**EABL0ck:** A Declarative Entity Alignment Block for Knowledge Graph Creation Pipelines

Samaneh Jozashoori\(^1\,2\), Ahmad Sakor\(^1\,2\), Enrique Iglesias\(^3\), Maria-Esther Vidal\(^1\,2\,3\)

\(^1\) TIB Leibniz Information Center for Science and Technology, Germany
\({\text{samaneh.jozashoori, ahmad.sakor, maria.vidal}@tib.eu}\)
\(^2\) Leibniz University of Hannover, Germany
\(^3\) L3S Research Center, Germany

\[\text{iglesias}@l3s.de\]

**Abstract.** Despite encoding enormous amount of rich and valuable data, existing data sources are mostly created independently, being a significant challenge to their integration. Mapping languages, e.g., RML and R2RML, facilitate declarative specification of the process of applying meta-data and integrating data into a knowledge graph. Mapping rules can also include knowledge extraction functions in addition to expressing correspondences among data sources and a unified schema. Combining mapping rules and functions represents a powerful formalism to specify pipelines for integrating data into a knowledge graph transparently. Surprisingly, these formalisms are not fully adapted, and many knowledge graphs are created by executing ad-hoc programs to pre-process and integrate data. In this paper, we present EABL0ck, an approach integrating Entity Alignment (EA) as part of RML mapping rules. EABL0ck includes a block of functions performing entity recognition from textual attributes and link the recognized entities to the corresponding resources in Wikidata, DBpedia, and domain specific thesaurus, e.g., UMLS. EABL0ck provides agnostic and efficient techniques to evaluate the functions and transfer the mappings to facilitate its application in any RML-compliant engine. We have empirically evaluated EABL0ck performance, and results indicate that EABL0ck speeds up knowledge graph creation pipelines that require entity recognition and linking in state-of-the-art RML-compliant engines. EABL0ck is also publicly available as a tool through a GitHub repository and a DOI.

**Keywords:** Knowledge Graph Creation · Mapping Rules · Functions · Entity Alignment · Semantic Data Integration
1 Introduction

Knowledge graphs (KGs) represent the convergence among data and knowledge using networks. Albeit coined by the research community for several decades, KGs are playing an increasingly relevant role in scientific and industrial areas [11]. Years of research on semantic data management and knowledge engineering have paved the way for the integration of factual statements spread across various data sources or collected from community-maintained data sources (e.g., Wikidata [20] and DBpedia [1]). The rich spectrum of knowledge represented in existing KGs, position them as sources of background knowledge to empower data-driven processes. Nevertheless, real-world applications require accountable methods to facilitate the traceability of data management processes performed to integrate data into a KG. Thus, KG management needs to be enriched with transparent methods to understand and validate the steps performed to transform disparate data into a unified KG.

Data integration systems (DIS) [15] represent generic frameworks to define a KG in terms of a unified schema, a set of data sources, and mapping rules between concepts in the unified schema and the sources. The declarative definition of mapping languages represents a building block for tracking down a KG creation; it also facilitates reusability and modularity. Mapping languages (e.g., R2RML [5] and RML [9]) have been proposed as standards describing correspondences between the concepts in the unified schema (e.g., classes, properties, and relations) and the data sources’ attributes. Thus, by following the global as view (GAV) paradigm [15] where concepts in the unified schema are defined in terms of the sources, they enable the resolution of interoperability conflicts among data sources defined using different schemas. However, data sources may
have diverse levels of structuredness (e.g., structured, semi-structured, and unstructured), suffer from data quality issues, or present several interpretations of the same real-world entity. The resolution of these conflicts as part of the process of KG creation can be defined as Data Operators in a Data Ecosystem (DE) proposed by Capiello et al. [3]. Alternatively, mapping languages have been extended to embrace Data Operators as functions that can be included as programming scripts directly in the mapping rules [7,14,21] or can follow a declarative approach (e.g., using the Function Ontology, FnO) [6]. They offer clear benefits in comparison to ad-hoc pre- and post-processing techniques in terms of reusability and reproducibility. Nonetheless, the lack of generic frameworks to deal with mapping rules and functions complicates mapping rule design because these functions need to be also implemented.

**Our Method:** We address the problem of EA using target knowledge to solve interoperability conflicts across data sources by proposing a method named EABlock. EABlock is a computational block composed of a set of FnO functions which can be called from RML mapping rules and an efficient strategy to evaluate them. The functions in EABlock are tuned to effectively align entities in a KG with their corresponding entities in existing KGs (e.g., DBpedia [1] and Wikidata [20]) and controlled vocabularies (e.g. UMLS [2]). These functions resort to another engine for solving the tasks of name entity recognition (NER) and entity linking (EL) required for EA: any engine performing NER and EL tasks can be utilized. EABlock follows an *eager evaluation* strategy and enables the execution of the EABlock functions before the RML mapping rules are executed. This evaluation strategy defined by Jozashoori et al. [13], facilitates the transformation of RML with the EABlock functions into function-free RML mapping rules that can be executed by any RML-compliant engine without requiring any modification in the engine. EABlock has been developed and experimentally evaluated in real-world datasets collected from DBpedia, Wikidata, and UMLS. The observed outcomes suggest that EABlock functions perform EA to domain-specific and encyclopedic KGs effectively. EABlock is utilized in three international projects to integrate data into the KGs developed in these projects. The results corroborate the role that declaratively defined functions have in KG management.

This paper is structured in six additional sections. Section 2 summarizes the state of the art, and section 3 motivates and defines the problem addressed by EABlock. While section 4 provides an overview of background knowledge and concepts, section 5 formally defines the problem and describes EABlock as the solution including its proposed strategy and techniques. In section 6 the results of the experimental study are reported. Lastly, section 7 wraps up and outlines future work.

## 2 Related Work

Entity Alignment (EA) is an important solution to overcome interoperability issues while creating a knowledge graph from heterogeneous data sources. Dimou et al. [8], Michel et al. [16], and Vidal et al. [19] propose EA as a pre-processing
step, prior to the semantic enrichment and integration of data. In this case, pre-processing performs the task of EA on the whole provided data sources, independent of their involvement in the goal KG. Hence, including EA as part of pre-processing can add a considerable overhead on the knowledge graph creation pipeline. Additionally, pre-processing steps are usually developed as ad-hoc programs, which are neither declarative nor easy to maintain. SemTab\footnote{https://www.cs.ox.ac.uk/isg/challenges/sem-tab/} is an effort in benchmarking systems dealing with the tabular data to KG matching problem and present existing challenges \cite{Zeng2014}. An alternative is to perform EA after the creation of KG, at the expense of creating the same nodes multiple times across different KGs. Zeng \cite{Zeng2014} provides a comprehensive survey of available techniques to add EA in post-processing, to find the equivalent entities in different created KGs. Lastly, EA can be part of the main pipeline of semantic data integration and knowledge graph creation applying transformation functions. In other words, EA can be involved in the mapping rules that enrich raw data semantically and transform them into RDF model. In this case, EA needs to be defined as a transformation function in the mapping rules. There are different mapping languages enabling the involvement of functions as part of the mapping rules such as RML+FnO \cite{RML+ FnO}, R2RML-F \cite{R2RML-F}, FunUL \cite{FunUL}, and D-REPR \cite{D-REPR}. There also exist different engines capable of processing functions in different languages. For instance, FunMap \cite{FunMap} is able of interpreting function-based mappings in RML+FnO into equivalent function-free mappings in RML efficiently. In spite of all the value that declarative mapping languages and corresponding techniques provide, their potential applications in the task of EA are neither well explored nor appreciated. Hereupon, we aim to fill this gap by enabling and facilitating the application of EA tools as part of mapping rules using transformation functions.
3 Motivating Example

We motivate our work with a mock example from a real-world scenario illustrated in [Figure 1](#). In this scenario, the aim is to integrate four datasets obtained from different sources into a KG. The datasets consist of a) Patient data extracted from two different clinical notes provided by a general practitioner (GP) and an oncologist including the comorbidities from which the patient is suffering, and b) The drug related data extracted from DrugBank including drug-drug-interaction providing information on the possible interactions between different drugs and the impact on effectiveness of each, and drug-disorder data revealing information on list of drugs that can be prescribed for each disorder. A portion of the KG created by a naive approach can be observed in [Figure 1](#). A closer look reveals that the same disorder instance exists as three separated nodes in the graph, i.e., there is an interoperability conflict among them. The existing interoperability issue can be traced back to the raw data where I. the same disorder is represented with different names by clinical physiologists, and II. the name of the disorder is misspelled in one of the records. Another important point is regarding the connection between the instances of the generated KG and instances in available domain-specific sources (e.g., UMLS) or encyclopedic KGs (e.g., DBpedia and Wikidata) which represent the same real-world entities. More specifically, the importance of mentioned connections appears while integrating or linking the other available data/knowledge bases, which are annotated by instances of such sources (i.e., UMLS, DBpedia, and Wikidata). Both observations emphasize the importance of including EA as a module in the pipeline of KG
creation. It should be noted that following FAIR principles, transparency and reproducibility are essential requirements in pipelines of KG creation. All blocks applied as part of the main process or pre- or post-processing of KG creation should be transparent and traceable. This leads to thinking about an independent transparent module for entity alignment, using a declarative language that can be integrated in any KG creation pipeline that is compliant with the same mapping language.

4 Preliminaries

Knowledge graphs (KGs) are data structures that represent factual knowledge as entities and their relationships using a graph data model. A KG is a directed graph \( G = (O, V, E) \), where: \( O \) is a unified schema that comprises classes, properties, and relations. \( V \) is a set of nodes in the KG; nodes in \( V \) correspond to classes or instances of classes in \( O \). \( E \) is a set of directed labeled edges in the KG that relate nodes in \( V \). Edges are labeled with properties and relations in \( O \). A KG creation process can be specified in terms of a Data Ecosystem (DE). A DE is defined as a 4-tuple \( DE = \langle DataSets, DataOperators, Meta\-Data, Mappings \rangle \) where DataOperators represent a set of operators that can be executed over data in DataSets, including a set of structured or unstructured data sets. Meta-Data describes the domain of knowledge and meaning of the data residing in DataSets accordingly. Meta-Data comprises: I. Ontologies and controlled vocabularies to provide a unified view of the domain knowledge; II. Properties to describe the data quality, provenance, and access regulations; and III. descriptions of the main characteristics of data. Finally, Mappings represent the correspondences among the concepts and properties in different domain ontologies or associations between data in DataSets and the domain ontology. The same real-world entity can be represented differently in the data sources in DataSets. Interoperability issues include: Structuredness: this conflict occurs whenever data sources are described at different levels of structuredness, e.g., structured, semi-structured, and unstructured. Schematic: this interoperability conflict exists among data sources that are modeled using different schemas, e.g., different attributes representing the same concept. Domain: this interoperability conflict occurs among various interpretations of the same entity. They include: i) homonym: the same name is used to represent concepts with different meaning, and ii) synonym: distinct names are used to model the same concept. Figure illustrates the interoperability issues: structuredness between the two data sources of drug-drug interactions; schematic among the attributes of dataset 1 and 2; and domain among the names representing the hypertension. In general, KG creation pipelines include an additional pre- / post-processing block to solve the interoperability issues between data sources. However, this block can be part of the DE as a DataOperators utilizing the knowledge encoded in the Meta-Data.

KGs are expressed in the Resource Description Framework (RDF), where nodes can be resources or literals, and edges correspond to predicates. RDF resources are identified by IRIs (Internationalized Resource Identifier) or blank nodes
(anonymous resources or existential variables), while literals correspond to instances of a data type (e.g., numbers, strings, or dates). Mapping rules in DE are declaratively defined using the RDF Mapping Language (RML), an extension of the W3C-standard mapping language R2RML. RML allows for the definition of sources in different formats (e.g., CSV, Relational, JSON, and XML). An RML mapping rule, named TriplesMap, follows the global as view paradigm [15], i.e., concepts in the unified schema are defined in terms of a data source. Figure 2 presents RML TriplesMaps. A rr:subjectMap defines the resources of an RDF class in the unified schema, while a set of predicate-object maps (rr:predicateObjectMap) define the properties and relations of a class. The values of a predicate-object map can be defined in terms of a data source attribute, or as a reference or a join with the rr:subjectMap in another TriplesMap. A reference to another triples map is denoted as rr:RefObjectMap; it can be stated only between triples maps defined over the same data source. Lastly, a rr:JoinCondition represents references between TriplesMap defined on different data sources. A function can define rr:subjectMap or rr:predicateObjectMap. The Function Ontology (FnO) is used to specify functions of the type FunctionMap [6]. A DE where its Mappings comprises RML TriplesMaps with FnO functions, can be executed following two strategies. a) Lazy evaluation delays the execution of a function until when it is needed to compute a value in a TriplesMap. b) Eager evaluation executes the functions in Mappings over the data sources before these values are needed in the RML triples maps. The lazy evaluation requires an understanding of RML and FnO. Contrary, the eager evaluation enables the transformation of the RML+FnO TriplesMaps into function-free RML triples maps. This evaluation can be done beforehand, and the results can be represented as sources of the translated function-free TriplesMaps. Another advantage of an eager evaluation is that an RML-compliant engine can be used to execute the function-free TriplesMaps and create a KG. Figure 2 a) presents two RML TriplesMaps (lines 1-23) and the function DBpedia-Function is defined in lines 25-35. Following a lazy evaluation, DBpedia-Function is executed each time a new entity of the class Drug is created. This execution requires that the RML engine is able to execute functions. Also, in presence of large number of duplicates in the data sources (i.e., drug.csv), it may be executed several times. On the other hand, Figure 2 b) depicts the translation performed for eager evaluation; this approach is described by Jozashoori et al. [13]. The transformation RML TriplesMaps are evaluated over new data sources. The data source PROJECT1.csv is created from the drug.csv following well-known properties of the relational algebra (e.g., pushing down of projections and selections into the data sources); they enable not only the reduction of the size of data sources but also the elimination of duplicates. Moreover, PROJECT2.csv is created from the materialization of DBpedia-Function. The reference between the two TriplesMaps is expressed with a join condition.
Our Approach: EABlock

Problem Statement: As shown in Figure 1, a KG can comprise entities that correspond to the same real-world entity (e.g., various entities representing hypertension). We address the problem of aligning entities in a KG $G_1=(O_1,V_1,E_1)$ with entities in an existing KG $G_2=(O_2,V_2,E_2)$ efficiently. Encyclopedic KGs like DBpedia [1] or Wikidata [20], or domain-specific (e.g., UMLS [2]) correspond to KGs $G_2$ against where the alignment is performed.

Proposed Solution: Entity alignment from $G_1$ to $G_2$, $\gamma(G_1 \mid G_2)$, is defined in terms of an ideal KG, $G^* = (O^*,V^*,E^*)$, that includes the nodes and edges in $G_1$ and $G_2$ plus all the edges that relate nodes in $G_1$ with nodes in $G_2$. A solution to $\gamma(G_1 \mid G_2)$ corresponds to a maximal partial function $\zeta: V_1 \rightarrow V_2$ such that $\gamma(G_1 \mid G_2, \zeta) = \{(s_1, \text{sameAs}, \zeta(s_1)) | (s_1, \text{sameAs}, \zeta(s_1)) \in E^*\}$. DE$G_{1,2} = \langle \text{DataSets}_1, \text{DataOperators}, \text{Meta-Data}_1, \text{Mappings}_{1,2} \rangle$ defines the KG, $G_{1,2}=(O_1 \cup \{\text{sameAs}\}, V_1 \cup V_2, E_1 \cup \gamma(G_1 \mid G_2, \zeta))$. The set $\text{Mappings}_{1,2}$ is a superset of $\text{Mappings}_1$ including all triples maps that define $\zeta$ and enable the computation of $\gamma(G_1 \mid G_2, \zeta)$.

EABlock is an approach proposing a computational block to solve entity alignment over textual attributes providing techniques bridging and utilizing all components of a DE i.e., DataSets, DataOperat- or, Meta-Data, Mappings: a) EABlock links entities encoded in labels and short text to controlled vocabularies described by meta-data and resources in encyclopedic and other domain-specific KGs. For this purpose, EABlock introduces a set of operating functions resorting to an entity and relation linking tool. b) EABlock functions are defined in a human and machine-readable medium, meeting the requirements of meta-data in terms of transparency and reusability. Although the outcome of EABlock representing the aligned entities and annotations provides meta-data for the KG, the addition of EABlock functions to the meta-data of the DE equips this layer for further reproduction or maintenance of the KG with newly added

---

6 A partial function $\zeta: V_1 \rightarrow V_2$ is a function from a subset of the $V_1$. $\zeta$ is maximal in the partial ordered set of all the functions from $V_1 \rightarrow V_2$. 
EABlock functions can be easily integrated into the mappings expressing the relations among the data and the ontology using RML language, applying available extensions of the language. **EABlock** also provides an efficient evaluation strategy to materialize the calls of the functions in the mappings extending data sources and transforming mappings to function-free RML mappings that are adaptable by any RML-compliant KG creation pipeline.

As shown in Figure 3, **EABlock** composes three components: **Functions** including the signatures of the **EABlock** functions in FnO. The functions can be divided into two categories based on their domains and ranges. **Keyword-based** functions receive case-insensitive keywords as input and generate one entity as the output, and **Short text-based** functions accept a case-insensitive short text as input and output a list of entities. **Entity Alignment** performs the NER and EL tasks. This component is agnostic, i.e., any tool solving the tasks of Named Entity Recognition (NER) and Entity Linking (EL) through an API can be employed; as a proof of concept, we use Falcon2.0 [18]. **Interpreter** connects the previous two components. It follows an eager evaluation strategy of the functions and retrieves the results of the entity alignment generated by the entity alignment tool. The eager evaluation strategy gives the basis for an efficient and RML engine-agnostic execution of the **EABlock** functions. It resorts to the approach proposed by Jozashoori et al. [13] to translate the input RML+FnO TriplesMaps into function-free RML TriplesMaps. As explained before, **EABlock** creates a new data set-output dataset-materializing the functions. The output dataset comprises two attributes; input and output attributes (*attr1* and *attr2*). Depending on the category of the function, **EABlock** deploys one of the following two techniques. **a.** If the function is a **Keyword-based** function, for each input value, one record is added to the output dataset including the input value and the retrieved linked entity as the values of *attr1* and *attr2*, respectively. **b.** However, if the function is **Short text-based**, after evaluation of the function and receiving the list of linked entities, **EABlock** generates the output dataset including one record for each entity in the list of linked entities, i.e., for each entity in the list of the retrieved linked entities, one record is added to the output dataset which includes input value and the linked entity as the values of *attr1* and *attr2*, respectively. In this way, **EABlock** ensures that the generated datasets can be translated by any RML-compliant engine and result in exactly the same RDF triples; since different RML engines may have different interpretations of an RDF list. **Implementation and Application** **EABlock** approach is implemented and available as a tool. As a proof of concept, **EABlock** integrates Falcon2.0 API[7] to perform the NER and EL tasks. Falcon [17,18] is empowered with background knowledge that allows for the accurate recognition and linking of biomedical concepts. **EABlock** is developed in Python3, open-source, and available under the Apache License 2.0[8].

---

[7] https://labs.tib.eu/sdm/falconmedical/falcon2/

[8] To keep the anonymity of the paper, **EABlock** will be publicly available as a tool at the Zenodo platform, GitHub, and Dockerhub after the review procedure.
(a) The performance of a KG creation pipeline applying RocketRML.
(b) The performance of a KG creation pipeline applying SDM-RDFizer.

Fig. 4: **Efficiency**. The impact of using *EABlock* in KG creation pipelines applying two different RML-compliant engines. Baseline corresponds to the execution of entity alignment in a pre-processing stage, while *EABlock* enables the specification of this process in the RML mapping rules. As observed, *EABlock* reduces the execution time of KG creation pipelines that involve entity alignment tasks in comparison to the application of the same functions but during a pre-processing stage.

6 **Empirical Evaluation**

Our goal is to empirically assess the performance of the *EABlock* in the resolution of the problem presented in section 5. The following research questions guide our experimental study: **RQ1**) What is the impact of applying *EABlock* in KG creation in terms of execution time? **RQ2**) How does applying *EABlock* in the process of KG creation impact the quality of the result KG? **RQ3**) How sensitive to the quality of input data is *EABlock*? As a proof of concept, we set up the experiments using biomedical data. Accordingly, we rely on an API of Falcon9 that provides a filtered subset of the BK [18] omitting the resources that are not related to the biomedical domain.

6.1 **EABlock** Efficiency- RQ1

To evaluate how the performance of a KG creation pipeline may be impacted applying EABlock, we set up 24 KG creation pipelines in overall. Experiments are grouped as Baseline or *EABlock*; Baseline corresponds to the pipelines where execution of EA is in a pre-processing stage, while *EABlock* represent the KG creation pipelines in which *EABlock* enables the specification of EA in the RML mapping rules. Experiments are grouped into six categories, each category utilizing a different *DE*, i.e., all the experiments in one category have the same

---

9 [https://labs.tib.eu/sdm/biofalcon/](https://labs.tib.eu/sdm/biofalcon/)
To avoid any bias caused by the techniques applied in the development of the state-of-the-art engines, we repeat the same experiments by two different available engines including RocketRML \textsuperscript{10} and SDM-RDFizer \textsuperscript{11}. Accordingly, the experiments in one category differs in \textbf{a.} the applied RML-compliant engine and \textbf{b.} whether \textit{EABlock} is used as part of the pipeline or not.

**Datasets and Mappings.** Considering the parameters that affect the performance of KG creation pipelines \textsuperscript{4}, we define three different sets of mapping rules, which are distinguished based on the complexity of the \texttt{rr:TriplesMap}s that refers to the \textit{EABlock} transformation functions. We manipulate the complexity of the mentioned rules by having different number of \texttt{rr:RefObjectMap}s, i.e., zero, one, or two \texttt{rr:RefObjectMap} (referred to as \texttt{noROM}, \texttt{1ROM}, and \texttt{2ROM} respectively, in Figure 4). In an attempt to prevent possible effects of data volume on the results of the experiments, we generate two relatively small datasets including 1,000 and 2,000 randomly selected records. Each dataset comprises 22 attributes, two of which are referenced in the mapping rules. **Setups.** We define two KG creation pipelines, \textit{Baseline} and \textit{EABlock}, which execute the same entity alignment tasks and produce the same KG. The \textit{Baseline} pipeline evaluates RML mapping rules while the entity alignment is performed in a pre-processing step. Contrary, the \textit{EABlock} pipeline encapsulates these tasks in the \textit{EABlock} functions that are called in the RML mapping rules. **Metrics.** Execution time: Elapsed time spent by the whole pipeline to complete the creation of a KG; it is measured as the absolute wall-clock system time as reported by the \texttt{time} command of the Linux operating system. The experiments were run in an Intel(R) Xeon(R) equipped with a CPU E5-2603 v3 @ 1.60GHz 20 cores, 64GB memory and with the O.S. Ubuntu 16.04LTS. **Results.** Figure 4 illustrates the performance of two approaches of KG creations i.e., \textit{Baseline} which perform the EA as pre-processing, and \textit{EABlock} which enables the specification of EA as part of the RML mapping rules. As it can be observed in Figure 4, independent of the applied RML-compliant engine utilizing \textit{EABlock} in all KG pipelines reduces the overall execution time of the KG creation. Figure 4 demonstrates that performing EA as pre-processing is more expensive than using \textit{EABlock} as part of the main pipeline of KG creations. It can also be observed that in case of having more complex mapping rules, the impact of \textit{EABlock} in decreasing the execution time is even more considerable and significant.

### 6.2 \textit{EABlock} Effectiveness - RQ2

We define two pipelines: \textit{Baseline} and \textit{EABlock}; the \textit{Baseline} pipeline includes no entity alignment task. The aim is to evaluate the connectivity in a KG created using the \textit{EABlock} pipeline and assess RQ2. **Datasets and Mappings.** We extract data related to drugs (11,293 records), the disorders for which the drugs are prescribed (416 records), and the interactions between the drugs (1,646,836 records) from DrugBank\textsuperscript{12} (version 5.1.8). We produce three mock datasets re-

\textsuperscript{10} https://github.com/semantifyit/RocketRML
\textsuperscript{11} https://github.com/SDM-TIB/SDM-RDFizer
\textsuperscript{12} https://go.drugbank.com/
sembling normal clinical notes for cancer patients, including the data related to the comorbidities (1,322 records) and prescribed oncological (1,764 records) and non-oncological drugs (1,325 records). We create a unified schema for these datasets and a set $D_4$ of RML mappings rules to integrate them. Also, we create a set $D_5$ with all the mapping rules in, $D_4$ plus the corresponding calls to EABlock functions to execute entity alignment for drugs and disorders.

**Analysis.** Let $KG_b$ and $KG_{eablock}$ be the KGs created by the Baseline and EABlock pipelines, respectively. $KG_{eablock}$ comprises 10,339,870 RDF triples, while $KG_b$ has 10,200,209. $KG_b$ and $KG_{eablock}$ are used to create two labelled directed graphs $G_b = (V, E_b)$ and $G_{eablock} = (V, E_{eablock})$ and traditional network analysis methods are applied to determine connectivity. Vertices in $V$ keeps the classes in $KG_b$ and $KG_{eablock}$ with at least one resource; $KG_b$ and $KG_{eablock}$ have the same resources and literals. A labelled directed edge $e = (q, p, k)$ belongs to $E_b$ (resp. to $E_{eablock}$) if there are classes $Q$ and $K$ in $V$, and $q$ and $k$ are instances of $Q$ and $K$ in $KG_b$ (resp., $KG_{eablock}$) and the RDF triple $(qpk)$ belongs to $KG_b$ (resp., $KG_{eablock}$). $G_b$ and $G_{eablock}$ provide an aggregated representation of $KG_b$ and $KG_{eablock}$. Figure 5 depicts $G_b$ and $G_{eablock}; G_{eablock}$ is composed of 11 vertices and 39 directed edges. While, $G_b$ comprises 11 vertices connected by only 10 edges. Table 5c compares $G_b$ and $G_{eablock}$ in terms of graph metrics generated by Cytoscape.

**Metrics.**

- **a.** Average number of neighbors indicates the average connectivity of a vertex or node in a graph.
- **b.** Network diameter measures the shortest path that connects the two most distant nodes in a graph.
- **c.** Clustering coefficient measures the tendency of nodes who share the same connections in a graph to become connected. If a neighborhood is fully connected, the clustering coefficient is 1.0 while a value close to 0.0 means that there is no connection in the neighborhood.
- **d.** Network density measures the portion of potential edges in a graph that are actually edges; a value close to 1.0 indicates that the graph is fully connected.
- **e.** The number of connected components indicates the number of subgraphs composed of vertices connected by at least one path.

**Results.** The results reported in Figure 5c indicate the average number of neighbors in $KG_{eablock}$ comprises entities that are more connected. Moreover, the clustering coefficient is relatively low, but the CUIs annotations and links to DBpedia and Wikidata included in $KG_{eablock}$ increase the connectivity in the neighborhoods of $G_{eablock}$. In particular, eablock:Patient, eablock:DrugDisorder, eablock:Annotation, eablock:Disease, wiki:Q12136, wiki:Q11173, and dbo:Drug have a neighborhood connectivity of 8 in $G_{eablock}$. On the other hand, in $G_b$, the neighborhood connectivity of eablock:Annotation, eablock:Di-base, wiki:Q12136, wiki:Q11173, and dbo:Drug is 0, and eabloc-k:Patient and eablock:DrugDisorderInteraction is 3. These results corroborate that connectivity is enhanced as a result of the entity alignment implemented by the EABlock functions.

[https://cytoscape.org/](https://cytoscape.org/)
6.3 **EABlock Effectiveness - RQ3**

We evaluate the performance of the *EABlock* and the impact that data quality issues may have on entity alignment. We define 15 testbeds, five for DBpedia, for Wikidata, and for UMLS. **Datasets** **Gold standard:** We create the gold standard datasets by extracting biomedical instances from DBpedia (06-2010) and Wikidata (04-2019) KGs, and the UMLS (November 2020) dataset. We obtain 51,850 records from Wikidata considering the classes “Q12136 (disease)”, “Q3736076 (biological function)”, “Q43229 (organization)”, “Q514 (anatomy)”, “Q79529 (chemical substance)”, and “Q86398 (nucleic acid sequence)”. From DBpedia, we extract 43,680 records related to the classes “ChemicalSubstance”, “Disease”, “Gene”, and “PersonFunction”. We consider the whole dataset of UMLS including 3,741,395 records. However, to avoid having a huge difference between the number of records with the experiments of DBpedia and Wikidata, we randomly select 1,496,557 records from the UMLS dataset. Meanwhile, to decrease bias, for each of the five experiments, we generate a new randomly selected dataset of UMLS with the same number of records, i.e., 1,496,557 records. **Testbeds:** Five testbeds are generated by manipulating the gold standard datasets considering frequent quality issues that may exist in datasets; character capitalization, elimination, insertion, and replacement. For this purpose, for DBpedia, Wikidata, and UMLS, we create five datasets from the related gold standard dataset, but with lower quality than the gold standard. To decrease the quality of the data, we intentionally introduce misspelling errors in values of the records. The errors include: a) capitalizing all the characters of a record value; b) randomly eliminating one character from the value; c) randomly replacing one character with another randomly selected character; and d) inserting a randomly selected character to a random location in the record value. Accordingly, each of mentioned errors are introduced in 50% of the records in one of the testbeds, the other 50% of the records carry the same values as the gold standards. The last testbed created by including all four types of errors, has the lowest quality. In this dataset, each 20% out of 80% of the records involves exactly one of the four errors. Therefore, 20% of the records are error-free, in contrast to the other four test beds, in which 50% of all records are free from errors. **Metrics.** We assess the sensitivity of *EABlock* in terms of I. precision; the fraction of correct results of entity alignments from all the entity alignment results returned by *EABlock*, II. recall; the fraction of correct results of performed entity alignment from the all expected results to be returned by *EABlock*, III. F1 score; the harmonic mean of precision and recall. **Results.** [Table 1] demonstrates the results of running *EABlock* over the five configurations of error types explained in Testbeds. The entity alignment engine (Falcon) reports relatively high performance. Cleaning the BK of Falcon and filtering it to contain only resources related to the biomedical domain, as explained in section 5 reduces the ambiguity among the resources in the BK. This plays a major role in having such high performance. Also, having the input of the entity alignment module as keywords without any noise, e.g., stopwords, helps the module recognize and link the labels precisely. Moreover, [Table 1] sug-
(a) Directed Labelled Graph $G_b$

(b) Directed Labelled Graph $G_{eablock}$

Fig. 5: Connectivity Analysis. Baseline and $EABlock$ pipelines generate $KG_b$ and $KG_{eablock}$, respectively. $KG_b$ and $KG_{eablock}$ have the same classes and entities. However, $KG_b$ does not include the entity alignments to UMLS, Wikidata, and DBpedia added to $KG_{eablock}$. $G_b$ and $G_{eablock}$ are directed labelled graphs that provide an aggregated representation of $KG_b$ and $KG_{eablock}$. The values of the graph metrics corroborate that connectivity is increased by entity alignment performed by the $EABlock$ pipeline.

gests that the used entity alignment module is able to overcome the proposed error types. There are records for which $EABlock$ fails to return the expected linked entity based on the gold standard. These failures are mostly caused by reasons other than the introduced error types; we enumerate a couple of these examples with possible explanations: a) there are cases that the retrieved linked entities are correct, although their identifiers differ from those in the gold standard. For instance, for the keyword “malignant histiocytosis” $EABlock$ retrieves “Q164952”\footnote{https://www.wikidata.org/wiki/Q164952} from Wikidata, while, in the gold standard the Wikidata identifier for the same keyword is “Q52962465”\footnote{http://www.wikidata.org/entity/Q52962465}. However, both identifiers lead to the same entry in Wikidata; for some keywords, more than one identifier exists in such KGs. b) Another example of unexpected retrieved linked entities, can be observed in case of having long combinations of keywords, such as “early infantile epileptic encephalopathy 19”. In this case, $EABlock$ links the first entity that can be recognized by the first couple of keywords; “Q61913448”\footnote{https://www.wikidata.org/wiki/Q61913448} which belongs to the label “early infantile epileptic encephalopathy 37”. The same failure case can be observed in retrieval from DBpedia as well; for the keyword “Chronic leukemia” $EABlock$ retrieves a link to “B-cell_chronic_lymphocytic_leukemia”\footnote{https://dbpedia.org/page/Chronic_lymphocytic_leukemia} while based on the gold standard the correct link is “Chronic_leukemia”\footnote{https://dbpedia.org/page/Chronic_leukemia}. 
6.4 Discussion

As is emphasized in previous sections, EABlock can be utilized in any KG creation pipeline that applies an RML-compliant engine for the tasks of mapping rules translation and RDF triples generation. Although there is a W3C community to tackle KG creation using R2RML and RML mapping languages, there is a lack of formal definitions, which is a barrier and impacts on general adoption of these engines. Another aspect of being agnostic is the application of a tool that can perform the NER and EL tasks. Despite all the values this independence brings, the performance of EABlock is subject to change based on the choice of the entity alignment tool.

Table 1: Effectiveness. The EABlock performance is assessed in 15 datasets. The aligned entities are compared with the resources of the original labels.

| Text Error Type          | Precision | Recall | F1 Score |
|--------------------------|-----------|--------|----------|
| UMLS                     |           |        |          |
| Capitalization of all characters | 1.0       | 0.97   | 0.99     |
| Elimination of a character     | 0.99    | 0.74   | 0.85     |
| Replacement of a character    | 0.99    | 0.75   | 0.85     |
| Insertion of a new character  | 0.99    | 0.50   | 0.66     |
| Combination of all 4 errors   | 1.0      | 0.78   | 0.88     |
| DBpedia                   |           |        |          |
| Capitalization of all characters | 0.78   | 0.78   | 0.78     |
| Elimination of a character     | 0.78   | 0.78   | 0.78     |
| Replacement of a character    | 0.98    | 0.98   | 0.98     |
| Insertion of a new character  | 0.78    | 0.78   | 0.78     |
| Combination of all 4 errors   | 0.78    | 0.78   | 0.78     |
| Wikidata                   |           |        |          |
| Capitalization of all characters | 0.99   | 0.99   | 0.99     |
| Elimination of a character     | 0.99   | 0.99   | 0.99     |
| Replacement of a character    | 0.99    | 0.99   | 0.99     |
| Insertion of a new character  | 0.99    | 0.99   | 0.99     |
| Combination of all 4 errors   | 0.99    | 0.99   | 0.99     |

7 Conclusions and Future Work

The diverse interoperability issues existing in textual data and the demand of having a transparent, traceable, and efficient pipeline of KG creation led us to introduce EABlock. EABlock is an approach to solve entity alignment problems by capturing knowledge from existing KGs while keeping the procedure transparent.

https://www.w3.org/community/kg-construct/
and traceable. With an eager evaluation strategy and efficient translation of mapping rules into function-free rules, EABlock ensures not to sacrifice the efficiency at the cost of reproducibility. The observed experimental results show the benefits of grounding solutions for KG creation in the well-established problems like NER, NL, and data integration systems. Thus, EABlock broadens the repertory of approaches for KG creation and provides the basis for developing real-world KGs. Our vision is that EABlock will be the starting point for the development of FnO functions which can be made available following the FAIR principles. In the future, we will empower the EABlock with a set of functions to overcome interoperability problems across multilingual and unstructured datasets.

Acknowledgments. This work has been partially supported by the EU H2020 RIA funded project CLARIFY with grant agreement No 875160 and P4-LUCAT No 53000015.

References

1. S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In Proceedings of ISWC + ASWC, 2007.
2. O. Bodenreider. The unified medical language system (umls): integrating biomedical terminology. Nucleic Acids Res, 1(32), 2004.
3. C. Capiello, A. Gal, M. Jarke, and J. Rehof. Data ecosystems: sovereign data exchange among organizations (dagsuhl seminar 19391). In Dagstuhl Reports, volume 9. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2020.
4. D. Chaves-Fraga, K. M. Endris, E. Iglesias, O. Corcho, and M.-E. Vidal. What are the Parameters that Affect the Construction of a Knowledge Graph? In OTM Conferences, 2019.
5. S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF Mapping Language, W3C Recommendation 27 September 2012. W3C, 2012.
6. B. De Meester, A. Dimou, R. Verborgh, and E. Mannens. An ontology to semantically declare and describe functions. In European Semantic Web Conference, pages 46–49. Springer, 2016.
7. C. Debruyne and D. O’Sullivan. R2RML-F: Towards Sharing and Executing Domain Logic in R2RML Mappings. In LDOW Workshop, 2016.
8. A. Dimou, G. Haesendonck, M. Vanbrabant, L. De Vocht, R. Verborgh, S. Latré, and E. Mannens. Ilastic: Linked data generation workflow and user interface for iminds scholarly data. In Semantics, Analytics, Visualization, pages 15–32. Springer, 2017.
9. A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, and R. Van de Walle. Rml: a generic language for integrated rdf mappings of heterogeneous data. In Ldow, 2014.
10. S. Geisler, M. Vidal, C. Cappiello, B. F. Lóscio, A. Gal, M. Jarke, M. Lenzerini, P. Missier, B. Otto, E. Paja, B. Pernici, and J. Rehof. Knowledge-driven data ecosystems towards data transparency. Accepted at the Special Issue of Data Transparency, ACM Journal Data and Information Quality, 2021.
11. C. Gutiérrez and J. F. Sequeda. Knowledge graphs. Commun. ACM, 64(3):96–104, 2021.
12. E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas, and V. Cutrona. Results of semtab 2020. In CEUR Workshop Proceedings, volume 2775, pages 1–8, 2020.
13. S. Jozashoori, D. Chaves-Fraga, E. Iglesias, M.-E. Vidal, and O. Corcho. Funmap: Efficient execution of functional mappings for knowledge graph creation. In International Semantic Web Conference, 2020.
14. A. C. Junior, C. Debruyne, R. Brennan, and D. O’Sullivan. FunUL: a method to incorporate functions into uplift mapping languages. In Intern. Confer. on Information Integration and Web-based Applications and Services, 2016.
15. M. Lenzerini. Data integration: A theoretical perspective. In ACM Symposium on Principles of Database Systems, 2002.
16. F. Michel, F. Gandon, V. Ah-Kane, A. Bobasheva, E. Cabrio, O. Corby, R. Gazzotti, A. Giboin, S. Marro, T. Mayer, et al. Covid-on-the-web: Knowledge graph and services to advance covid-19 research. In International Semantic Web Conference, pages 294–310. Springer, 2020.
17. A. Sakor, I. O. Mulang, K. Singh, S. Shekarpour, M. Vidal, J. Lehmann, and S. Auer. Old is gold: Linguistic driven approach for entity and relation linking of short text. In Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, 2019.
18. A. Sakor, K. Singh, A. Patel, and M. Vidal. Falcon 2.0: An entity and relation linking tool over wikidata. In International Conference on Information and Knowledge Management (CIKM), 2020.
19. M.-E. Vidal, K. M. Endris, S. Jazashoori, A. Sakor, and A. Rivas. Transforming heterogeneous data into knowledge for personalized treatments—a use case. Datenbank-Spektrum, 19(2), 2019.
20. D. Vrandecic and M. Krötzsch. Wikidata: a free collaborative knowledgebase. Commun. ACM, 57(10):78–85, 2014.
21. B. Vu, J. Pujara, and C. A. Knoblock. D-REPR: A Language for Describing and Mapping Diversely-Structured Data Sources to RDF. In Intern. Confer. on Knowledge Capture, 2019.
22. M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Bank, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al. The fair guiding principles for scientific data management and stewardship. Scientific data, 3(1):1–9, 2016.
23. K. Zeng, C. Li, L. Hou, J. Li, and L. Feng. A comprehensive survey of entity alignment for knowledge graphs. AI Open 2021.