstrates the strength of SolvER in effectively handling missing values and performing accurate entity resolution, making SolvER a powerful and reliable solution for real-world datasets where incomplete information is a common challenge.

Thus, we believe SolvER could benefit from the adoption of some optimization methods like implementing serialization, summarization, and fine-tuning in the construction of the vector database, thus improving the performance of both the imputation and ER processes.

4

Data Imputation in Data Querying and Exchange

Incomplete data is common in querying and data exchange. However, the existence of such incompleteness, in most cases, lead to false query outcomes. In schema mapping, most target schemas are usually constructed with missing information; however, without appropriate measures, such absence usually affects the overall performance of the exchanged information. Techniques like missing data imputation effectively address this problem as it ascertains a favorable level of data integrity; thus, data queries are more precise, and the exchanged information is also more reliable. If the issue of missing information is not effectively addressed, the entire data exchange process and the analysis of its results via queries would be compromised profoundly. Our work emphasizes the importance of managing data to maintain accurate and reliable results in both data exchange and query processes.

4.1 The Data Exchange Problem

Data exchange is the problem of transferring data from a source schema to a target schema, where the transfer process is usually described via so-called schema mappings, specifying how the data should be moved and restructured. Furthermore, the target schema may have its own constraints to be satisfied. Schema mappings and target constraints are usually encoded via standard database dependencies. Thus, given a database instance I (or simply instance) over the source (database) schema $\mathbf{S}$, the goal is to materialize an instance J over the target schema $\mathbf{T}$, called *solution*, in such a way that I and J together satisfy the dependencies.

By now, the *certain answers* semantics is the most accepted one for answering queries. The certain answers to a query is the set of all tuples that are answers to the query in every solution of the data exchange setting [37]. Although it has been formally shown that for positive queries (e.g., conjunctive queries) the notion of solution of [37] is the right one to use, for more general queries such solutions become inappropriate, as they easily lead to counterintuitive results. Before describing the problem, we introduce some notation.

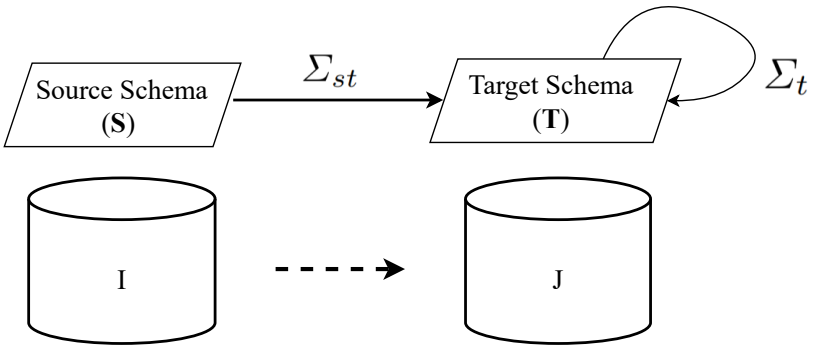


Fig. 4.1: An overview of the Data Exchange problem [43].

Dependencies.

Other than constants, attributes and relation names, we assume the presence of another pairwise disjoint countably infinite set Var of *variables*. A *term* is a constant, a variable, or a null. For a set of atoms S, $\mathsf{dom}(S)$ is the set of all terms in S, whereas $\mathsf{var}(S)$ is the set $\mathsf{dom}(S) \cap Var$. A *homomorphism* from a set of atoms S to a set of atoms S' is a function $h : \mathsf{dom}(S) \to \mathsf{dom}(S')$ that is the identity on *Consts*, and such that for each atom $R(t) = R(term_1, \ldots, term_n) \in S$, $R(h(t)) = R(h(term_1), \ldots, h(term_n)) \in S'$.

A *tuple-generating dependency* (TGD) ρ (over a schema $\mathbf{S}$) is a first-order formula of the form $\forall \mathbf{x}, \mathbf{y}\, \varphi(\mathbf{x}, \mathbf{y}) \to \exists \mathbf{z}\, \psi(\mathbf{y}, \mathbf{z})$, where $\mathbf{x}, \mathbf{y}, \mathbf{z}$ are disjoint tuples of variables, and φ and ψ are conjunctions of atoms (over $\mathbf{S}$) without nulls, and over the variables in $\mathbf{x}, \mathbf{y}$ and $\mathbf{y}, \mathbf{z}$ respectively. The *body* of ρ, denoted $\mathsf{body}(\rho)$, is $\varphi(\mathbf{x}, \mathbf{y})$, whereas the *head* of ρ, denoted $\mathsf{head}(\rho)$, is $\psi(\mathbf{y}, \mathbf{z})$. We use $\mathsf{exvar}(\rho)$ to denote the tuple $\mathbf{z}$ and $\mathsf{fr}(\rho)$ to denote the tuple $\mathbf{y}$, also called the *frontier of* ρ.

An *equality-generating dependency* (EGD) ζ (over a schema $\mathbf{S}$) is a first-order formula of the form $\forall \mathbf{x}\, \varphi(\mathbf{x}) \to x = y$, where $\mathbf{x}$ is a tuple of variables, φ a conjunction of atoms (over $\mathbf{S}$) without nulls, and over $\mathbf{x}$, and $x, y \in \mathbf{x}$. The *body of* ζ, denoted $\mathsf{body}(\zeta)$, is $\varphi(\mathbf{x})$, and the *head of* ζ, denoted $\mathsf{head}(\zeta)$, is the equality $x = y$. For clarity, we will omit the universal quantifiers in front of dependencies and replace the conjunction symbol $\wedge$ with a comma. Moreover, with a slight abuse of notation, we sometimes treat a conjunction of atoms as the *set* of its atoms. We say that an instance I:

- *satisfies a TGD* ρ if for every homomorphism h from $\mathsf{body}(\rho)$ to I, there is an extension h' of h such that h' is a homomorphism from $\mathsf{head}(\rho)$ to I;
- *satisfies an EGD* $\zeta = \varphi(\mathbf{x}) \to x = y$, if for every homomorphism h from $\mathsf{body}(\zeta)$ to I, $h(x) = h(y)$;
- *satisfies* a set of TGDs and EGDs Σ if I satisfies every TGD and EGD in Σ.

Queries.

A *query* $Q(\mathbf{x})$, with free variables $\mathbf{x}$, is a first-order (FO) formula $\varphi(\mathbf{x})$ with free variables $\mathbf{x}$. The *arity* of $Q(\mathbf{x})$, denoted $ar(Q)$, is the number $|\mathbf{x}|$. The *output* of $Q(\mathbf{x})$ over an instance I, denoted $Q(I)$, is the set $\{t \in \mathsf{dom}(I)^{|\mathbf{x}|} \mid I \models \varphi(t)\}$, where $\models$ is FO entailment.[1] A query is *Boolean* if it has arity 0, in which case its output over an instance is either the empty set or the empty tuple $\langle\rangle$. A *conjunctive query* (CQ) is a query of the form $Q(\mathbf{x}) = \exists \mathbf{y}\, \varphi(\mathbf{x}, \mathbf{y})$, where $\varphi(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms over $\mathbf{x}$ and $\mathbf{y}$. A *union of conjunctive queries* (UCQ) is a query of the form $Q(\mathbf{x}) = \bigvee_{i=1}^{n} Q_i(\mathbf{x})$, where each $Q_i(\mathbf{x})$ is a CQ. We also refer to UCQs as *positive queries*.

Definition 4.1 (Data Exchange Setting). *A* data exchange setting *(or simply setting) is a tuple of the form $\mathcal{S} = \langle \boldsymbol{S}, \boldsymbol{T}, \Sigma_{st}, \Sigma_t \rangle$, where $\boldsymbol{S}, \boldsymbol{T}$ are disjoint schemas, called* source *and* target *schema, respectively; Σ_{st} is a finite set of TGDs, called the* source-to-target TGDs *of $\mathcal{S}$, such that for each TGD $\rho \in \Sigma_{st}$, $\mathsf{body}(\rho)$ is over $\boldsymbol{S}$ and $\mathsf{head}(\rho)$ is over $\boldsymbol{T}$; Σ_t is a finite set of TGDs and EGDs over $\boldsymbol{T}$, called the* target dependencies *of $\mathcal{S}$.*

We say $\mathcal{S}$ is *TGD-only* if Σ_t contains only TGDs. A *source (resp., target) instance of $\mathcal{S}$* is an instance I over $\mathbf{S}$ (resp., $\mathbf{T}$). We assume that source instances are complete databases, i.e., they do not contain nulls. Given a source instance I of $\mathcal{S}$, a *solution of I w.r.t. $\mathcal{S}$* is a target instance J of $\mathcal{S}$ such that $I \cup J$ satisfies Σ_{st} and J satisfies Σ_t [37]. We use $\mathbf{sol}(I, \mathcal{S})$ to denote the set of all solutions of I w.r.t. $\mathcal{S}$.

Definition 4.2 (Certain answers). *Given a data exchange setting $\mathcal{S} = \langle \boldsymbol{S}, \boldsymbol{T}, \Sigma_{st}, \Sigma_t \rangle$, a source instance I of $\mathcal{S}$ and a query Q over $\boldsymbol{T}$, the* certain answers to Q over I w.r.t. *$\mathcal{S}$ is the set $\mathsf{cert}_{\mathcal{S}}(I, Q) = \bigcap_{J \in \mathbf{sol}(I,\mathcal{S})} Q(J)$.*

To distinguish between the notion of solution (resp., certain answers) above and the one defined in Section 4.2, we will refer to the former as *classical*.

A *universal solution* of I w.r.t. $\mathcal{S}$ is a solution $J \in \mathbf{sol}(I, \mathcal{S})$ such that, for every $J' \in \mathbf{sol}(I, \mathcal{S})$, there is a homomorphism from J to J' [37]. Letting $Q(J)_{\downarrow} = Q(J) \cap \mathit{Consts}^{|\mathbf{x}|}$, for any instance J and query $Q(\mathbf{x})$, the following is well-known:

Theorem 4.3 ([37]). *Consider a data exchange setting $\mathcal{S}$, a source instance I of $\mathcal{S}$ and a positive query Q. If J is a universal solution of I w.r.t. $\mathcal{S}$, then $\mathsf{cert}_{\mathcal{S}}(I, Q) = Q(J)_{\downarrow}$.*

Example 4.4. Consider a data exchange setting denoted by $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t \rangle$, where $\mathbf{S}$ is the source schema, storing orders about products in a binary relation Ord, where the first argument is the id of the order, and the second one specifies whether the order has been paid. Moreover, $\mathbf{T}$ is the target schema having unary relations AllOrd and Paid, storing all orders and paid orders, respectively. The schema mapping is described by the source-to-target TGDs Σ_{st}:

[1] We assume active domain semantics, i.e., quantifiers range over the terms in the given instance.

$$\rho_1 = \forall x, y\ \mathrm{Ord}(x, y) \rightarrow \mathsf{AllOrd}(x), \qquad \rho_2 = \forall x\ \mathsf{Ord}(x, \mathsf{yes}) \rightarrow \mathsf{Paid}(x).$$

In this example, we assume that the set of target dependencies Σ_t is empty. The above schema mapping states that all orders in the source schema must be copied to the AllOrd relation, and all the paid orders must be copied to the Paid relation. Assume the source instance is as follows:

$$I = \{\mathsf{Ord}(1, \mathsf{yes}), \mathsf{Ord}(2, \mathsf{no})\},$$

and assume we want to pose the query Q over the target schema asking for all the unpaid orders. This can be written as the following FO query:

$$Q(x) = \mathsf{AllOrd}(x) \wedge \neg \mathsf{Paid}(x).$$

One would expect the answer to be $\{2\}$, since the schema mapping above is simply copying I to the target schema, and hence $J = \{\mathsf{AllOrd}(1), \mathsf{AllOrd}(2), \mathsf{Paid}(1)\}$ should be the only candidate solution. However, under the classical notion of solution of [37], also the instance $J' = \{\mathsf{AllOrd}(1), \mathsf{AllOrd}(2), \mathsf{Paid}(1), \mathsf{Paid}(2)\}$ is a solution (since $I \cup J'$ satisfies the TGDs), and every order in J' is paid. Hence, the certain answers to Q, which are computed as the intersection of the answers over all solutions, are empty. □

The issue above arises because the classical notion of solution is too permissive, in that it allows the existence of facts in a solution that have no support from the source (e.g., $\mathsf{Paid}(2)$ in the solution J' of Example 4.4 above).

Some efforts exist in the literature that provide alternative notions of solutions for which certain answers to general queries become more meaningful. Prime examples are the works of [59] and [58]. In both approaches, the certain answers in the example above are $\{2\}$. However, also the works above have their own drawbacks. In [59], so-called *CWA-solutions* are introduced, which are a subset of the classical solutions with some restrictions. However, these restrictions are so severe that certain answers over such solutions fail to capture certain answers over classical solutions, when focusing on positive queries. Moreover, even when focusing on more general queries, answers can still be counterintuitive.

Example 4.5. Consider the data exchange setting $\mathcal{S} = \langle \mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t \rangle$, where $\mathbf{S}$ stores employees of a company in the unary relation Emp. For *some* employees, the city they live in is known, and it is stored in the binary relation KnownC.

The target schema $\mathbf{T}$ contains the binary relation EmpC, storing employees and the cities they live in, and the binary relation SameC, storing pairs of employees living in the same city.

The sets $\Sigma_{st} = \{\rho_1, \rho_2\}$ and $\Sigma_t = \{\rho_3, \zeta\}$ are as follows (for simplicity, we omit the universal quantifiers):

$\rho_1 = \mathsf{Emp}(x) \rightarrow \exists z\, \mathsf{EmpC}(x, z)$,
$\rho_2 = \mathsf{KnownC}(x, y) \rightarrow \mathsf{EmpC}(x, y)$,
$\rho_3 = \mathsf{EmpC}(x, y), \mathsf{EmpC}(x', y) \rightarrow \mathsf{SameC}(x, x')$,
$\zeta = \mathsf{EmpC}(x, y), \mathsf{EmpC}(x, z) \rightarrow y = z$.

The above setting copies employees from the source to the target. The TGD ρ_1 states that every copied employee x must have some city z associated, whereas ρ_2 states that when the city y of an employee x is known, this should be copied as well. Moreover, the target schema requires that employees living in the same city should be stored in relation SameC (ρ_3), and each employee must live in only one city (ζ). Assume the source instance is

$$I = \{\mathsf{Emp}(\mathsf{john}), \mathsf{Emp}(\mathsf{mary}), \mathsf{KnownC}(\mathsf{john}, \mathsf{miami})\},$$

and assume our query Q asks for all pairs of employees living in different cities. This can be written as:

$$Q(x, x') = \exists y \exists y' \, \mathsf{EmpC}(x, y) \wedge \mathsf{EmpC}(x', y') \wedge \neg \mathsf{SameC}(x, x').$$

One would expect that the set of certain answers to Q is empty, since it is not certain that john and mary live in different cities. However, no CWA-solution admits mary and john to live in the same city, and thus $(\mathsf{john}, \mathsf{mary})$ is a certain answer under the CWA-solution-based semantics. □

The approach of [58], where the notion of GCWA*-solution is presented, seems to be the most promising one. For positive queries, certain answers w.r.t. GCWA*-solutions coincide with certain answers w.r.t. classical solutions. Moreover, GCWA*-solutions solve some other limitations of CWA-solutions, like the one discussed in Example 4.5. However, the practical applicability of this semantics is somehow limited, since the (rather involved) construction of GCWA*-solutions easily makes certain query answering undecidable, even for very simple settings with only two source-to-target TGDs, and no target dependencies.

Other semantics have been proposed in [87], but they are only defined for data exchange settings without target dependencies. Hence, one needs to assume that the target schema has no dependencies at all.

We propose a new notion of data exchange solution, dubbed *supported solution*, which allows us to deal with general queries, but at the same time is suitable for practical applications. That is, we show that certain answers under supported solutions naturally generalize certain answers under classical solutions, when focusing on positive queries. Moreover, such solutions do not make any assumption on how values associated to existential variables compare to other values, hence solving issues like the ones of Example 4.5.

As expected, there is a price to pay to get meaningful answers over general queries: we show that certain answering is undecidable for general settings, but becomes coNP-complete when we focus on weakly-acyclic dependencies.

4.2 Proposed Approach

The goal of this section is to introduce a new notion of solution for data exchange that we call *supported*. As already discussed, the main issue we want to solve w.r.t.

classical solutions is that such solutions are too permissive, i.e., they allow for the presence of facts that are not a certain consequence of the source instance and the dependencies. Consider again Example 4.4. The (classical) solution J' in Example 4.4 is not supported, since from the source instance I and the dependencies, we cannot conclude that the fact $\mathsf{Paid}(2)$ should occur in the target. On the other hand, the solution $J = \{\mathsf{AllOrd}(1), \mathsf{AllOrd}(2), \mathsf{Paid}(1)\}$ is supported: it contains precisely the facts supported by I and the dependencies, and no more than that. Similarly, considering Example 4.5, the instance $J = \{\mathsf{EmpC}(\mathsf{john}, \mathsf{miami}), \mathsf{EmpC}(\mathsf{mary}, \mathsf{chicago}), \mathsf{SameC}(\mathsf{john}, \mathsf{mary})\}$ is a solution, but it is not supported, since from the source and the dependencies we cannot certainly conclude that john and mary live in the same city. We now formalize the above intuitions.

Consider a TGD ρ and a mapping h from the variables of ρ to *Consts*. We say that a TGD ρ' is a *ground version of* ρ (via h) if $\rho' = h(\mathsf{body}(\rho)) \rightarrow h(\mathsf{head}(\rho))$.

Definition 4.6 (ex-choice). *An* ex-choice *is a function γ, that given as input a TGD $\rho = \varphi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, \psi(\mathbf{y}, \mathbf{z})$ and a tuple $t \in \mathit{Consts}^{|\mathbf{y}|}$, returns a set $\gamma(\rho, t)$ of pairs of the form (z, c), one for each existential variable $z \in \mathsf{exvar}(\rho)$, where c is a constant of Consts.*

Note that if ρ does not contain existential variables, $\gamma(\rho, t)$ is the empty set.

Intuitively, given a TGD, an ex-choice specifies a valuation for the existential variables of the TGD which depends on a given valuation of its frontier variables.

We now define when a ground version of a TGD indeed assigns existential variables according to an ex-choice.

Definition 4.7 (Coherence). *Consider a TGD $\rho = \varphi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, \psi(\mathbf{y}, \mathbf{z})$, an ex-choice γ and a ground version ρ' of ρ via some mapping h. We say that ρ' is* coherent with γ *if for each existential variable $z \in \mathsf{exvar}(\rho)$, $(z, h(z)) \in \gamma(\rho, h(\mathbf{y}))$.*

For a set Σ of TGDs and EGDs, and an ex-choice γ, Σ^γ denotes the set of dependencies obtained from Σ, where each TGD ρ in Σ is replaced with all ground versions of ρ that are coherent with γ. Note that the set Σ^γ can be infinite. We now present our notion of solution.

Definition 4.8 (Supported Solution). *Consider a setting $\mathcal{S} = \langle \boldsymbol{S}, \boldsymbol{T}, \Sigma_{st}, \Sigma_t \rangle$ and a source instance I of $\mathcal{S}$. A target instance J of $\mathcal{S}$ is a* supported solution of I w.r.t. $\mathcal{S}$ *if there exists an ex-choice γ such that $I \cup J$ satisfies Σ_{st}^γ and J satisfies Σ_t^γ, and there is no other target instance $J' \subsetneq J$ of $\mathcal{S}$ such that $I \cup J'$ satisfies Σ_{st}^γ and J' satisfies Σ_t^γ.* □

Note that a supported solution contains no nulls. We use $\mathsf{ssol}(I, \mathcal{S})$ to denote the set of all supported solutions of I w.r.t. $\mathcal{S}$.

Example 4.9. Consider the data exchange setting $\mathcal{S}$ and the source instance I of Example 4.5. The target instance $J = \{\mathsf{EmpC}(\mathsf{john}, \mathsf{miami}), \mathsf{EmpC}(\mathsf{mary}, \mathsf{chicago})\}$ is a supported solution of I w.r.t. $\mathcal{S}$. Indeed, consider the ex-choice γ such that $\gamma(\rho_1, \mathsf{john}) = \{(z, \mathsf{miami})\}$, and $\gamma(\rho_1, \mathsf{mary}) = \{(z, \mathsf{chicago})\}$. Then, Σ_{st}^γ is

$$\{\mathsf{KnownC}(\alpha,\beta) \rightarrow \mathsf{EmpC}(\alpha,\beta) \mid \alpha,\beta \in \mathit{Consts}\} \cup$$
$$\{\mathsf{Emp}(\alpha) \rightarrow \mathsf{EmpC}(\alpha,\beta) \mid \alpha \in \mathit{Consts} \wedge (z,\beta) \in \gamma(\rho_1,\alpha)\},$$

whereas Σ_t^γ is the set containing the EGD ζ of Example 4.5, and the set of TGDs

$$\{\mathsf{EmpC}(\alpha,\beta), \mathsf{EmpC}(\alpha',\beta) \rightarrow \mathsf{SameC}(\alpha,\alpha') \mid \alpha,\alpha',\beta \in \mathit{Consts}\}.$$

Clearly, $I \cup J$ satisfies Σ_{st}^γ, and J satisfies Σ_t^γ, and any other strict subset J' of J is such that $I \cup J'$ does not satisfy Σ_{st}^γ. Another supported solution is $\{\mathsf{EmpC}(\mathsf{john},\mathsf{miami}), \mathsf{EmpC}(\mathsf{mary},\mathsf{miami}), \mathsf{SameC}(\mathsf{john},\mathsf{mary})\}$. □

With the notion of supported solution in place, it is now straightforward to define the supported certain answers.

Definition 4.10 (Supported Certain Answers). *Consider a data exchange setting $\mathcal{S}$, a source instance I of $\mathcal{S}$ and a query Q over $\boldsymbol{T}$. The* supported certain answers to Q over I w.r.t. $\mathcal{S}$ *is the set of tuples* $\mathsf{scert}_\mathcal{S}(I,Q) = \bigcap_{J \in \mathsf{ssol}(I,\mathcal{S})} Q(J)$.

Example 4.11. Consider the data exchange setting $\mathcal{S}$, the source instance I, and the query Q of Example 4.4. It is not difficult to see that the only supported solution of I w.r.t. $\mathcal{S}$ is the instance $J = \{\mathsf{AllOrd}(1), \mathsf{AllOrd}(2), \mathsf{Paid}(1)\}$. Thus, the supported certain answers to Q over I w.r.t. $\mathcal{S}$ are $\mathsf{scert}_\mathcal{S}(I,Q) = Q(J) = \{2\}$. Consider now the data exchange setting $\mathcal{S}$, the source instance I, and the query Q of Example 4.5. Then, one can verify that $\mathsf{scert}_\mathcal{S}(I,Q) = \emptyset$. □

We now start establishing some important results regarding supported solutions and supported certain answers. The following theorem states that supported solutions are a refined subset of the classical ones, but whether a supported solution exists is still tightly related to the existence of a classical one.

Theorem 4.12. *Consider a data exchange setting $\mathcal{S}$. For every source instance I of $\mathcal{S}$, its holds that* (1) $\mathsf{ssol}(I,\mathcal{S}) \subseteq \mathbf{sol}(I,\mathcal{S})$, *and* (2) $\mathsf{ssol}(I,\mathcal{S}) = \emptyset$ *iff* $\mathbf{sol}(I,\mathcal{S}) = \emptyset$.

Regarding certain answers, we show that supported solutions indeed enjoy an important property: supported certain answers and classical certain answers coincide, when focusing on positive queries. Note that this does not necessarily follow from Theorem 4.12.

Theorem 4.13. *Consider a setting $\mathcal{S} = \langle \boldsymbol{S}, \boldsymbol{T}, \Sigma_{st}, \Sigma_t \rangle$ and a positive query Q over $\boldsymbol{T}$. For every source instance I of $\mathcal{S}$,* $\mathsf{scert}_\mathcal{S}(I,Q) = \mathsf{cert}_\mathcal{S}(I,Q)$.

From the above, we conclude that for positive queries, certain query answering can be performed as done in the classical setting, and thus all important results from that setting, like query answering via universal solutions, carry over.

Corollary 4.14. *Consider a setting $\mathcal{S} = \langle \boldsymbol{S}, \boldsymbol{T}, \Sigma_{st}, \Sigma_t \rangle$ and a positive query Q over $\boldsymbol{T}$. If J is a (classical) universal solution of I w.r.t. $\mathcal{S}$, then* $\mathsf{scert}_\mathcal{S}(I,Q) = Q(J)_\downarrow$.

Proof. It follows from Theorem 4.3 and Theorem 4.13.

We now move to the complexity analysis of the two most important data exchange tasks: deciding whether a supported solution exists, and computing the supported certain answers to a query.

4.3 Complexity Analysis and Results

In data exchange, it is usually assumed that a setting $\mathcal{S}$ does not change over time, and a given query Q is much smaller than a given source instance. Thus, for understanding the complexity of a data exchange problem, it is customary to assume that $\mathcal{S}$ and Q are fixed, and only I is considered in the complexity analysis, i.e., we consider the *data complexity* of the problem. Hence, the problems we are going to discuss will always be parametrized via a setting $\mathcal{S}$, and a query Q (for query answering tasks). The first problem we consider is deciding whether a supported solution exists; $\mathcal{S}$ is a fixed data exchange setting.

PROBLEM :	EXISTS-SSOL($\mathcal{S}$)
INPUT :	A source instance I of $\mathcal{S}$.
QUESTION :	Is $\mathsf{ssol}(I, \mathcal{S}) \neq \emptyset$?

The above problem is very important in data exchange, as one of the main goals is to actually construct a target instance that can be exploited for query answering purposes. Hence, knowing in advance whether at least a supported solution exists is of paramount importance.

Thanks to Item 2 of Theorem 4.12, all the complexity results for checking the existence of a classical solution can be directly transfered to our problem.

Theorem 4.15. *There exists a data exchange setting $\mathcal{S}$ such that* EXISTS-SSOL($\mathcal{S}$) *is undecidable.*

Despite the negative result above, we also inherit positive results from the literature, when focusing on some of the most important data exchange scenarios, known as *weakly-acyclic*. Such settings only allow target TGDs to belong to the language of weakly-acyclic TGDs, which have been first introduced in the seminal paper [37], and is now well-established as the main language for data exchange purposes. We refer to [37] for more details on the definition of weak-acyclicity.

Theorem 4.16. *For every weakly-acyclic data exchange setting $\mathcal{S}$,* EXISTS-SSOL($\mathcal{S}$) *is in* PTIME.

We now move to the second crucial task: computing supported certain answers. Since this problem outputs a set, it is standard to focus on its decision version. For a fixed data exchange setting $\mathcal{S}$ and a fixed query Q, we consider the following decision problem:

PROBLEM :	SCERT($\mathcal{S}, Q$)
INPUT :	A source instance I of $\mathcal{S}$ and a tuple $t \in \mathit{Consts}^{ar(Q)}$.
QUESTION :	Is $t \in \mathsf{scert}_{\mathcal{S}}(I, Q)$?

One can easily show that the above problem is logspace equivalent to the one of computing the supported certain answers.

We start by studying the problem in its full generality, and show that there is a price to pay for query answering with general queries.

Theorem 4.17. *There exists a data exchange setting $\mathcal{S} = \langle \boldsymbol{S}, \boldsymbol{T}, \Sigma_{st}, \Sigma_t \rangle$, with Σ_t having only TGDs, and a query Q over $\boldsymbol{T}$, such that* $\mathsf{SCERT}(\mathcal{S}, Q)$ *is undecidable.*

Although the complexity result above tells us that computing supported certain answers might be infeasible in some settings, we can show that for weakly-acyclic settings, the complexity is more manageable.

Theorem 4.18. *For every weakly-acyclic setting $\mathcal{S}$ and every query Q,* $\mathsf{SCERT}(\mathcal{S}, Q)$ *is in* coNP*, and there exists a weakly-acyclic setting $\mathcal{S}$ that is TGD-only and a query Q such that* $\mathsf{SCERT}(\mathcal{S}, Q)$ *is* coNP-hard*.*

We point out that the above result is in contrast with all the data exchange semantics discussed in the introduction, for which computing certain answers is undecidable, even for weakly-acyclic settings [59, 58].

We conclude this section by recalling that for positive queries, supported certain answers coincide with the classical ones (Theorem 4.13), and computing (classical) certain answers for weakly-acyclic settings, under positive queries, is tractable [37] and can be accomplished via the well-known chase procedure (e.g., see [138]). Hence, the result below follows.

Corollary 4.19. *For every weakly-acyclic setting $\mathcal{S}$ and every positive query Q,* $\mathsf{SCERT}(\mathcal{S}, Q)$ *is in* PTIME*.*

4.4 Summary

We presented a novel notion of supported solutions for data exchange, in the presence of general FO queries. We have shown that supported solutions overcome different limitations of previous approaches: supported certain answers precisely recover the (classical) certain answers when focusing on positive queries, and at the same time, keep the complexity of computing supported certain answers decidable, for weakly-acyclic settings. Since explaining query answering has recently drawn considerably attention under existential rule languages (e.g., see [91, 93, 18, 17, 92, 19]), and knowledge representation in general (e.g., in the context of argumentation [2]) an interesting direction for future work is to address such issue in our setting. Also, it would be interesting to account for user preferences when answering queries, as recently done in [14], possibly considering other ways of expressing preferences, e.g. by means of CP-nets [89, 90].

5

Data Imputation in Fraud Detection

In this chapter, we focus on a possible application of data imputation technique within the detection of fraudulent transactions. Our research on fraudulent transaction detection explores innovative methodologies for identifying and mitigating fraudulent activities within financial systems, enhancing security and trust.

The Importance of Data Imputation in Fraud Detection

The integration of machine learning algorithms has been of remarkable importance in fraud detection cases because they can simply analyze and sort through immense transactional data to identify unusual patterns that are likely to be fraudulent. This also makes it possible to monitor all previous transactions and associated behaviours and predict what will likely happen, i.e., whether fraudulent or non-fraudulent transactions will occur [31]. Regarding fraud, it has been observed that decision trees [123], gradient boosting [133], and neural networks [44], have all shown a reasonable degree of success in flagging potentially fraudulent transactions that sometimes may not be captured in the rule row systems. Nonetheless, there is a restriction in utilizing these machine learning models for fraud detection even when they are more effective in solving numerous tasks. First, although the number of spurious records has been increasing and are now relatively significant, they remain a minor fraction of all transactions, often less than 1%, all other dominant classes on the datasets. This means that the transaction datasets are highly unbalanced and machine learning models tailored for such datasets tend to focus on predicting the dominant class of transactions which are real instead of identifying fraudulent transactions, resulting in a lot of false negatives.

Another crucial problem is missing data and incomplete transactions. Missing data can be attributed to various factors, including systems glitch and incomplete files as well as restrictive privacy issues that arise when certain elements of a transaction cannot be revealed [130]. To overcome this challenges, it is essential to apply missing data imputation. Imputation of missing data refers to the process through

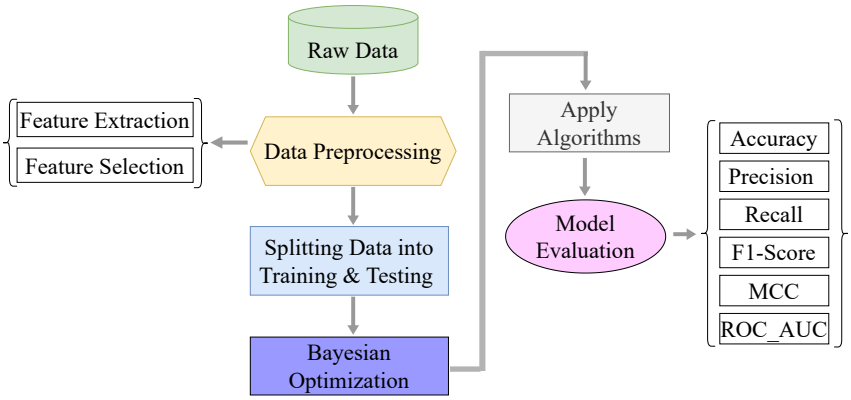


Fig. 5.1: Proposed framework for credit card fraud detection.

which incomplete portions of data sets are filled with some values, which makes it possible for the machine learning models to be trained on a complete dataset.

5.1 The Fraudulent Transaction Detection Problem

In recent years, there has been a significant increase in the volume of financial transactions due to the expansion of financial institutions and the popularity of web-based e-commerce. Fraudulent transactions have become a growing problem in online banking, and fraud detection has always been challenging [100, 40].

Along with credit card development, the pattern of credit card fraud has always been updated. Fraudsters do their best to make it look legitimate, and credit card fraud has always been updated. Fraudsters do their best to make it look legitimate. They try to learn how fraud detection systems work and continue to stimulate these systems, making fraud detection more complicated. Therefore, researchers are constantly trying to find new ways or improve the performance of the existing methods [131].

There are two mechanisms, fraud prevention and fraud detection, that can be exploited to avoid fraud-related losses. Fraud prevention is a proactive method that stops fraud from happening in the first place. On the other hand, fraud detection is needed when a fraudster attempts a fraudulent transaction [79].

Fraud detection in banking is considered a binary classification problem in which data is classified as legitimate or fraudulent [96]. Because banking data is large in volume and with datasets containing a large amount of transaction data, manually reviewing and finding patterns for fraudulent transactions is either impossible or takes a long time. Therefore, machine learning-based algorithms play a pivotal role in fraud detection and prediction [51].

In this paper, we propose an efficient approach for detecting credit card fraud that has been evaluated on publicly available datasets and has used optimized algo-

Variable Name	Description	Type
$V_1, V_2, \ldots, V_{28}$	Transaction feature after PCA transformation	Integer
Time	Seconds elapsed between each transaction with the first transaction	Integer
Amount	Transaction Value	Integer
Class	Legitimate or Fraudulent	0 or 1

Table 5.1: Features of the credit-card fraud dataset.

rithms LightGBM, XGBoost, CatBoost, and logistic regression individually, majority voting combined methods, as well as deep learning and hyperparameter settings as shown in Figure 5.1. An ideal fraud detection system should detect more fraudulent cases, and the precision of detecting fraudulent cases should be high, i.e., all results should be correctly detected, which will lead to the trust of customers in the bank, and on the other hand, the bank will not suffer losses due to incorrect detection.

The main contributions of this paper are summarized as follows:

- We adopt Bayesian optimization for fraud detection and propose to use the weight-tuning hyperparameter to solve the unbalanced data issue as a pre-processing step.
- We propose a majority-voting ensemble learning approach to combine CatBoost, XGBoost, and LightGBM and review the effect of the combined methods on the performance of fraud detection on real, unbalanced data.
- To better cover the unbalanced datasets, we use recall-precision in addition to the typically used ROC-AUC. We also evaluate the performance using F1_score and MCC metrics. According to the results, the proposed methods outperform the existing and based methods. For evaluations, we use publicly available datasets and also publish the source codes [1] with public access to be used by other researchers.

5.2 Proposed Approach

The proposed framework for credit card fraud detection is presented in Figure 5.1. We first apply the pre-processing on the dataset and further split the data into two sections: training and testing, followed by performing Bayesian optimization on the training data to find the best hyper-parameters that lead to the improvement of the performance. We use the cross-validation method to obtain performance comparison in an unbalanced set and then examine the algorithms using different evaluation metrics, including accuracy, precision, recall, the Matthews correlation coefficient (MCC), the F1-score, and AUC diagrams.

we use a real dataset so that the outcome of the proposed algorithm can be used in practice. We consider a dataset named “credit card” that contains 284,807 records

[1] The codes are available at https://github.com/khadijehHashemi/Fraud-Detection-in-Banking-Data-by-Machine-Learning-Techniques

No. of Transactions	No. of legitimate Transactions	No. of fraudulent Transactions	Legitimate (%)	fraudulent (%)
284,807	284,315	492	99.83%	0.17%

Table 5.2: The transaction label distribution in the "credit card" dataset This unbalanced data is expected in real-life datasets.

of two days of transactions made by credit card holders in September 2013. There are 492 fraudulent transactions, and the rest of the transactions are legitimate. The positive class (frauds) accounts for 0.172% of all transactions; hence, the dataset is highly imbalanced. the original features and background information about the data are not given due to confidentiality and privacy considerations. PCA yielded the following principal components: V_1, V_2, V_{28}. The untransformed features with PCA are "time" and "amount." The "Time" column contains the time (in seconds) elapsed between each transaction and the first transaction in the dataset. The feature "Amount" shows the transaction amount. Feature "Class" is the response variable, and it takes the value 1 in case of fraud and 0 otherwise. The summary of the variables and features is presented in table 5.1. The total number of fraudulent transactions are significantly lower than the total number of legitimate transactions, indicating that the data distribution is unbalanced as shown in Table 5.2. This data imbalance causes performance issues in machine learning algorithms, and having a class with the majority of the samples influences the evaluation results[114]. Therefore, in many studies, under-sampling and over-sampling methods are used to solve the data imbalance problem [66]. Using under-sampling methods leads to data loss [163]. Besides, using over-sampling methods leads to the production of duplicate data that doesn't provide information (the data and information are different, and the subject is discussed under the "Entropy"). Some researchers use synthetic minority oversampling (SMOTE) as a solution, which avoids the drawbacks of under and oversampling [110, 134, 3]. However, the SMOTE method causes an increase in the false-positive rate, which is not acceptable in banking for customer orientation. To solve this problem, in this study, we use class weight tuning hyperparameter to solve the mentioned disadvantages [110, 134, 3]. However, the SMOTE method causes an increase in the false-positive rate, which is not acceptable in banking for customer orientation. To solve this problem, in this study, we use a class weight tuning hyperparameter to solve the mentioned disadvantages.

Features extraction and selection

The "time" feature includes the time (in seconds) elapsed between each transaction and the first transaction. The features are unknown except for "Time" and "Amount", and we have no additional information. Feature selection tries to find a subset of features that improve the classifier's performance on effectively detecting credit card fraud [26]. The information gain (IG) method is used to select the most important features that lead to a dimension reduction of the training data. Information gain

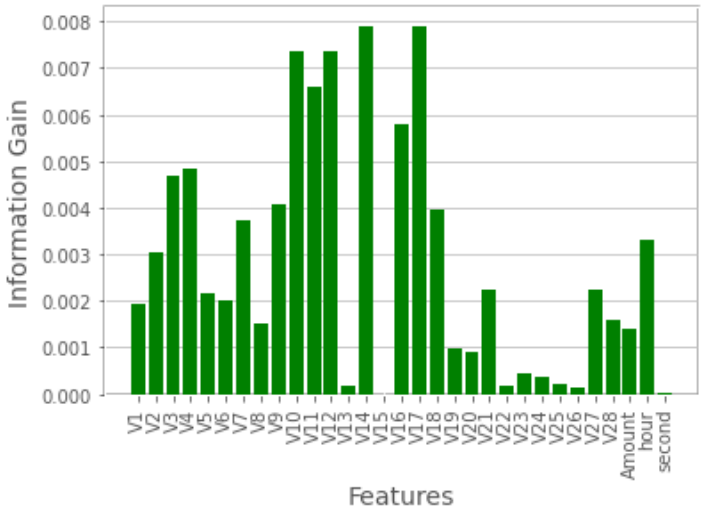


Fig. 5.2: Feature importance diagram.

functions by extracting similarities between credit card transactions and then awarding the greatest weight to the most significant features based on the class of legitimate and fraudulent credit card transactions [113, 134]. Figure 5.2 shows the diagram of the IG, and the top six features extracted by this method have been used to evaluate the proposed algorithm.

Algorithms

Hyperparameters have a significant effect on the performance of machine learning models. We refer to optimization as the process of finding the best set of hyperparameters that configure a machine learning algorithm during its training. In this paper, we use the Bayesian optimization algorithm to tune the hyperparameters that lead to computational time reduction and performance improvement.

Logistic Regression.

This algorithm could not be used for unbalanced data. Therefore, we used hyperparameter class weight to solve the class imbalance prior to applying logistic regression. We show that the ROC-AUC curve cannot be used for the evaluation of unbalanced data and leads to false interpretations.

LightGBM.

The LightGBM algorithm is built on the GBDT framework and aims to improve computational efficiency, particularly on big data prediction problems [86]. The

high-performance LightGBM algorithm can quickly handle large amounts of data, and the distributed processing of data [134]. In LightGBM, the histogram-based algorithm and trees' leaf-wise growth strategy with a maximum depth limit is adopted to increase the training speed and reduce memory consumption. The tuned hyperparameters include the "num_leaves", which is the number of leaves per tree, "max_depth", which denotes the maximum depth of the tree, and "learning_rate" which is also balanced by tuning the weight of the class. With the excessive increase of the leaves, the problem fits horizontally. Therefore, we need to consider a suitable range for this algorithm to obtain good optimization results.

XGBoost.

Extreme Gradient Boosting (XGBoost) has become a dominant algorithm in the field of applied machine learning. This algorithm is a hybrid technique in which new models are added to fix errors caused by existing models. XGBoost includes parallel computation to construct trees using all the CPUs during training. Instead of traditional stopping criteria (i.e., criterion first), it makes use of the "max depth" parameter and starts tree pruning from the backward direction, which significantly improves the computational performance and speed of XGBoost [86]. XGBoost employs a more regularised technique called "formalization" to control over-fitting and achieve better performance [67]. The tuned hyperparameters include learning rate, number of trees, and maximum tree depth, as well as applying weight to classes.

Majority-voting.

Ensemble learning (EL), which is a type of machine learning, combines several classifiers, minimizes the error of the classifiers, and achieves more reasonable results than a single technique. A voting majority classifier is not a real classifier, but a method that is trained and evaluated in parallel in order to use the different features of each algorithm. We can train the data using different hybrid algorithms to predict the final output. The final result of the prediction is determined by a majority of votes according to two different strategies: hard voting and soft voting. If voting is hard, it uses the predicted class labels to vote for the majority law. Otherwise, if the vote is soft, it predicts the class label based on "Argmax," the sum of the predicted probabilities, which is recommended for a set of well-calibrated classifiers. In this case, the probability vector is calculated on average for each predicted class (for all classifiers). The winning class is the one with the highest value [75, 48].

Deep Learning.

Deep learning is shown to be a very promising solution to deal with fraud in financial transactions, making the best use of banks' big data. [119]. In this paper, we use a sequential model, which is a linear stack of layers to construct an artificial neural network model. We use the Relu activation function, and in the last layer, we use "Sigmoid", since our output is binary. The Sigmoid function generates values in a range of zero and one. The function of the Relu activation function is in many ways

Model	Accuracy	AUC	Recall	Precision	F1-score	MCC
Keras	0.9994	0.9401	0.8222	0.8043	0.8132	0.8129

Table 5.3: Deep Learning Model Results

Model	Accuracy	AUC	Recall	Precision	F1-score	MCC
Log_Reg	0.97477	0.9578	0.8730	0.0617	0.1143	0.2248
LGBM	0.99919	0.9472	0.7990	0.7534	0.7699	0.7727
XGB	0.99923	0.9517	0.7949	0.7862	0.7830	0.7864
CatBoost	0.99880	0.9390	0.8096	0.6431	0.7066	0.7158
Vot_Lg, Xg, Ca	0.99924	0.9501	0.8033	0.7720	0.7825	0.7847
Vot_Lg, Xg	0.99927	0.9522	0.8012	0.7901	0.7901	0.7925
Vot_g, Ca	0.99923	0.9492	0.8097	0.7681	0.7823	0.7852
Vot_Lg, Ca	0.99912	0.9459	0.8075	0.7260	0.7581	0.7620

Table 5.4: Performance evaluation of Algorithms

similar to the function of our biological neurons. We use kernel-initializer, which defines the method of determining the random weights of the primary Keras layers. To overcome the unbalanced data problem, we consider the ratio of 1 to 4 for the weight of the majority class to the minority class. This causes an increase in the processing speed as well as increasing the efficiency of the model. The size of the input layer is equal to the number of features plus the extracted features. We also remove the ”time” feature. To build the Keras model, we optimise the number of layers and neurons, the number of epochs, and the batch size, which leads to an increase in speed. Commonly, batch size is set to 32 or 128. However, our dataset is highly unbalanced, and by choosing the common batch size, there may be no fraud cases in the batch during training. Therefore, our range is chosen so that we can see fraudulent samples in each batch. Also, by choosing a larger batch size, the processing is faster, and we also need less memory. Large epoch sizes can result in either over- or under-fitting. Therefore, selecting the appropriate range for optimization not only increases the efficiency of the algorithm but also reduces the time required to find the optimal points. By performing Bayesian optimization, the number of neurons in the first hidden layer is set to 86, the number of epochs is set to 117, and the batch size is set to 1563.

5.3 Experimental Analysis and Results

We use the stratified 5-fold cross-validation method and the boosting algorithms with the Bayesian optimization method to evaluate the performance of the proposed framework. We extract the hyperparameters and evaluate each algorithm individually before using the majority voting method. We examine the algorithms in triple and double precision. The comparison results are presented in Table 5.4.

Model	Accuracy	AUC	Recall	Precision	F1-score
Method presented in [134]	0.984	0.909	0.406	0.973	0.569
Proposed LightGBM	0.9992	0.947	0.799	0.753	0.769
Proposed Approach	0.9993	0.952	0.801	0.79	0.79

Table 5.5: Performance comparison of the proposed approach and the method presented in [134]

Most studies in the literature rely on AUC diagrams to evaluate performance. However, as can be seen from the ROC-AUC curve in Figure 5.3, the value of AUC in severely unbalanced data is not a good evaluation metric. It is influenced by the real positives and considers the negatives irrelevant. According to the ROC-AUC, the logistic regression algorithm 0.9583 has the highest number of fraud detection, but it has the lowest value in other criteria. The precision-recall curve is illustrated in Figure 5.4 and shows the system performance in a more precise manner compared with the ROC-AUC curve. Comparing the precision, recall, and F1-score as well as the MCC, the algorithms used are shown in Figure 5.5. The evaluation results of the proposed approach using different pre-processing and class weight hyperparameter tuning to deal with the problem of data unbalance. In Table 5.4, it is shown that the proposed methods outperform the intelligence method.

5.4 Summary

A number of methods have been proposed by researchers in order to detect and prevent fraudulent credit card transactions. Halvaiee & Akbari develop a new fraud detection model called the AIS-based fraud detection model (AFDM). To improve fraud detection accuracy, they use Immune System Inspired Algorithms (AIRS). According to their paper, their proposed AFDM improves accuracy by up to 25%, reduces costs by up to 85%, and reduces system response time by up to 40% compared to basic algorithms [53].

Randhawa et al., analyze the effectiveness of machine learning algorithms for detecting credit card fraud. To evaluate the available datasets, they used Naive Bayes, stochastic forest and decision trees, neural networks, linear regression (LR), and logistic regression, as well as support vector machine standard models. By applying AdaBoost and majority voting, they propose a hybrid method. Additionally, they add noise to the data samples in order to evaluate robustness. On publicly available datasets, they demonstrate that majority voting is effective at detecting credit card fraud[114].

In [144] the authors propose a group learning framework based on partitioning and clustering of the training set. Their proposed framework has two goals: 1) to ensure the integrity of the sample features, and 2) to solve the high imbalance of the dataset. The main feature of their proposed framework is that every base estimator can be trained in parallel, which improves the effectiveness of their framework.

To detect fraud in credit card transactions, Altyeb et al., propose an intelligent approach consisting of a Bayesian-based hyperparameter optimization algorithm for tuning LightGBM parameters [134]. A publicly available dataset of credit card transactions is used for their experiments. Transactions from both legitimate and fraudulent sources are included in these datasets. Their evaluation results are reported in terms of accuracy, the area under the receiver operating characteristic curve (ROC-AUC), precision, and F1-score metrics.

Verma and Tyagi investigate machine learning algorithms in order to determine the best supervised ML-based algorithm for credit card fraud detection in the presence of an imbalanced dataset. They evaluate five classification techniques and show that the supervised vector classifier and logistic regression classifier outperform other algorithms in an imbalanced dataset [142].

In this chapter, we studied the credit card fraud detection problem in real unbalanced datasets. We proposed a machine-learning approach to improve the performance of fraud detection. Our experimental results showed that the proposed LightGBM method improved the fraud detection cases by 50 percent and the F1-score by 20 percent compared with SOTA. For future studies and work, we propose using other hybrid models as well as working specifically in the field of CatBoost by changing more hyperparameters.

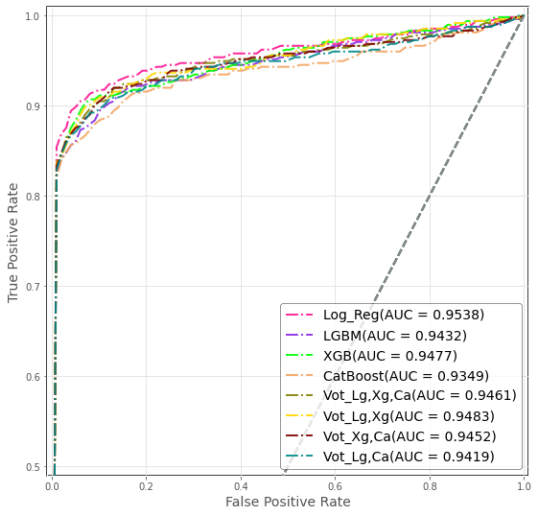


Fig. 5.3: AUC-ROC

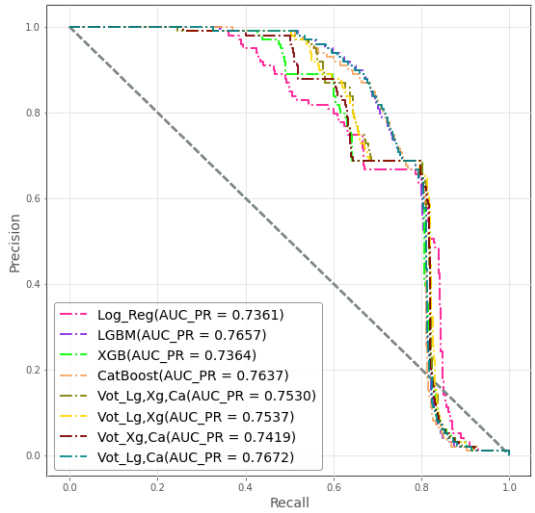


Fig. 5.4: Precision-Recall

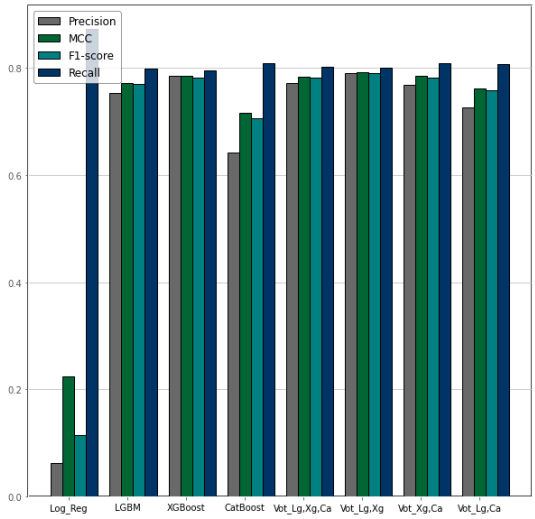


Fig. 5.5: Performance comparing algorithms with different evaluation criteria.

6

Conclusion and Future Work

We explored the problem of missing data and the possible solutions to this problem by introducing *SENtence Transformer-based Imputation* (SENT-I) Algorithm 1, a transformer-based approach for imputing missing values in real-world datasets, particularly those that contain heavy text. By embedding data into a high dimensional semantic space using sentence transformers and utilizing *Facebook AI Similarity Search* (FAISS) (*a vector database to store embeddings*), SENT-I demonstrated its ability to accurately impute missing values even in heavy textual datasets, addressing the limitations of state-of-the-art data imputation method, GRIMP. Through extensive evaluations on diverse real-world datasets, with varying percentage of missing values, SENT-I demonstrated its supremacy. SENT-I achieved an average accuracy improvement of over 40% compared to GRIMP. This highlights the effectiveness, robustness and adaptability of SENT-I, making it an effective solution for enhancing data quality in heavy textual datasets. Furthermore, we introduced *Sentence transformer-based solver for Entity Resolution* SolvER (Algorithm 2). SolvER is the combination of a data imputation technique with any existing ER approach, in our case we use SENT-I for data imputation while Ditto for the ER task. SolvER improve the accuracy of ER in the presence of missing values, further validating the effectiveness of SENT-I (Algorithm 1). We employed SolvER on 3 real-world datasets with varing percentage of nulls. SolvER consistently achieved higher F1-scores compared to the state-of-the-art Ditto. The achieved results emphasize the dual advantage of SENT-I, not only in improving the quality of incomplete datasets and enhancing downstream tasks (i.e., ER) that rely on accurate and complete data.

For future work, we aim to adopt optimization techniques like serialization and summarization for handling long strings, as well as fine-tuning during the construction of the vector database, to improve the efficiency and accuracy of both the imputation and ER processes. Furthermore, to extend SolvER's capabilities, we intend to handle multilingual datasets by leveraging cross-lingual embeddings for accurate entity resolution across multiple languages.

Acknowledgement

I would like to express my deepest gratitude to my supervisors, Prof. Sergio Greco and Prof. Gianvincenzo Alfano. I firmly believe that a crucial aspect of successfully completing a PhD is finding the right supervisors, and I consider myself extremely fortunate in this regard. Their exceptional guidance, support, and expertise have been invaluable throughout this journey, and without them, this work would not have been possible. I would also like to extend my heartfelt thanks to my colleague, Dott. Lucio La Cava, for his collaboration, support, and shared dedication, which have been invaluable throughout this process.

References

[1] S Abiteboul and G Grahne. Update semantics for incomplete information. In *Proc. 11th Int. Conf. Very Large Data Bases, Stockholm*, pages 1–12, 1985.

[2] Gianvincenzo Alfano, Marco Calautti, Sergio Greco, Francesco Parisi, and Irina Trubitsyna. Explainable acceptance in probabilistic abstract argumentation: Complexity and approximation. In *KR*, pages 33–43, 2020.

[3] Doaa Almhaithawi, Assef Jafar, and Mohamad Aljnidi. Example-dependent cost-sensitive credit cards fraud detection using smote and bayes minimum risk. *SN Applied Sciences*, 2(9):1–12, 2020.

[4] Emily Grace Armitage, Joanna Godzien, Vanesa Alonso-Herranz, Ángeles López-Gonzálvez, and Coral Barbas. Missing value imputation strategies for metabolomics data. *Electrophoresis*, 36(24):3050–3060, 2015.

[5] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *International conference on learning representations*, 2017.

[6] Paolo Atzeni and Valeria De Antonellis. *Relational database theory*. Benjamin-Cummings Publishing Co., Inc., 1993.

[7] Ibrahim Berkan Aydilek and Ahmet Arslan. A hybrid method for imputation of missing values using optimized fuzzy c-means with support vector regression and a genetic algorithm. *Information Sciences*, 233:25–35, 2013.

[8] Carlo Batini, Cinzia Cappiello, Chiara Francalanci, and Andrea Maurino. Methodologies for data quality assessment and improvement. *ACM computing surveys (CSUR)*, 41(3):1–52, 2009.

[9] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*, volume 1. MIT press Cambridge, MA, USA, 2017.

[10] Mikhail Bilenko and Raymond J Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 39–48, 2003.

[11] Sarah binti Yusoff and Yap Bee Wah. Comparison of conventional measures of skewness and kurtosis for small sample size. In *2012 International Confer-*

ence on Statistics in Science, Business and Engineering (ICSSBE), pages 1–6. IEEE, 2012.

[12] James H Boyd, Sean M Randall, and Anna M Ferrante. Application of privacy-preserving techniques in operational record linkage centres. *Medical data privacy handbook*, pages 267–287, 2015.

[13] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[14] Marco Calautti, Sergio Greco, Cristian Molinaro, and Irina Trubitsyna. Preference-based inconsistency-tolerant query answering under existential rules. *Artif. Intell.*, 312:103772, 2022.

[15] Luca Cappelletti, Tommaso Fontana, Guido Walter Di Donato, Lorenzo Di Tucci, Elena Casiraghi, and Giorgio Valentini. Complex data imputation by auto-encoders and convolutional neural networks—a case study on genome gap-filling. *Computers*, 9(2):37, 2020.

[16] Riccardo Cappuzzo, Saravanan Thirumuruganathan, and Paolo Papotti. Relational data imputation with graph neural networks. In *Proceedings 27th International Conference on Extending Database Technology, EDBT 2024, Paestum, Italy, March 25 - March 28*.

[17] İsmail İlkan Ceylan, Thomas Lukasiewicz, Enrico Malizia, Cristian Molinaro, and Andrius Vaicenavicius. Explanations for negative query answers under existential rules. In *Proc. KR*, pages 223–232, 2020.

[18] İsmail İlkan Ceylan, Thomas Lukasiewicz, Enrico Malizia, Cristian Molinaro, and Andrius Vaicenavicius. Preferred explanations for ontology-mediated queries under existential rules. In *Proc. AAAI*, pages 6262–6270, 2021.

[19] İsmail İlkan Ceylan, Thomas Lukasiewicz, Enrico Malizia, and Andrius Vaicenavicius. Explanations for query answers under existential rules. In *Proc. IJCAI*, pages 1639–1646, 2019.

[20] Qun Chen, Zhaoqiang Chen, Youcef Nafa, Tianyi Duan, Wei Pan, Lijun Zhang, and Zhanhuai Li. Adaptive deep learning for entity resolution by risk analysis. *Knowledge-Based Systems*, 260:110118, 2023.

[21] Ching-Hsue Cheng, Chia-Pang Chan, and Yu-Jheng Sheu. A novel purity-based k nearest neighbors imputation method and its application in financial distress prediction. *Engineering Applications of Artificial Intelligence*, 81:283–299, 2019.

[22] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

[23] Arman Cohan, Iz Beltagy, Daniel King, Bhavana Dalvi, and Daniel S Weld. Pretrained language models for sequential sentence classification. *arXiv preprint arXiv:1909.04054*, 2019.

[24] William W Cohen and Jacob Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 475–480, 2002.

[25] Adriana Fonseca Costa, Miriam Seoane Santos, Jastin Pompeu Soares, and Pedro Henriques Abreu. Missing data imputation via denoising autoencoders: the untold story. In *Advances in Intelligent Data Analysis XVII: 17th International Symposium, IDA 2018,'s-Hertogenbosch, The Netherlands, October 24–26, 2018, Proceedings 17*, pages 87–98. Springer, 2018.

[26] Jipeng Cui, Chungang Yan, and Cheng Wang. Learning transaction cohesiveness for online payment fraud detection. In *The 2nd International Conference on Computing and Data Science*, pages 1–5, 2021.

[27] Nilesh Dalvi, Vibhor Rastogi, Anirban Dasgupta, Anish Das Sarma, and Tamás Sarlós. Optimal hashing schemes for entity matching. In *Proceedings of the 22nd international conference on world wide web*, pages 295–306, 2013.

[28] Tamraparni Dasu. Data glitches: Monsters in your data. In *Handbook of data quality: Research and practice*, pages 163–178. Springer, 2013.

[29] Tamraparni Dasu, Ji Meng Loh, and Divesh Srivastava. Empirical glitch explanations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 572–581, 2014.

[30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[31] Sahil Dhankhad, Emad Mohammed, and Behrouz Far. Supervised machine learning algorithms for credit card fraudulent transaction detection: a comparative study. In *2018 IEEE international conference on information reuse and integration (IRI)*, pages 122–125. IEEE, 2018.

[32] David P Doane and Lori E Seward. Measuring skewness: a forgotten statistic? *Journal of statistics education*, 19(2), 2011.

[33] A Rogier T Donders, Geert JMG Van Der Heijden, Theo Stijnen, and Karel GM Moons. A gentle introduction to imputation of missing values. *Journal of clinical epidemiology*, 59(10):1087–1091, 2006.

[34] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *CoRR*, abs/2401.08281, 2024.

[35] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. Deeper–deep entity resolution. *arXiv preprint arXiv:1710.00597*, 2017.

[36] Ahmed Elmagarmid, Ihab F Ilyas, Mourad Ouzzani, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, and Si Yin. Nadeef/er: generic and interactive entity resolution. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1071–1074, 2014.

[37] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: semantics and query answering. *TCS*, 336(1):89–124, 2005.

[38] Ivan P Fellegi and Alan B Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[39] Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. Language-agnostic bert sentence embedding. *arXiv preprint arXiv:2007.01852*, 2020.

[40] Haonan Feng. Ensemble learning in credit card fraud detection using boosting methods. In *2021 2nd International Conference on Computing and Data Science (CDS)*, pages 7–11. IEEE, 2021.

[41] Nazanin Fouladgar and Kary Främling. A novel lstm for multivariate time series with massive missingness. *Sensors*, 20(10):2832, 2020.

[42] Ray FOWLER, Ahamed H IDRIS, Christoph U LEHMANN, and Samuel A MCDONALD. Improving emergency medical services information exchange: Methods for automating entity resolution. *Accident and Emergency Informatics*, 291:17, 2022.

[43] Ariel Fuxman, Phokion G Kolaitis, Renée J Miller, and Wang-Chiew Tan. Peer data exchange. *ACM Transactions on Database Systems (TODS)*, 31(4):1454–1498, 2006.

[44] Sushmito Ghosh and Douglas L Reilly. Credit card fraud detection with a neural-network. In *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*, volume 3, pages 621–630. IEEE, 1994.

[45] Chaitanya Gokhale, Sanjib Das, AnHai Doan, Jeffrey F Naughton, Narasimhan Rampalli, Jude Shavlik, and Xiaojin Zhu. Corleone: Hands-off crowdsourcing for entity matching. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 601–612, 2014.

[46] Lovedeep Gondara and Ke Wang. Mida: Multiple imputation using denoising autoencoders. In *Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part III 22*, pages 260–272. Springer, 2018.

[47] David Gordon, Panayiotis Petousis, Henry Zheng, Davina Zamanzadeh, and Alex AT Bui. Tsi-gnn: extending graph neural networks to handle missing data in temporal settings. *Frontiers in big Data*, 4:693869, 2021.

[48] Anil Goyal and Jihed Khiari. Diversity-aware weighted majority vote classifier for imbalanced data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[49] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[50] Gösta Grahne. Dependency satisfaction in databases with incomplete information. In *Proc. Very Large Data Bases (VLDB) Conference*, pages 37–45, 1984.

[51] Kavya Gupta, Kirtivardhan Singh, Gaurav Vikram Singh, Mohd. Hassan, Garg Himani, and Upasana Sharma. Machine learning based credit card fraud detection - a review. In *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, pages 362–368, 2022.

[52] Tanya Gupta and Varad Deshpande. Entity resolution for maintaining electronic medical record using oyster. In *EAI International Conference on Big Data Innovation for Sustainable Cognitive Computing: BDCC 2018*, pages 41–50. Springer, 2020.

[53] Neda Soltani Halvaiee and Mohammad Kazem Akbari. A novel model for credit card fraud detection using artificial immune systems. *Applied soft computing*, 24:40–49, 2014.

[54] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[55] Yikun Han, Chunjiang Liu, and Pengfei Wang. A comprehensive survey on vector database: Storage and retrieval technique, challenge. *CoRR*, abs/2310.11703, 2023.

[56] Jan L Harrington. *Relational database design and implementation*. Morgan Kaufmann, 2016.

[57] Yuan He, Zhangdie Yuan, Jiaoyan Chen, and Ian Horrocks. Language models as hierarchy encoders. *arXiv preprint arXiv:2401.11374*, 2024.

[58] Andre Hernich. Answering Non-Monotonic Queries in Relational Data Exchange. *LMCS*, Volume 7, Issue 3, 2011.

[59] André Hernich, Leonid Libkin, and Nicole Schweikardt. Closed world data exchange. *TODS*, 36(2):14:1–14:40, 2011.

[60] T. Hori, D. Montcho, C. Agbangla, K. Ebana, K. Futakuchi, and H. Iwata. Multi-task gaussian process for imputing missing data in multi-trait and multi-environment trials. *Theor. Appl. Genet*, 129(11):pp. 2101–2115, Nov. 2016.

[61] Badr Hssina, Abdelkarim Merbouha, Hanane Ezzikouri, and Mohammed Erritali. A comparative study of decision tree id3 and c4. 5. *International Journal of Advanced Computer Science and Applications*, 4(2):13–19, 2014.

[62] Jiacheng Huang, Wei Hu, Zhifeng Bao, Qijin Chen, and Yuzhong Qu. Deep entity matching with adversarial active learning. *The VLDB Journal*, 32(1):229–255, 2023.

[63] Farhad Imani, Changqing Cheng, Ruimin Chen, and Hui Yang. Nested gaussian process modeling and imputation of high-dimensional incomplete data under uncertainty. *IISE Transactions on Healthcare Systems Engineering*, 9(4):315–326, 2019.

[64] Tomasz Imieliński and Witold Lipski Jr. Incomplete information in relational databases. *Journal of the ACM (JACM)*, 31(4):761–791, 1984.

[65] Katie Irvine, Rick Hall, and Lee Taylor. A profile of the centre for health record linkage. *International journal of population data science*, 4(2), 2022.

[66] Fayaz Itoo, Mittal Meenakshi, and Satwinder Singh. Comparison and analysis of logistic regression, naive bayes and knn machine learning algorithms for credit card fraud detection. *International Journal of Information Technology*, 13(4):1503–1511, 2021.

[67] Sami Ben Jabeur, Cheima Gharib, Salma Mefteh-Wali, and Wissal Ben Arfi. Catboost model and artificial intelligence techniques for corporate failure prediction. *Technological Forecasting and Social Change*, 166:120658, 2021.

[68] Anil Jadhav, Dhanya Pramod, and Krishnan Ramanathan. Comparison of performance of data imputation methods for numeric dataset. *Applied Artificial Intelligence*, 33(10):913–933, 2019.

[69] Sebastian Jäger, Arndt Allhorn, and Felix Bießmann. A benchmark for data imputation methods. *Frontiers in big Data*, 4:693674, 2021.

[70] Myoungshic Jhun, Hyeong Chul Jeong, and Ja-Yong Koo. On the use of adaptive nearest neighbors for missing value imputation. *Communications in Statistics—Simulation and Computation®*, 36(6):1275–1286, 2007.

[71] Huang Jianglin, Federica Sarro Jacky Wai Keung, Yan-Fu Li, Yuen-Tak Yu, W. K. Chan, and Hongyi Sun. Cross-validation based k nearest neighbor imputation for software quality datasets: an empirical study. *Journal of Systems and Software*, 2017.

[72] Zhi Jing, Yongye Su, Yikun Han, Bo Yuan, Haiyun Xu, Chunjiang Liu, Kehai Chen, and Min Zhang. When large language models meet vector databases: a survey. *arXiv preprint arXiv:2402.01763*, 2024.

[73] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Trans. Big Data*, 7(3):535–547, 2021.

[74] Paris C Kanellakis. Elements of relational database theory. In *Formal models and semantics*, pages 1073–1156. Elsevier, 1990.

[75] Fairoz Nower Khan, Amit Hasan Khan, and Lamiah Israt. Credit card fraud prediction and classification using deep neural network and ensemble learning. In *2020 IEEE Region 10 Symposium (TENSYMP)*, pages 114–119. IEEE, 2020.

[76] Pradap Venkatramanan Konda. *Magellan: Toward building entity matching management systems*. The University of Wisconsin-Madison, 2018.

[77] Nihel Kooli, Robin Allesiardo, and Erwan Pigneul. Deep learning based approach for entity resolution in databases. In *Asian conference on intelligent information and database systems*, pages 3–12. Springer, 2018.

[78] Hanna Köpcke, Andreas Thor, and Erhard Rahm. Evaluation of entity resolution approaches on real-world match problems. *Proceedings of the VLDB Endowment*, 3(1-2):484–493, 2010.

[79] Nishamathi Kumaraswamy, Mia K Markey, Tahir Ekin, Jamie C Barner, and Karen Rascati. Healthcare fraud data mining methods: A look back and look ahead. *Perspectives in Health Information Management*, 19(1), 2022.

[80] Tan Duy Le, Razvan Beuran, and Yasuo Tan. Comparison of the most influential missing data imputation algorithms for healthcare. In *2018 10th international conference on knowledge and systems engineering (KSE)*, pages 247–251. IEEE, 2018.

[81] Bing Li, Yukai Miao, Yaoshu Wang, Yifang Sun, and Wei Wang. Improving the efficiency and effectiveness for bert-based entity resolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13226–13233, 2021.

[82] Bo-Han Li, Yi Liu, An-Man Zhang, Wen-Huan Wang, and Shuo Wan. A survey on blocking technology of entity resolution. *Journal of Computer Science and Technology*, 35:769–793, 2020.

[83] Dong Li, Linhao Li, Xianling Li, Zhiwu Ke, and Qinghua Hu. Smoothed lstm-ae: A spatio-temporal deep model for multiple time-series missing imputation. *Neurocomputing*, 411:351–363, 2020.

[84] Linchao Li, Jian Zhang, Yonggang Wang, and Bin Ran. Missing value imputation for traffic-related time series data based on a multi-view learning method. *IEEE Transactions on Intelligent Transportation Systems*, 20(8):2933–2943, 2018.

[85] Yuliang Li, Jinfeng Li, Yoshi Suhara, AnHai Doan, and Wang-Chiew Tan. Effective entity matching with transformers. *The VLDB Journal*, 32(6):1215–1235, 2023.

[86] Weizhang Liang, Suizhi Luo, Guoyan Zhao, and Hao Wu. Predicting hard rock pillar stability using gbdt, xgboost, and lightgbm algorithms. *Mathematics*, 8(5):765, 2020.

[87] Leonid Libkin and Cristina Sirangelo. Data exchange and schema mappings in open and closed worlds. *JCSS*, 77(3):542–571, 2011.

[88] Xin Liu, Xiaochen Lai, and Liyong Zhang. A hierarchical missing value imputation method by correlation-based k-nearest neighbors. In *Intelligent Systems and Applications: Proceedings of the 2019 Intelligent Systems Conference (IntelliSys) Volume 1*, pages 486–496. Springer, 2020.

[89] Thomas Lukasiewicz and Enrico Malizia. Complexity results for preference aggregation over (*m*)cp-nets: Pareto and majority voting. *Artif. Intell.*, 272:101–142, 2019.

[90] Thomas Lukasiewicz and Enrico Malizia. Complexity results for preference aggregation over (*m*)cp-nets: Max and rank voting. *Artif. Intell.*, 303:103636, 2022.

[91] Thomas Lukasiewicz, Enrico Malizia, Maria Vanina Martinez, Cristian Molinaro, Andreas Pieris, and Gerardo Simari. Inconsistency-tolerant query answering for existential rules. *Artif. Intell.*, 307:103685, 2022.

[92] Thomas Lukasiewicz, Enrico Malizia, and Cristian Molinaro. Explanations for inconsistency-tolerant query answering under existential rules. In *Proc. AAAI*, pages 2909–2916, 2020.

[93] Thomas Lukasiewicz, Enrico Malizia, and Cristian Molinaro. Explanations for negative query answers under inconsistency-tolerant semantics. In *Proc. IJCAI*, pages 2705–2711, 2022.

[94] Jun Ma, Jack CP Cheng, Feifeng Jiang, Weiwei Chen, Mingzhu Wang, and Chong Zhai. A bi-directional missing data imputation scheme based on lstm and transfer learning for building energy data. *Energy and Buildings*, 216:109941, 2020.

[95] Oded Maimon and Lior Rokach. *Data mining and knowledge discovery handbook*, volume 2. Springer, 2005.

[96] Esraa Faisal Malik, Khai Wah Khaw, Bahari Belaton, Wai Peng Wong, and XinYing Chew. Credit card fraud detection using a new hybrid machine learning architecture. *Mathematics*, 10(9):1480, 2022.

[97] Adam Marcus, Eugene Wu, David Karger, Samuel Madden, and Robert Miller. Human-powered sorts and joins. *arXiv preprint arXiv:1109.6881*, 2011.

[98] Mehdi Mirza. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[99] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 international conference on management of data*, pages 19–34, 2018.

[100] Jay Nanduri, Yung-Wen Liu, Kiyoung Yang, and Yuting Jia. Ecommerce fraud detection through fraud islands and multi-layer machine learning model. In *Future of Information and Communication Conference*, pages 556–570. Advances in Information and Communication, Springer, San Francisco, USA, 2020.

[101] Fulufhelo V Nelwamondo, Shakir Mohamed, and Tshilidzi Marwala. Missing data: A comparison of neural network and expectation maximization techniques. *Current Science*, pages 1514–1521, 2007.

[102] Daniel A Newman. Missing data: Five practical guidelines. *Organizational research methods*, 17(4):372–411, 2014.

[103] Matteo Paganelli, Donato Tiano, Francesco Del Buono, Andrea Baraldi, Riccardo Benassi, Giacomo Guiduzzi, Francesco Guerra, et al. How transformers are revolutionizing entity matching. In *CEUR WORKSHOP PROCEEDINGS*, volume 3741, pages 662–670. CEUR-WS, 2024.

[104] George Papadakis, Ekaterini Ioannou, and Themis Palpanas. Entity resolution: Past, present and yet-to-come. In *Proceedings of the 23rd International Conference on Extending Database Technology, EDBT 2020, Copenhagen, Denmark, March 30 - April 02, 2020*, pages 647–650, 2020.

[105] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. Blocking and filtering techniques for entity resolution: A survey. *ACM Computing Surveys (CSUR)*, 53(2):1–42, 2020.

[106] Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, 2016.

[107] Norman W Paton, M Howard Williams, Kosmas Dietrich, Olive Liew, Andrew Dinn, and Alan Patrick. Vespa: A benchmark for vector spatial databases. In *Advances in Databases: 17th British National Conference on Databases, BNCOD 17 Exeter, UK, July 3–5, 2000 Proceedings 17*, pages 81–101. Springer, 2000.

[108] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[109] Simon JD Prince. *Understanding Deep Learning*. MIT press, 2023.

[110] Maja Puh and Ljiljana Brkić. Detecting credit card fraud using selected machine learning algorithms. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1250–1255. IEEE, 2019.

[111] Yeping Lina Qiu, Hong Zheng, and Olivier Gevaert. Genomic data imputation with variational auto-encoders. *GigaScience*, 9(8):giaa082, 2020.

[112] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[113] Morteza Rakhshaninejad, Mohammad Fathian, Babak Amiri, and Navid Yazdanjue. An ensemble-based credit card fraud detection algorithm using an efficient voting strategy. *The Computer Journal*, 2021.

[114] Kuldeep Randhawa, Chu Kiong Loo, Manjeevan Seera, Chee Peng Lim, and Asoke K Nandi. Credit card fraud detection using adaboost and majority voting. *IEEE access*, 6:14277–14284, 2018.

[115] N Reimers. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[116] Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2020.

[117] Filipe Rodrigues, Kristian Henrickson, and Francisco C Pereira. Multi-output gaussian processes for crowdsourced traffic data imputation. *IEEE Transactions on Intelligent Transportation Systems*, 20(2):594–603, 2018.

[118] Abhimanyu Roy, Jingyi Sun, Robert Mahoney, Loreto Alonzi, Stephen Adams, and Peter Beling. Deep learning detecting fraud in credit card transactions. In *2018 systems and information engineering design symposium (SIEDS)*, pages 129–134. IEEE, 2018.

[119] Abhimanyu Roy, Jingyi Sun, Robert Mahoney, Loreto Alonzi, Stephen Adams, and Peter Beling. Deep learning detecting fraud in credit card transactions. In *2018 Systems and Information Engineering Design Symposium (SIEDS)*, pages 129–134. IEEE, 2018.

[120] Donald B Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

[121] Zahriah Sahri, Rubiyah Yusof, and Junzo Watada. FINNIM: iterative imputation of missing values in dissolved gas analysis dataset. *IEEE Trans. Ind. Informatics*, 10.

[122] Sunita Sarawagi and Anuradha Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–278, 2002.

[123] Prajal Save, Pranali Tiwarekar, Ketan N Jain, and Neha Mahyavanshi. A novel idea for credit card fraud detection using decision tree. *International Journal of Computer Applications*, 161(13), 2017.

[124] Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann, and Andreas Grafberger. Automating large-scale data quality verification. *Proceedings of the VLDB Endowment*, 11(12):1781–1794, 2018.

[125] Reza Shahbazian and Sergio Greco. Generative adversarial networks assist missing data imputation: A comprehensive survey & evaluation. *IEEE Access*, 2023.

[126] Ashok K Singh, Laxmi P Gewali, and Jiwan Khatiwada. New measures of skewness of a probability distribution. *Open Journal of Statistics*, 9(5):601–621, 2019.

[127] Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. Synthesizing entity matching rules by examples. *Proceedings of the VLDB Endowment*, 11(2):189–202, 2017.

[128] TR Sivapriya, ANB Kamal, and V Thavavel. Imputation and classification of missing data using least square support vector machines–a new approach in dementia diagnosis. *Int. J. Adv. Res. Artif. Intell*, 1(4):29–33, 2012.

[129] Marek Smieja, Maciej Kolomycki, Lukasz Struski, Mateusz Juda, and Mario AT Figueiredo. Iterative imputation of missing data using auto-encoder dynamics. In *International Conference on Neural Information Processing*, pages 258–269. Springer, 2020.

[130] Wei Wen Soh and Rika Mohd Yusuf. Predicting credit card fraud on a imbalanced data. *International Journal of Data Science and Advanced Analytics*, 1(1):12–17, 2019.

[131] Mohammad Soltani Delgosha, Nastaran Hajiheydari, and Sayed Mahmood Fahimi. Elucidation of big data analytics in banking: a four-stage delphi study. *Journal of Enterprise Information Management*, 34(6):1577 – 1596, 2020.

[132] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. Missing data imputation with adversarially-trained graph convolutional networks. *Neural Networks*, 129:249–260, 2020.

[133] Altyeb Altaher Taha and Sharaf Jameel Malebary. An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine. *IEEE access*, 8:25579–25587, 2020.

[134] Altyeb Altaher Taha and Sharaf Jameel Malebary. An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine. *IEEE Access*, 8:25579–25587, 2020.

[135] Divyanshu Talwar, Aanchal Mongia, Debarka Sengupta, and Angshul Majumdar. Autoimpute: Autoencoder based imputation of single-cell rna-seq data. *Scientific reports*, 8(1):16329, 2018.

[136] Fei Tang and Hemant Ishwaran. Random forest missing data algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 10(6):363–377, 2017.

[137] Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. Augmented SBERT: data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. *CoRR*, abs/2010.08240, 2020.

[138] Efthymia Tsamoura, David Carral, Enrico Malizia, and Jacopo Urbani. Materializing knowledge bases via trigger graphs. *Proc. VLDB Endow.*, 14(6):943–956, 2021.

[139] Bhekisipho Twala. An empirical comparison of techniques for handling incomplete data using decision trees. *Applied Artificial Intelligence*, 23(5):373–405, 2009.

[140] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[141] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[142] Pradeep Verma and Poornima Tyagi. Analysis of supervised machine learning algorithms in the context of fraud detection. *ECS Transactions*, 107(1):7189, 2022.

[143] Aiguo Wang, Jing Yang, and Ning An. Regularized sparse modelling for microarray missing value estimation. *IEEE Access*, 9:16899–16913.

[144] Hongyu Wang, Ping Zhu, Xueqiang Zou, and Sujuan Qin. An ensemble learning framework for credit card fraud detection based on training set partitioning and clustering. In *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 94–98. IEEE, 2018.

[145] Jianguo Wang, Eric Hanson, Guoliang Li, Yannis Papakonstantinou, Harsha Simhadri, and Charles Xie. Vector databases: What's really new and what's next?(vldb 2024 panel). *Proceedings of the VLDB Endowment*, 17(12):4505–4506, 2024.

[146] Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. Crowder: Crowdsourcing entity resolution. *arXiv preprint arXiv:1208.1927*, 2012.

[147] Jiannan Wang, Guoliang Li, Jeffrey Xu Yu, and Jianhua Feng. Entity matching: How similar is similar. *Proceedings of the VLDB Endowment*, 4(10):622–633, 2011.

[148] Shuohang Wang and Jing Jiang. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*, 2016.

[149] William E Winkler. Matching and record linkage. *Wiley interdisciplinary reviews: Computational statistics*, 6(5):313–325, 2014.

[150] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5):1–37, 2022.

[151] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[152] Xingrui Xie, Han Liu, Wenzhe Hou, and Hongbin Huang. A brief survey of vector databases. In *2023 9th International Conference on Big Data and Information Analytics (BigDIA)*, pages 364–371. IEEE, 2023.

[153] Chen Yin and Zixuan Zhang. A study of sentence similarity based on the all-minilm-l6-v2 model with "same semantics, different structure" after fine tuning. In *2024 2nd International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI 2024)*, pages 677–684. Atlantis Press, 2024.

[154] Jiaxuan You, Xiaobai Ma, Yi Ding, Mykel J Kochenderfer, and Jure Leskovec. Handling missing data with graph representation learning. *Advances in Neural Information Processing Systems*, 33:19075–19087, 2020.

[155] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[156] Hongwu Yuan, Guoming Xu, Zijian Yao, Ji Jia, and Yiwen Zhang. Imputation of missing data in time series for air pollutants using long short-term memory recurrent neural networks. In *Proceedings of the 2018 ACM international joint conference and 2018 international symposium on pervasive and ubiquitous computing and wearable computers*, pages 1293–1300, 2018.

[157] Rui Zhang, Bayu Distiawan Trisedya, Miao Li, Yong Jiang, and Jianzhong Qi. A benchmark and comprehensive survey on knowledge graph entity alignment via representation learning. *VLDB J.*, 31(5):1143–1168, 2022.

[158] Zhongheng Zhang. Missing data imputation: focusing on single imputation. *Annals of translational medicine*, 4(1), 2016.

[159] Xuandong Zhao, Siqi Ouyang, Zhiguo Yu, Ming Wu, and Lei Li. Pre-trained language models can be fully zero-shot learners. *arXiv preprint arXiv:2212.06950*, 2022.

[160] Xiantong Zhen, Mengyang Yu, Xiaofei He, and Shuo Li. Multi-target regression via robust low-rank learning. *IEEE transactions on pattern analysis and machine intelligence*, 40(2):497–504, 2017.

[161] Jia Zhu, Changqin Huang, and Pasquale De Meo. DFMKE: a dual fusion multi-modal knowledge graph embedding framework for entity alignment. *Inf. Fusion*, 90:111–119, 2023.

[162] Ming Zhu and Xingbing Cheng. Iterative knn imputation based on gra for missing values in tplms. In *2015 4th international conference on computer science and network technology (ICCSNT)*, volume 1, pages 94–99. IEEE, 2015.

[163] Junyi Zou, Jinliang Zhang, and Ping Jiang. Credit card fraud detection using autoencoder neural network. *arXiv preprint arXiv:1908.11553*, 2019.

List of Figures

List of Tables