Chapter 8
Context-Based Entity Matching for Big Data

Mayesha Tasnim[1], Diego Collarana[1], Damien Graux[2], and Maria-Esther Vidal[3](✉)

[1] Fraunhofer IAIS, Sankt Augustin, Germany
[2] ADAPT SFI Research Centre, Trinity College, Dublin, Ireland
[3] TIB Leibniz Information Centre For Science and Technology, Hannover, Germany
maria.vidal@tib.eu

Abstract. In the Big Data era, where variety is the most dominant dimension, the RDF data model enables the creation and integration of actionable knowledge from heterogeneous data sources. However, the RDF data model allows for describing entities under various contexts, e.g., people can be described from its demographic context, but as well from their professional contexts. Context-aware description poses challenges during entity matching of RDF datasets—the match might not be valid in every context. To perform a contextually relevant entity matching, the specific context under which a data-driven task, e.g., data integration is performed, must be taken into account. However, existing approaches only consider inter-schema and properties mapping of different data sources and prevent users from selecting contexts and conditions during a data integration process. We devise COMET, an entity matching technique that relies on both the knowledge stated in RDF vocabularies and a context-based similarity metric to map contextually equivalent RDF graphs. COMET follows a two-fold approach to solve the problem of entity matching in RDF graphs in a context-aware manner. In the first step, COMET computes the similarity measures across RDF entities and resorts to the Formal Concept Analysis algorithm to map contextually equivalent RDF entities. Finally, COMET combines the results of the first step and executes a 1-1 perfect matching algorithm for matching RDF entities based on the combined scores. We empirically evaluate the performance of COMET on testbed from DBpedia. The experimental results suggest that COMET accurately matches equivalent RDF graphs in a context-dependent manner.

1 Introduction

In the Big Data era, variety is one of the most dominant dimensions bringing new challenges for data-driven tasks. Variety alludes to the types and sources of data that are becoming increasingly heterogeneous with new forms of data

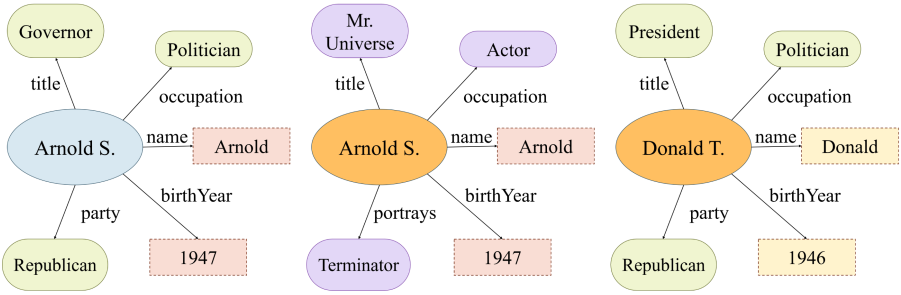
V. Janev et al. (Eds.): Knowledge Graphs and Big Data Processing, LNCS 12072, pp. 122–146, 2020.
https://doi.org/10.1007/978-3-030-53199-7_8

collection being introduced with time. At one point in time, the only source of digital data was spreadsheets and databases. Today data is collected from emails, photographs, digital documents, or audio. The variety of unstructured and semi-structured data creates issues during data analysis. Therefore, these varying forms of data must be integrated for consistency in storage, mining, and analysis. The process of integrating these complex and semi-structured data poses its own set of challenges. For example, the same real-world object may be represented in different data sources as different entities; it therefore challenging to identify entities that refer to the same real-world object.

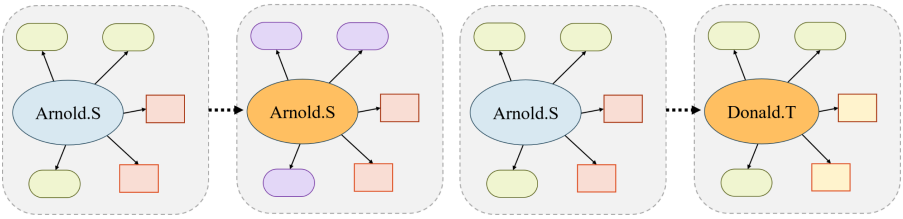
The Resource Description Framework (RDF) data model enables the description of data integrated from heterogeneous data sources. RDF is designed to have a simple data model with formal semantics to provide inference capabilities. The syntax of RDF describes a simple graph-based data model, along with formal semantics, which allows for well-defined entailment regimes that provide the basis for logical deductions. RDF has the following principal use cases as a method for describing web metadata: (i) to allow applications to use an information model which is open rather than constrained; (ii) to allow web data to be machine-processable; and (iii) to combine and integrate data from several sources incrementally. RDF is designed to represent information in a minimally constraining and flexible way; it can be used in isolated applications, where individually designed formats might be easily understood, and the RDF generality offers higher value from sharing. Thus, the value of RDF data increases as it becomes accessible to more applications across the entire internet.

RDF is a semi-structured data model that allows for the encoding of multiple contexts of an entity within the same graph. A context describes a situation that limits the validity of particular information. The so-called "Context as a Box" approach [63] considers context as the conditions and constraints which define whether or not a piece of information is accurate. Contextual information (or meta information) represents the conditions and constraints which describe the situation of a context. For example, the fact "Donald Trump is the President of the United States of America" is valid only in the context of "the presidential period between the years 2017 and 2021". The RDF data model allows for representing entities of the same type with different properties. This in turn allows for the encoding of multiple contexts of an entity within the same graph. For example, the entity *Donald Trump* in an RDF graph can have properties relating to the context of his career as a politician, and also the ones that describe his role as a reality TV celebrity. This feature of RDF is useful for addressing the data complexity challenge of variety– a dominant dimension of data in the Big Data era [218]. Nevertheless, enabling diverse representations of the same entity poses new challenges during the analysis of RDF graphs. This is particularly prevalent in cases where specific contexts need to be considered for the effective identification of similar entities [35]. Two entities may be similar in one context but dissimilar in another. In this chapter[1], we present a novel approach to tackle

[1] This chapter is based on the master thesis of Mayesha Tasnim.



(a) Arnold S.'s entity in D_1 (b) Arnold S.'s entity in D_2 (c) Donald T.'s entity in D_2



(d) Entity matching using similarity (e) Entity matching using similarity and context

Fig. 1. Motivation Example. The top row shows three entities across two datasets. The bottom row shows two matching scenarios, the left one not considering context during entity matching, and the right one taking context into consideration.

the problem of entity matching considering context as a new dimension of the matching algorithm.

1.1 Motivating Example

Following the principle of the web of linked data, RDF allows for the representation of different contexts of an entity within the same graph. This means that applications attempting to match entities from different graphs have to deal with entities that are valid in different contexts. In order to match entities in such a way that they comply with the context specified by the user of the application, the system context must be taken into account. A system's context represents any kind of information that describes the system and its requirements. If this system context is not considered, the entity matching operation will match entities that are not relevant or valid under the definition of system context.

This can be demonstrated using the example of a context-based entity matching scenario using RDF entities representing persons. *Arnold Schwarzenegger* is a person with an extensive career in both politics and acting. Consequently, there is data available regarding both his career in politics and his achievements in the movie industry. Consider a system that contains data about American politicians and is searching other data sources to match relevant data. The system's

dataset D_1 contains information about Arnold Schwarzenegger and his political career. In another dataset D_2 available on the web there exists information about Arnold's acting career, e.g. the movies he has acted in and the roles he has played. The same dataset D_2 also contains information about other celebrities, like Donald Trump, President of the United States. These entities are presented in Figs. 1a, 1b and 1c, respectively.

In a typical entity matching scenario where context is not considered, entities are matched to the ones that are most similar to them. In such a case, Arnold Schwarzenegger's entity from D_1 will be matched with the entity in D_2 containing information about his acting career, as shown in Fig. 1d. However, in the context of politics, Arnold's political career is more similar to Donald Trump's than his own career in acting. They are politicians of almost the same age who both support the Republican party. In a political context, their careers are far more similar than when Arnold's post as the Governor of California is compared with his portrayal of the Terminator in Terminator 2. Therefore, when the context of American politics is considered, the entity of Arnold S. from D_1 should be matched with the Donald T. entity from D_2. This is an example of *context-aware entity matching*.

1.2 Challenges and Problems

To match entities from heterogeneous sources in a unified way, Bellazi et al. [37] explain the importance of analyzing all data sources to identify interoperability conflicts. Vidal et al. [447] characterize the interoperability conflicts into six categories. We summarizes the main characteristics of each interoperability conflict.

1. Structuredness (C1): data sources may be described at different levels of structuredness, i.e. structured, semi-structured, and unstructured. The entities in a structured data source are described in terms of fixed schema and attributes, e.g. the entity-relationship model. In semi-structured data sources, a fixed schema is not required, and entities can be represented using different attributes and properties. Examples of semi-structured data models are the Resource Description Framework (RDF) or XML. Lastly, in unstructured data sources, the no data model is used, so the data does not follow any structured. Typically unstructured data formats are: textual, numerical, images, or videos.
2. Schematic (C2): the following conflicts arise when data sources are modeled with different schema. i) the same entity is represented by different attributes; ii) different structures model the same entity, e.g., classes versus properties; iii) the same property is represented with different data types, e.g., string versus integer; iv) different levels of specialization/generalization describe the same entity; v) the same entity is named differently; and vi) different ontologies are used, e.g., to describe a gene function the following ontologies may be used UMLS, SNOMED-CT, NCIT, or GO.
3. Domain (C3): various interpretations of the same domain exist on different data sources. These interpretations include: homonyms, synonyms,

acronyms, and semantic constraints—different integrity constraints are used to model a concept.
4. Representation (C4): different representations are used to model the same entity. These representation conflicts include: different scales and units, values of precision, incorrect spellings, different identifiers, and various encodings.
5. Language (C5): the data and schema may be specified using different languages, e.g. English and Spanish.
6. Granularity (C6): the data may be collected under different levels of granularity, e.g. samples of the same measurement observed at different time-frequency, various criteria of aggregation, and data model at different levels of detail.

2 Applications of Entity Matching

Entity Matching (EM) is an important operation in the field of data science and data management, and as such there are many practical applications where entity matching is necessary. In this section, we explore two applications of entity matching, namely *Data Integration* and *Knowledge Summarization*.

2.1 Semantic Data Integration

Semantic data integration is a research field that deals with integrating and reconciling *semantic heterogeneity* in different data sources. Towards this goal, the inclusion of semantics as a tool to aid data integration makes the entire process more powerful [101]. Using semantics in data integration means building data integration systems where the semantics of data are explicitly defined, and these semantics are used in turn during all the phases of data integration. It is unrealistic to entertain the idea that various data sources across the web will publish data using the same set of rules and conventions. Indeed, in reality data available across the World Wide Web have very different representations of the same information and concepts (entities). The stack of semantic technologies allows the opportunity for describing data semantically, and for interlinking disparate data sources. Thus, semantic integration is a useful approach for integrating semantically heterogeneous data. The bulk of the work done surrounding semantic data integration revolves around three aspects [332]. The first aspect is *mapping discovery*, or the process of automatically finding similarities between two ontologies and mapping properties that present the same real-world concept. The second is *mapping representation*, which is concerned with the specific method of representing mappings between two ontologies. The third and final aspect is *enabling reasoning*, which concerns itself with the process of performing reasoning over ontologies once the mapping has been established.

An example of an approach to achieve semantic data integration is the MINTE framework proposed by Collarana et al. [84]. MINTE is a *semantic integration technique* that is able to match and merge semantically equivalent

RDF entities in a single step through the utilization of semantics present in the vocabularies. MINTE uses both semantic similarity measures and the implicit knowledge present in the RDF vocabularies in order to match and merge RDF graphs that refer to the same real-world entity. MINTE's performance is powered by *semantic similarity measures*, *ontologies*, and *fusion policies* that consider not only textual data content but also logical axioms encoded into the graphs.

MINTE implements a two-step approach for determining the similarity between two RDF entities and then merging them. In the first step, MINTE implements a 1-1 weighted perfect matching algorithm to identify semantically equivalent RDF entities in input data sources. Then MINTE relies on *fusion policies* to merge triples from these semantically equivalent RDF entities. Fusion policies are rules operating on RDF triples, which are triggered by certain configurations of predicates and objects. Fusion policies can also resort to an ontology O to resolve possible conflicts. Collarana et al. define multiple fusion policies, e.g. *union policy*, *subproperty policy* and *authoritative graph policy*, which are each designed for flexible management and targeted control of an integrated knowledge graph. Figure *MINTE architecture* depicts the main components of the MINTE architecture. The accuracy of the process of determining when two RDF molecules are semantically equivalent in MINTE is impacted by the characteristics of the similarity measure Sim_f. Collarana et al. report the best performance when the GADES [371] similarity metric is used.

2.2 Summarization of Knowledge Graph

Another application of entity matching lies in the summarization of knowledge graphs. A knowledge graph is an ontology combined with a collection of instances that represents a collection of interlinked descriptions of entities. Knowledge graphs often capture domain-specific knowledge in the form of a graph. It has a data layer that contains the actual information and a semantic layer that represents the schema or the ontology. Typically knowledge graphs contain millions of entities and billions of properties describing these entities. This can lead to information overload, and therefore it is important to compress and summarize knowledge graphs for efficient representation of data [175].

The task of entity summarization is an essential part of knowledge graph summarization. Entity summaries allow the concise representation of the most important information about a certain real-world object. In the process of entity summarization, *entity matching* plays an important role. In order to summarize entities that either refer to the same real-world entity or are similar according to some summarization paradigm, it is first necessary to *identify* entities that belong in the same summary unit. For example, several knowledge graphs contain information about Marie Curie, each containing hundreds of facts about her life. For typical use cases, a summary containing a few basic items of information, namely her name, birth year, occupation and notable contributions, is enough to distinguish the most relevant aspects about her. To achieve this goal, it is first required to isolate entities from each knowledge graph that refer to Marie Curie. This is done using an entity matching technique. Knowledge graph

summarization can either be *concise* – containing only a subset of original facts, or *comprehensive* – containing an overview for all the original facts. The need for either a concise or a comprehensive summary depends on the particular case.

Knowledge graphs can also be summarized along different axes. For example, information can be summarized based on the semantic layer, i.e. ontology. It can also be summarized along different contextual layers, e.g., along time, geographic location, etc. In Chapter *Use Cases*, a temporal summarization technique for knowledge graph entities using COMET is described.

3 Novel Entity Matching Approaches

The problem of entity matching between disparate data sources is essential to the field of data integration. This is because one of the primary tasks in data integration is to reconcile varying schemas, thereby creating mapping entities between different data sources. Multiple approaches for inter-schema mapping exist both in the relational and graph database community. Multiple approaches also exist for the Entity Summarization – another application of entity matching.

A substantial amount of research has also been done over the idea of context and its role in data-driven tasks, particularly in the semantic web where the concept of data is intricately related to its semantics. The bulk of this research is limited to the formalization of context, although not much work has been done in practically implementing this concept. The following are some of the related carried out in formalizing context as well as a few practical approaches towards data integration and entity summarization.

3.1 Context in the Semantic Web

Principles for Formalizing Context: Bozzato et al. [63] present an argument that context needs to be represented in a more advanced manner in the Semantic Web and Linked Open Data (LOD). They further define a set of properties that a representation of context *should* abide by. These properties allow context to be an integral part of RDF data and its reasoning. The properties are as follows:

1. *Encapsulation*: data that share the same context must be encapsulated for ease in access and identification.
2. *Explicit meta knowledge*: contextual information must be represented in a logical language.
3. *Separation*: there must be a way to clearly distinguish meta knowledge from object knowledge.
4. *Relationship*: relationships between contexts must be explicitly represented.
5. *Encapsulation*: data that share the same context must be encapsulated for ease in access and identification.
6. *Contextual reasoning*: the representation should allow for reasoning to be done using the contextual knowledge.
7. *Locality*: each unit of context representation should allow the definition of axioms which are valid only within the local scope.

8. *Knowledge Lifting*: it should be possible to reuse knowledge from one context and apply it in another.
9. *Overlap*: the representation should allow for overlaps of knowledge between different contexts.
10. *Complexity invariance*: the addition of this contextual layer should not increase the complexity of reasoning.

The definition of context in COMET is guided by the principles defined above. Our definition particularly focuses on implementing the properties of *Explicit meta knowledge* and *Contextual reasoning*.

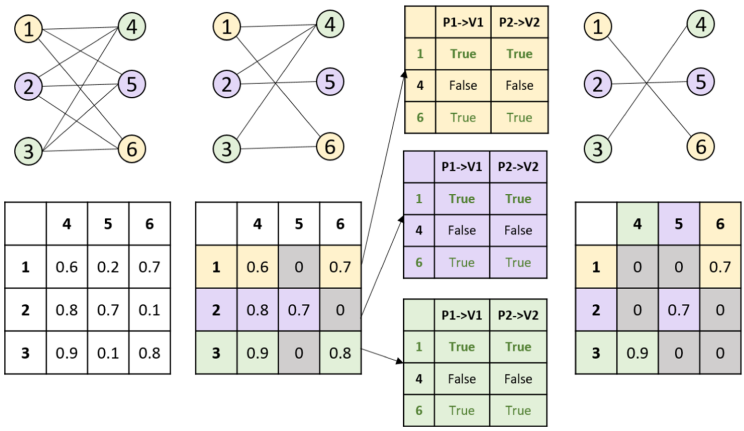


Fig. 2. Context Ontology for Data Integration (CODI) [409]. An overview of contextual elements (CE) defined in the context ontology

Context Ontology for Data Integration (CODI). The Context Ontology for Data Integration [409] was developed by Souza et al. to formally represent context in data integration processes. They first define the concept of Contextual Elements (CE) to represent the context of any domain-specific scenario. This is shown in Fig. 2. They then build a context ontology suited to the domain-specific scenario after having meetings with domain experts. This context ontology is then used during the reconciliation of schema during data integration. Although this approach works with the formal definition of context in data integration scenarios, it is still quite expensive since it cannot work without extensive input from domain experts in the modeling of the Context Ontology. It also does not make use of the semantics already existing in the data instances to guide its modeling of context.

3.2 Entity Matching Approaches

Applications in Data Integration. Data Integration (DI) is one of the most common applications that require entity matching. This matching is done at either at a schema-level or at an instance-level. There are a number of approaches that aim at the integration of disparate RDF data sources. We divide these works based on whether they match ontologies, or the instances themselves.

Ontology Matching Approaches. Many of the data integration approaches based on RDF data apply the concept of mapping heterogeneous data sources to a common ontology. One approach using ontologies is KARMA, proposed by Knoblock et al. [247]. This is a framework for integrating a variety of data sources including databases, spreadsheets, XML, JSON, and Web APIs. KARMA implements a hybrid approach that relies on supervised machine algorithms for identifying mapping rules from structured sources to ontologies; these mapping rules can be refined by users via a user interface.

Another approach is suggested by Schultz et al. [388], who describe the Linked Data Integration Framework (LDIF). LDIF is oriented to integrate RDF datasets from the Web and provides a set of independent tools to support interlinking tasks. LDIF provides an expressive mapping language for translating data from various vocabularies to a unified ontology. LDIF tackles the problem of identity resolution by defining linking rules using the SILK tool [213]. Based on the defined rules, SILK identifies `owl:sameAs` links among entities of two datasets.

Instance Matching Approaches. In the task of identifying whether given entities refer to the same real-world entity, growing attention in the relational databases field is given to crowdsourcing mechanisms [242,445]. Reporting impressive results, such approaches, however, might struggle in sophisticated domains with multiple contexts due to a lack of human experts who could reliably provide necessary example data.

ODCleanStore [307] and UnifiedViews [246] are ETL frameworks for integrating RDF data. ODCleanStore relies on SILK to perform instance matching and provides custom data fusion modules to merge the data of the discovered matches.

The MINTE framework proposed by Collarana et al. [84] also tackles the task of matching entities in different datasets that correspond to the same real-world entity by making use of the semantics encoded in the data itself. They first apply a *semantic similarity metric* in order to identify *semantically equivalent* entities from two different RDF graphs. Next they make use of a set of novel *fusion policies* to merge these semantically equivalent entities. Although MINTE makes use of the semantics encoded into the RDF graph itself, it does not consider the context during the step of entity matching. The work done in COMET is in essence a context-based extension of MINTE (Fig. 3).

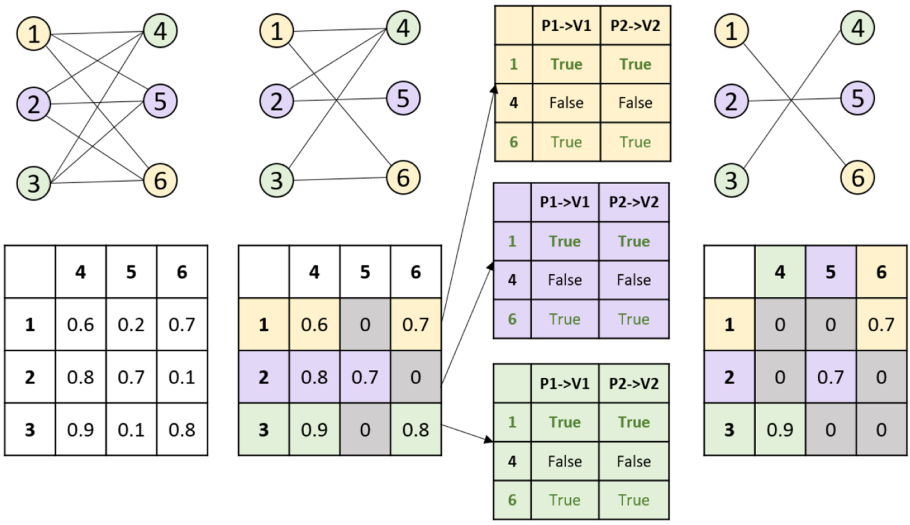


Fig. 3. Entity Summarization. Summarizing a single entity as envisioned by LinkSum [424]

Applications in Entity Summarization. Entity summarization is the process of creating a concise representation of an entity in order to describe the whole entity. A number of approaches have been formulated in order to generate summaries of entities [175]. One such approach is RELIN [79] by Cheng et al. where they defined the problem of entity summarization using RDF graphs and demonstrated its utility in entity identification. RELIN makes use of the PageRank algorithm to select relevant features in the creation of the summary entity. In 2014, Thalhammer and Rettinger proposed SUMMARUM [425], a dbpedia-based entity summarization framework that also uses PageRank in order to rank the features of an entity. It also uses the global popularity of DBPedia resources corresponding to their Wikipedia pages. They later proposed LinkSum [424], which in addition to PageRank also makes use of an adaptation of the Back-Link method combined with new methods for predicate selection. These entity summarization frameworks focus on the rank of features (attributes) in order to create the summary, but do not take into consideration any contextual dimension of the data. The above-mentioned integration frameworks aim at mapping different data sources with possibly varying schema, i.e., they perform inter-schema mapping. ontext-based integration could only be supported in these frameworks on a superficial level via filtering query results without applying many inherent semantics. Similarly, the entity summarization frameworks aim at summarizing via some order of properties instead of considering contextual information present in the data. Therefore, we identify a need for context-based entity matching mechanisms and present our approach, which can be adapted for both integration and summarization of RDF data (Fig. 4).

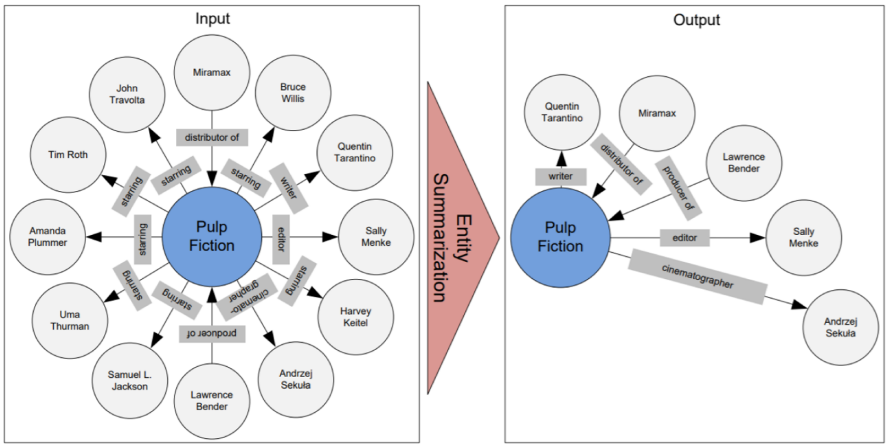


Fig. 4. Entity Summarization. Summarizing a single entity as envisioned by LinkSum [424]

4 COMET: A Context-Aware Matching Technique

To provide a solution to the *problem of contextually matching* RDF entities, COMET – a context-aware RDF molecule matching technique – is proposed. This technique is grounded on the semantic data integration techniques proposed by Collarana et al. [84], whose work deals with matching and merging RDF molecules that are semantically similar using semantic similarity metric and fusion policies. This work makes use of the concepts of RDF molecules but contributes a new approach as to taking into consideration the *context* of the system while matching entities. COMET is an entity matching framework designed to create, identify, and match contextually equivalent RDF entities. Grounded on the entity matching component from the data integration technique proposed by Collarana et al. [84], we propose COMET, an entity matching approach to merge equivalent RDF entities based on context. Thus, a solution to the *problem of contextually matching entities* is provided (Fig. 5).

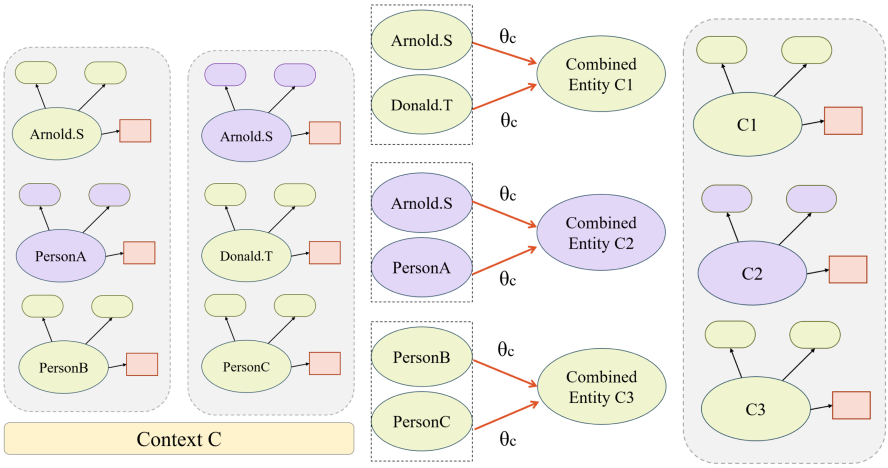
4.1 Problem Definition

RDF Molecule [84] – If $\Phi(G)$ is a given RDF Graph, we define RDF Molecule M as a subgraph of $\Phi(G)$ such that,

$$M = \{t_1, \ldots, t_n\}$$

$$\forall \quad i, j \in \{1, \ldots, n\} \big(subject(t_i) = subject(t_j)\big)$$

Where $t_1, t_2, \ldots, t_n$ denote the triples in M. In other words, an RDF Molecule M consists of triples which have the same subject. That is, it can be represented by a tuple $M = (R, T)$, where R denotes the URI of the molecule's



(a) $\Phi(G)$, $\Phi(D)$ and Context C (b) Homomorphism θ_C (c) Idealized F_c

Fig. 5. Problem Definition. The left side shows two RDF Graphs the system Context. The right side shows the application of homomorphism θ_C on the RDF graphs, resulting in the formation of Contextualized RDF Graph F_c.

subject, and T denotes a set of property and value pairs p = *(prop, val)* such that the triple *(R, prop, val)* belongs to M. For example, the RDF molecule for *Arnold Schwarzenegger* is (dbr:Arnold-Schwarzenegger, { (`dbo:occupation`, *Politician*), (`dbp:title`, *Governor*)}). An RDF Graph $\Phi(G)$ described in terms of RDF molecules is defined as follows:

$$\Phi(G) = \{M = (R, T) | t = (R, prop, val) \in G \wedge (prop, val) \in T\}$$

Context – We define a context C as any Boolean expression which represents the criteria of a system. Two entities, such as an RDF molecule M_1 and M_2, can be either similar or not similar with respect to a given context. That is, C is a Boolean function that takes as input two molecules M_1 and M_2 and returns `true` if they are similar according to system context, and `false` otherwise. Below is an example of context C, modeled after the example presented in Fig. 1, where two molecules are similar if they have the same occupation. If $P = (p, v)$ is the predicate representing the occupation property of a molecule, then context.

$$C(M_1, M_2) = \begin{cases} \texttt{true}, & \text{if } P \in M_1 \wedge P \in M_2. \\ \texttt{false}, & \text{otherwise.} \end{cases}$$

Depending on the requirements of the integration scenario, this context can be any Boolean expression.

Semantic Similarity Function – Let M_1 and M_2 be any two RDF molecules. Then *semantic similarity function* Sim_f is a function that measures the *semantic*

similarity between these two molecules and returns a value between [0, 1]. A value of 0 expresses that the two molecules are completely dissimilar and 1 expresses that the molecules are identical. Such a similarity function is defined in GADES [371].

Contextually Equivalent RDF Molecule – Let $\Phi(G)$ and $\Phi(D)$ be two sets of RDF molecules. Let M_G and M_D be two RDF molecules from $\Phi(G)$ and $\Phi(D)$, respectively. Then, M_G and M_D are defined as contextually equivalent iff

1. They are in the same context. That is, $C(M_1, M_2) = \texttt{true}$
2. They have the highest similarity value, i.e.,
   $Sim_f(M_G, M_D) = max(\forall_{m \in \Phi(D)} Sim_f(M_G, m))$

Let F_c be an idealized set of *contextually integrated* RDF molecules from $\Phi(G)$ and $\Phi(D)$. Let θ_C be a homomorphism such that $\theta_C : \Phi(G) \cup \Phi(D) \rightarrow F_c$. Then there is an RDF Molecule M_F from F_c such that $\theta(M_D) = \theta(M_G) = M_F$. From the motivation example, this means that the molecule of *Arnold Schwarzenegger*, the politician, is *contextually equivalent* to the molecule of *Donald Trump* as they are similar *and* they satisfy the context condition of having the same occupation.

In this work, we tackle the problem of explicitly modeling the context and then matching RDF molecules from RDF graphs that are both highly similar and equivalent in terms of this context. This problem is defined as follows: given RDF graphs $\Phi(G)$ and $\Phi(D)$, let M_G and M_D be two RDF molecules such that $M_G \in \Phi(G)$ and $M_D \in \Phi(D)$. The system is supplied with a context parameter C, which is a Boolean function evaluating if two molecules are in the same context. It is also supplied with a similarity function Sim_f, which evaluates the semantic similarity between M_G and M_D.

The problem of creating a contextualized graph Φ_C consists of building a homomorphism $\theta_C : \Phi(G) \cup \Phi(D) \rightarrow F_c$, such that for every pair of RDF molecules belonging to Φ_C there are none that are *contextually equivalent* according to system context C. If M_G and M_D are contextually equivalent molecules belonging to F_c, then $\theta_C(M_G) = \theta_C(M_D)$, otherwise $\theta_C(M_G) \neq \theta_C(M_D)$.

An example of this problem is illustrated in Figure X, which depicts a use case with two RDF graphs and a single context condition C. With respect to C, the RDF molecule *Arnold.S* from $\Phi(G)$ is in the same context as *Donald.T* from $\Phi(D)$, but not in the same context as the molecule *Arnold.S* from $\Phi(G)$. So the problem is to identify a homomorphism θ_C which evaluates the RDF molecules based on system context and maps these RDF molecules in a way that they can be integrated into a contextualized graph.

4.2 The COMET Architecture

We propose COMET, an approach to match contextually equivalent RDF graphs according to a given context, thus providing a solution to the problem of *contextually matching* RDF graphs. Figure 6 depicts the main components of the COMET architecture. COMET follows a two-fold approach to solve the problem

of entity matching in RDF graphs in a context-aware manner: First, COMET computes the similarity measures across RDF entities and resorts to the Formal Concept Analysis algorithm to map contextually equivalent RDF entities. Finally, COMET combines the results of the first step and executes a 1-1 perfect matching algorithm for matching RDF entities based on the combined scores to finally synthesize the matching into a contextualized RDF graph.

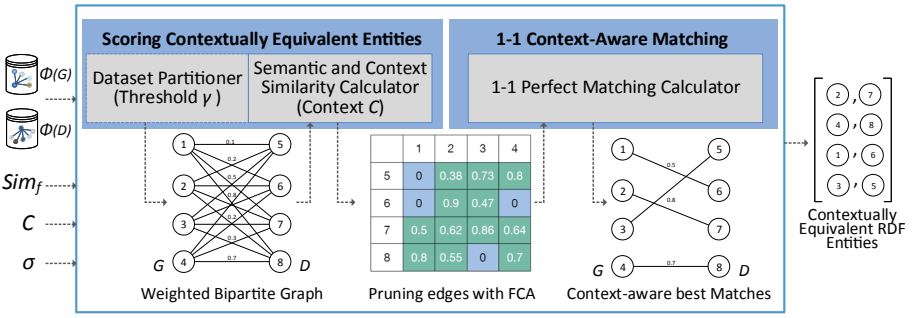
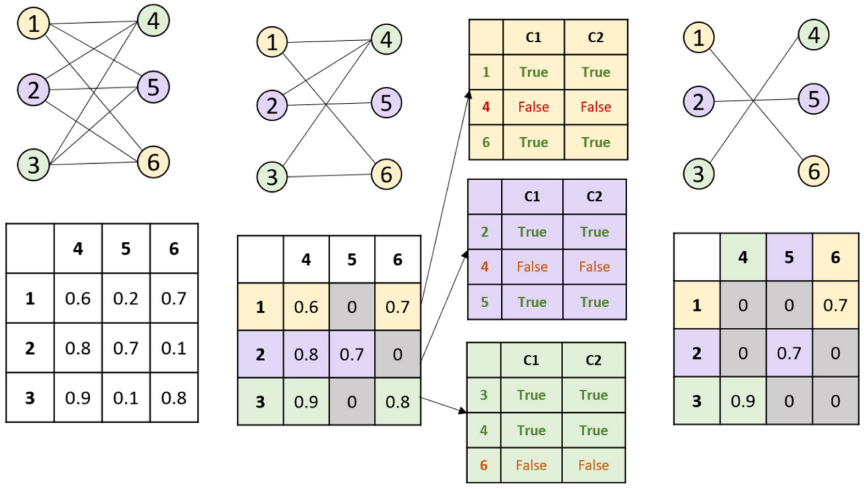


Fig. 6. The COMET Architecture. COMET receives two RDF datasets, e.g., G and D; a similarity function Sim_f; and a context C. The output is a set of contextually matching RDF entities.

4.3 Identifying Contextually Equivalent Entities

Building a Bipartite Graph. The COMET pipeline receives two RDF graphs $\Phi(G), \Phi(D)$ as input, along with context parameter C, and a similarity function Sim_f. COMET first constructs a bipartite graph between the sets $\phi(G)$ and $\phi(D)$. The *Dataset Partitioner* employs a similarity function Sim_f and ontology O to compute the similarity between RDF molecules in $\phi(G)$ and $\phi(D)$ assigning the similarity score as vertices weight in the bipartite graph. COMET allows for arbitrary, user-supplied similarity functions that leverage different algorithms to estimate similarity between RDF molecules. Thus, COMET supports a variety of similarity functions including simple string similarity. However, as shown in [84], semantic similarity measures are advocated (in the implementation of this work we particularly use GADES [371]) as they achieve better results by considering semantics encoded in RDF graphs.

After RDF molecules similarity comparison, the result of the similarity function is tested against a threshold γ to determine entity similarity (the similarity threshold's minimum acceptable score). Thus, edges are discarded from the bipartite graph whose weights are lower than γ. A threshold equal to 0.0 does not impose any restriction on the values of similarity; thus the bipartite graph includes all the edges. High thresholds, e.g. 0.8, restrict the values of similarity, resulting in a bipartite graph comprising just a few edges.



(a) Bipartite graph after applying threshold $\gamma = 0.5$ (b) Context validation using FCA (c) Perfect 1-1 matches

Fig. 7. Context Validation. The left side shows a bipartite graph after the application of threshold. The remaining edges go through a special 1-1 matching algorithm which takes into account the system context using FCA. The result is a perfect match between contextually equivalent molecules.

Pruning RDF Entities According to ContexB. The main step on the COMET pipeline is to validate and prune pairs of RDF molecules that do not comply with the input context C, making COMET a context-aware approach. For identifying contextually equivalent RDF entities, the *Context Validator* component employs the Formal Concept Analysis (FCA) algorithm. FCA is the study of binary data tables that describe the relationship between objects and their attributes. Applying this context validation step over the RDF molecules ensures that only contextually relevant tuples are kept. In COMET, context is modeled as any Boolean function. Two molecules are matched if they satisfy this condition, otherwise they are not matched. The algorithm by V. Vychodil [451] is applied in COMET; it performs formal concept analysis to compute formal concepts within a set of objects and their attributes. This algorithm is extended in our approach for validating complex *Boolean conditions*. A typical formal concept analysis table is shown in Table 1.

Table 1. Object-Attribute table for performing FCA.

	Attribute 1	Attribute 2	Attribute 3
Object 1		X	X
Object 2		X	
Object 3	X		X

Instead of using attributes in the column of the FCA matrix, in our approach, we replace the attributes with a *boolean condition* C. This is the same as the context condition C used in our approach. For example, the context C from the motivating example can be broken down into $C = C_1 \wedge C_2$ where $C_1 =$ "contains property `dbo:occupation`", and $C_2 =$ "has the same value for property `dbo:occupation`". The execution of the FCA algorithm remains unchanged by this adaptation since the format of the input to FCA is still a binary matrix.

When applied to RDF molecules, formal concept analysis returns a set of formal concepts $< M, C >$ where M is a set of all the molecules that contain all conditions contained in C. That is, by applying FCA, the set of molecules that satisfy a certain context condition can be obtained. Thus, the molecules that do not meet the context condition are pruned. In Fig. 7, an example of context validation is demonstrated. Edges in a bipartite graph are filtered according to a threshold value γ as detailed in the previous section. Next, the remaining edges are validated by constructing an FCA matrix according to context condition C. The FCA algorithm returns the edge satisfying the context conditions. The edges that do not satisfy the context condition are discarded.

4.4 The 1-1 Perfect Matching Calculator

COMET solves the problem of *context-aware entity matching* by computing a 1-1 weighted perfect matching between the sets of RDF molecules. The input of the 1-1 weighted perfect matching component is the weighted bipartite graph created on the previous step. Since each weight of an edge between two RDF molecules corresponds to a combined score of semantic similarity and context equivalence value, we call this a 1-1 context-aware matching calculator. The effect of this 1-1 context aware matching calculator is demonstrated in Fig. 9 Finally, a combinatorial optimization algorithm like the Hungarian algorithm [267] is utilized to compute the matching.

4.5 Integration Use Case: Applying Fusion Policies

In order to apply this context-aware entity matching pipeline into a data integration scenario, we envision the usage of *fusion policies* defined by Collarana et al. [84]. To consolidate entities identified as contextually equivalent, COMET can make use of synthesis policies, i.e. a user-supplied function that defines how the RDF molecules should be combined to form a connected whole. COMET can adopt the following synthesis policies:

1. The *Union Policy*, which includes all predicates-object pairs, removing the one that is syntactically the same;
2. The *Linking Policy*, which produces `owl:sameAs` links between contextually equivalent RDF molecules;
3. The *Authoritative Policy*, which allows for defining one RDF graph as a prevalent source selecting its properties in case of property conflicts, i.e., properties annotated as `owl:FunctionalProperty`, equivalent properties `owl:equivalentProperty`, and equivalent classes annotated with `owl:sameAs` or `owl:equivalentClass`.

Algorithm 1: closure(B,y)

```
for j ← 0 to n do
    D[j] ← 1;
foreach i in rows[y] do
    match ← True;
    for j ← 0 to n do
        if { B[j] = 1
           { context[i, j] = 0   then
            match ← False;
            break for loop;
    if match = True then
        for j ← 0 to n do
            if context[i, j] = 0 then
                D[j] ← 0;
return D
```

Algorithm 2: generate(B,y)

```
process B ;                    // Printing B
if B = Y | y > n then
    return
for j ← y to n do
    if B[j] = 0 then
        B[j] ← 1;
        D ← closure(B,j);
        skip ← False;
        for k ← 0 to j − 1 do
            if D[k] ≠ B[k] then
                skip ← True;
                break for loop;
        if skip = False then
            generate(D,j + 1);
        B[j] ← 0;
return
```

Fig. 8. Implemented algorithms (extended from [451]).

By applying these policies, the end output is a synthesized graph with linked entities that are *contextually equivalent*. In the next chapter, we take a look at another use case of context-aware entity matching: the temporal summarization of knowledge graph entities.

5 Empirical Evaluation

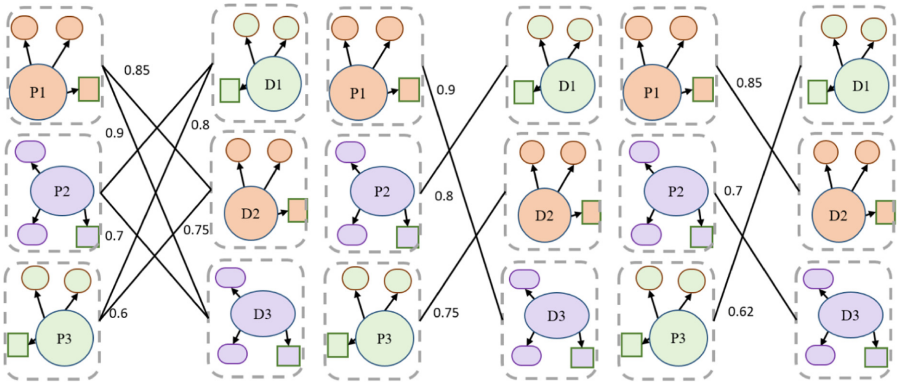
This section presents an overview of the technical details, execution and the results obtained in the empirical evaluation.

5.1 Research Questions

We conducted an empirical evaluation to study the effectiveness and performance of COMET in solving the entity matching problem among RDF graphs. We address the following research questions:

- **(RQ1)** Is COMET able to perform entity matching with regard to context more accurately than the MINTE [84] entity matching component?
- **(RQ2)** Does the content of the dataset with respect to the context condition affect the accuracy of COMET?
- **(RQ3)** How much overhead does the context-evaluation step in COMET add to the overall pipeline?

(RQ1) and **(RQ2)** are combined to conduct **Experiment 1** in order to evaluate the effectiveness or accuracy of COMET. **(RQ3)** is addressed by **Experiment 2** where the overhead of the context-evaluation step is measured.



(a) Original bipartite graph with γ=0.5 (b) 1-1 matching without context validation (c) 1-1 matching with context validation

Fig. 9. The **1-1 Perfect Matching.** COMET applies a special 1-1 perfect matching algorithm which evaluates context as well as similarity between two molecules. A traditional 1-1 perfect matching algorithm only considers similarity (weight of edges). Without evaluating context, the 1-1 matching algorithm matches molecules that are not in the same context. When context is evaluated alongside similarity, molecules in the same context are matched.

5.2 Implementation

Practically, COMET is implemented in Python and hosted in GitHub[2] along with the datasets and logs used in this evaluation. The experiments were executed on a Windows 10 (64 bits) machine with CPU: Intel Core i7-8650U 1.9 GHz (4 physical cores) and 16 GB RAM. For the COMET pipeline we use the semantic similarity measure *GADES* [371], which Collarana et al. have previously demonstrated to have the best performance in terms of accuracy when added to their MINTE pipeline [84]. *GADES* relies on semantic description encoded in ontologies to determine relatedness. *GADES* examines both string similarity and hierarchy similarity by making use of graph neighbourhoods.

[2] https://github.com/RDF-Molecules/COMET.

Table 2. Benchmark Description. Datasets used in the evaluation including: number of RDF molecules (M), number of triples (T), evaluated contexts (C).

	Experiment 1: Effectiveness					
Configuration	**A**		**B**		**C**	
Datasets	*A1*	*A2*	*B1*	*B2*	*C1*	*C2*
Molecules	1000	1000	1000	1000	1000	1000
Triples	70,660	70,660	70,776	70,776	71,124	71,124
Context	$C(M_{D1}, M_{D2}) =$ `true`, if `dbo:occupation` match					

	Experiment 2: Runtime							
Datasets	*XS1*	*XS2*	*S1*	*S2*	*M1*	*M2*	L1	L2
Molecules	100	100	500	500	1,000	1,000	2,000	2,000
Triples	7,084	7,084	33,916	33,916	71,124	71,124	138,856	138,856

5.3 Baseline

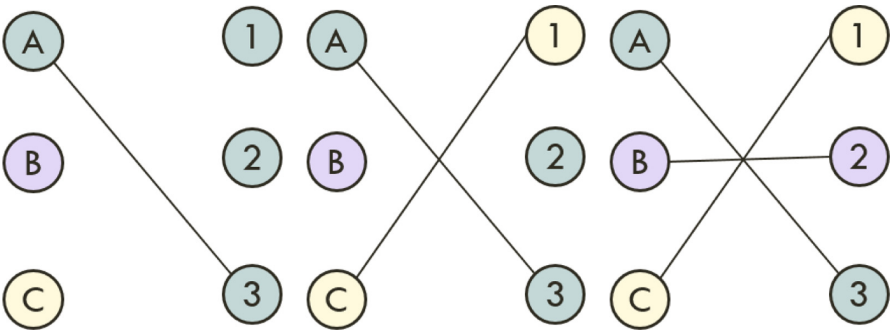
As a baseline, we compare the effectiveness of COMET against the MINTE pipeline proposed by Collarana et al. [84]. Towards **(RQ1)** and **(RQ2)** we design an experiment to measure the *precision*, *recall* and *f-measure* of COMET in comparison to MINTE. We also run COMET and MINTE on datasets with different compositions of molecules with respect to context in order to observe the effect of contextual content of datasets on the effectiveness of COMET. Towards **(RQ3)**, we observe the impact of COMET context-evaluation step on temporal and memory performance.

5.4 Effectiveness Evaluation

Metrics. Although each experiment has different datasets and gold standards, we use the same metrics for all the experiments: *Precision*, *Recall*, and *F-meaure*. *Precision* measures what proportion of the performed entity matches are actually correct. That is, *precision* is the fraction of RDF molecules that has been identified as contextually equivalent by COMET (C), which intersects with the Gold Standard (GS). On the other hand, *recall* measures the overall proportion of integrated RDF molecules that were identified correctly. That is, *recall* is measured by the fraction of correctly identified similar molecules with respect to the Gold Standard,i.e., $Precision = \frac{|C \cap GS|}{|C|}$ and $Recall = \frac{|C \cap GS|}{|GS|}$. *F-measure* is the harmonic mean of *Precision* and *Recall*.

Datasets. For this experiment, we use datasets containing 1,000 people entities from DBpedia. In order to test the effect of contextual data content on the accuracy of COMET, three pairs of datasets *(A1, A2)*, *(B1, B2)*, and *(C1, C2)* are generated using configurations *A*, *B*, and *C*, respectively. These configurations are as follows:

1. **Configuration A:** Every molecule *a1* in dataset A1 has **2** *highly similar* molecules *a2* and *a3* in dataset A2, such that *a2* satisfies context condition, but *a3* does not. That is, $C(a1, a2) = \texttt{true}$ and $C(a1, a3) = \texttt{false}$.
2. **Configuration B:** Every molecule *b1* in dataset B1 has **3** *highly similar* molecules *b2*, *b3* and *b4* in dataset B2, such that *b2* and *b3* satisfy the context but *b4* does not.
3. **Configuration C:** Every molecule *c1* in dataset C1 has **4** *highly similar* molecules in dataset C2, two of which satisfy the context condition, and two that do not.



(a) Molecule A has only one perfect match (b) Molecule A has two perfect matches (c) Molecule A has three perfect matches.

Fig. 10. Effect of dataset content on matching accuracy. The goal of COMET is to choose the most similar molecule which is also in the same context. With a higher number of similar molecules within the same context, the probability of COMET choosing the correct match every time decreases.

The motivation of curating datasets using these three configurations is as follows: As seen in Sect. 4, COMET applies a special 1-1 perfect matching algorithm to find the best match according to both similarity and context condition. For this reason, the varying number of highly similar molecules that are also in the same context will affect the way COMET performs on the dataset. A higher number of similar molecules in the same context means a lesser chance of COMET identifying the correct match.

This is demonstrated in Fig. 10. Here, circles displaying the same color denote that they are molecules in the same context. In Fig. 10a, molecule A has only one perfect match available in the matching dataset and COMET makes this match accordingly. But in Fig. 10b and 10c, the number of perfect matches within the same context increases to two and three, respectively. This means that the probability of COMET identifying the true match for Molecule A decreases. Therefore we aim to evaluate exactly how the varying numbers of similar molecules in a dataset affect the accuracy of COMET.

Table 3. Effectiveness evaluation of COMET.

Configuration	COMET			MINTE		
	Precision	*Recall*	*F-Measure*	*Precision*	*Recall*	*F-Measure*
A	1.0	1.0	1.0	0.54	0.54	0.54
B	0.708	0.708	0.708	0.449	0.449	0.449
C	0.558	0.558	0.558	0.408	0.408	0.408

Every pair of datasets is synthesized as follows: First, molecules from the original set of 1,000 DBpedia person entities are duplicated according to the configuration condition to create n number of highly similar molecules in the second dataset. Then predicates inside the similar molecules are randomly edited and deleted to create some variation of similarity. The predicates are then edited to ensure that the correct number of similar molecules in the second dataset satisfy the context according to the original dataset.

Context and Gold Standard. Similar to the motivation example shown in Fig. 1, the context C used in this experiment checks if two molecules have the same value for the predicate `dbo:occupation`. The Gold Standard contains matches between molecules that (i) satisfy the context condition; and (ii) are highest in similarity among all other molecules. For every pair of datasets belonging to the three configurations (i.e. configuration A, B and C), there is a corresponding Gold Standard G_A, G_B and G_C. The datasets, gold standard and the experiment code are all available on GitHub.

Experiment 1: Contextually Matching DBpedia RDF Molecules. Table 2 describes the dataset used during our evaluations. This experiment was conducted on MINTE and COMET once for each pair of datasets *(A1, A2)*, *(B1, B2)* and *(C1, C2)*, with the context condition requiring that every pair of matched molecules must have the same value for `dbo:occupation` property. The threshold value γ for this experiment is applied at the 97th percentile in every case. Then comparing against the Gold Standard G_A, G_B and G_C for configurations A, B and C respectively, the metrics Precision and Recall were calculated each time. The results are presented in Table 3.

Experiment 2: Impact of Context Evaluation on Performance. In addition to the previous reported experiments focusing on effectiveness, we also pay particular attention to the evaluation of performance. Indeed, we specifically design an experiment to analyze how much overhead is added to COMET for evaluating context in its entity matching pipeline with respect to MINTE, which does not evaluate context.

Metrics. During our tests, we monitored each task by measuring not only execution time but a broader set of metrics:

- *Time* (seconds): measures the time used to process the various tasks and sub-tasks relevant in our study;
- *Memory (& SWAP)* (Bytes): allows for keeping track of the memory allocations during the experiments; the idea is to search for possible bottlenecks caused due to adding context evaluation to our approach.
- *CPU usage* (Percentage): simply records the processing activity.

Datasets. Table 2 reports on the datasets used during this set of experiments. As shown, we fed COMET with four datasets, each one involving more triples than the previous ones; they contain from 7 000 up to 100,000 triples. The molecules range from sets of 100 to 2,000.

Since COMET performs analysis of molecules both in its creation of bipartite graphs and context evaluation step, we wanted to observe how the performance is affected by increases in molecule number.

Temporal Performance. In Fig. 11, we present the obtained results with the datasets XS, S, M, and L. This representation is twofold. Firstly, the bar chart represents for each dataset the time distribution according to the various phases which COMET involved: i.e. computing similarity in a bipartite graph, evaluating context using FCA computation, and performing 1-1 perfect matching in blue, purple, and yellow, respectively. Secondly, the red curve presents for each dataset the total time required to run the experiment; notice that we use a logarithmic scale. As a consequence, we successfully find experimentally that the context evaluation step does not take any more time than the other phases. As shown in the bars of Fig. 11, the purple section representing the context evaluation step does occupy a greater percentage of the total runtime as the size of the dataset increases, but it still consumes less than half of the runtime in comparison to the other phases.

Memory Consumption. To analyze how the memory is used by COMET during its experimental runs, we pay attention to the RAM & SWAP consumption for each dataset tested; in addition we also recorded the CPU usage. It appears that COMET did not use much of the resources to run the experiments with the datasets. Moreover, we even observed that the pressure on the CPU and the amount of memory used by the system at every second is almost the same for all the considered datasets. This, therefore, means that the current implementation of COMET rather spreads its computations along the time (see again the red curve of Fig. 11) instead of allocating more resources when the dataset size increases.

5.5 Discussion of Observed Results

Based on the values of Precision, Recall, and F-measure reported in Experiment 1 (Table 3), we can positively answer **(RQ1)**, and **(RQ2)** i.e., COMET is able to effectively match entities across RDF graphs according to context, and indeed the content of the datasets does affect the accuracy. In every case, COMET performs better than MINTE, and the reason is clear – MINTE does not take context into consideration during its 1-1 perfect matching whereas COMET does.

Moreover, the decrease in precision in recall of COMET with the increase of the number of highly similar molecules within the dataset also makes sense. With a higher number of similar molecules to choose from, COMET has less of a chance of correctly identifying the perfect match. On the other hand, in the case of *configuration A*, the precision and recall is perfect. This is because the dataset in *configuration A* supplies only 1 perfect option (a highly similar molecule that also meets the context). The perfect precision and recall demonstrate that in an ideal condition with only 1 perfect option, COMET will always match correctly.

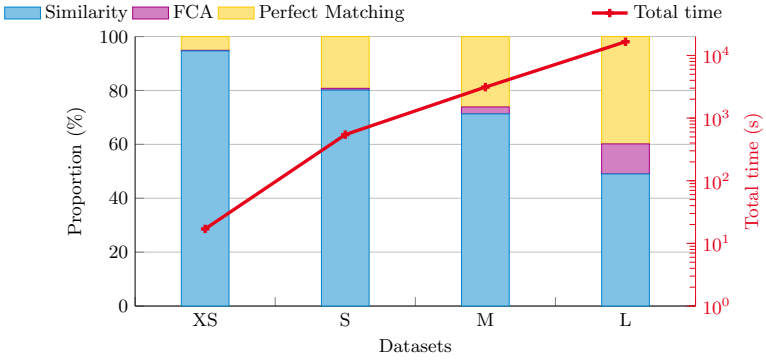


Fig. 11. Temporal performance for datasets XS, S, M and L. The bars along the y-axis represent the time distribution required for each of the phases of COMET. The red curve presents the total time required to run the experiment in logarithmic scale (Color figure online).

By observing the temporal performance (Fig. 11) and memory consumption of COMET when supplied with datasets of increasing volumes, we can also answer **(RQ3)**, i.e., by measuring the amount of overhead the context-evaluation step of COMET adds to the overall pipeline. We show that the context evaluation step adds a fraction of the temporal overhead with respect to a traditional 1-1 matching algorithm, and does not have any observable overhead in terms of memory consumption.

6 Grand Challenges and Conclusions

In the age of data variety, adding and considering data context is more important than ever. Context lends information to its scope of validity and affects most data-driven tasks, such as data integration. In this chapter, we presented COMET – an approach to match contextually equivalent RDF entities from different sources into a set of 1-1 perfect matches between entities. COMET follows a two-fold approach where first contextually equivalent RDF molecules are identified according to a combined score of semantic and context similarity. Then, a 1-1 perfect matching is executed to produce a set of matches considering context. COMET utilizes the Formal Concept Analysis algorithm to decide whenever two RDF molecules are contextually equivalent. The behavior of COMET

was empirically studied on two real-world RDF graphs under different context configurations. The observed results suggest that COMET is able to effectively identify and merge contextually equivalent entities in comparison to a baseline framework which does not consider context. We also envision an approach for creating entity summaries automatically out of different temporal versions of a knowledge graph. To do so, the proposed approach utilizes the concepts of RDF molecules, Formal Concept Analysis, and Fusion Policies. The entity evolution summaries created by the approach may serve to create documentation, to display a visualization of the entity evolution, or to make an analysis of changes.

This work is a first step in the direction of the formalization of context and its effect on data-driven tasks. Therefore, there are still **grand challenges** to face towards consolidating context-based similarity approaches. Thus, we present the four grand challenges that should be tackled as next steps, i.e.: 1) *measuring context* with probabilistic functions; 2) the *performance* of the context-aware matching algorithms; 3) full *usage of the semantic representation* of entities as knowledge graphs; and furthermore, 4) the *application* of context-aware entity matching on a variety of data-driven tasks.

We now describe them in detail:

1. **Measuring context with probabilistic functions:** In this chapter, we employ a straightforward definition of context conditions, i.e. modeling context as a Boolean function of entities. According to this model, an entity is either valid within a context or invalid. The real-world meaning and scope of context are much more general, and therefore context should be modeled in a more generalized way. For example, the measure of the validity of an entity concerning different contexts can be a probabilistic function. Meaning the range of the context function could be any value between the interval [0,1] instead of being only 0 or 1. We suggest the use of Probabilistic Soft Logic (PSL) to implement this concept.
2. **Performance:** Although in this chapter, we focus on the variety dimension of big data, context-based approaches should apply to the volume dimension as well. In COMET, for example, the complexity of the 1-1 matching algorithm is quadratic as COMET employs the original Formal Concept Analysis algorithm. As such, it is possible to evaluate a distributed version of the Formal Concept Analysis algorithm that may improve the run-time overhead in this work. Big data frameworks such as Hadoop and Spark can be used in the implementation of this distributed version of COMET.
3. **Exploitation of the semantic representation of entities:** The proposed approach, presented in this chapter, utilizes the knowledge encoded in RDF graphs themselves to create context parameters. Nevertheless, not all the potential of semantics has been studied to improve the accuracy of context-based matching approaches. A natural next step, for example, would be to take advantage of the implicit knowledge encoded in RDF Knowledge Graphs. Employing a reasoner additional contextual data can be inferred, empowering the modeling and evaluating of context.

4. **Application of context-aware matching on various data-driven tasks:** We mentioned during the chapter the application of the COMET approach in the entity summarization use-case. Tasnim et al. [422] show the architecture and pipeline modifications to COMET in order to produce a summary along one contextual axis, i.e. temporal axis. The approach can be adapted to other contextual axes, e.g., geographic location, hierarchical position, and more. Depending on the contextual axis, many more use cases of context-aware entity matching can be explored. Also, the elements used in the creation of the entity evolution summary, e.g., the ontology, can be investigated and further developed to empower the approach.