Contextual Propagation of Properties for Knowledge Graphs

A Sentence Embedding Based Approach

Pierre-Henri Paris[1] [0000-0002-9665-1187], Fayçal Hamdi[1] [0000-0002-5314-1419], Nobal Niraula[2], and Samira Si-said Cherfi[1]

[1] CEDRIC, Conservatoire National des Arts et Métiers, Paris, France
pierre-henri.paris@upmc.fr, faycal.hamdi@cnam.fr, samira.cherfi@cnam.fr
[2] Nowa Lab, Alabama, USA
nobal@nowalab.com

Abstract. With the ever-increasing number of RDF-based knowledge graphs, the number of interconnections between these graphs using the *owl:sameAs* property has exploded. Moreover, as several works indicate, the identity as defined by the semantics of *owl:sameAs* could be too rigid, and this property is therefore often misused. Indeed, identity must be seen as context-dependent. These facts lead to poor quality data when using the *owl:sameAs* inference capabilities. Therefore, contextual identity could be a possible path to better quality knowledge. Unlike classical identity, with contextual identity, only certain properties can be propagated between contextually identical entities. Continuing this work on contextual identity, we propose an approach, based on sentence embedding, to find semi-automatically a set of properties, for a given identity context, that can be propagated between contextually identical entities. Quantitative experiments against a gold standard show that our approach achieved promising results. Besides, the use cases provided demonstrate that identifying the properties that can be propagated helps users achieve the desired results that meet their needs when querying a knowledge graph, i.e., more complete and accurate answers.

Keywords: RDF, contextual identity, property propagation, knowledge graph, linked data, sentence embedding

1 Introduction

Open and RDF-based knowledge graphs (KGs), like prominent Wikidata[3] or DBpedia[4], are continuously growing in terms of size and usage. Consequently, the number of entities described in those KGs leads to a problem for both data publishers and data users: **how to know if two entities are the same or not?** According to Noy et al. [18], this question remains one of the top challenges

[3] https://www.wikidata.org
[4] https://wiki.dbpedia.org/

in knowledge graphs industry. To interlink KGs, the *owl:sameAs* property has been defined by the W3C[5] in 2004 to link entities that are the same. Indeed, a (real world) object is described across several KGs, and those descriptions are linked thanks to the *owl:sameAs* property. However, the semantic definition of *owl:sameAs* is very strict. It is based on Leibniz's identity definition, i.e., the identity of indiscernibles: $\forall x, \forall y(\forall p, \forall o, (\langle x, p, o\rangle \textit{ and } \langle y, p, o\rangle) \rightarrow x = y)$. And its converse, the indiscernibility of identicals: $\forall x, \forall y(x = y \rightarrow \forall p, \forall o, (\langle x, p, o\rangle \rightarrow \langle y, p, o\rangle))$. Hence, two entities are considered identical if they share all their ⟨ *property,value* ⟩ pairs in all possible and imaginable contexts. In other words, two entities are identical if **all their properties are indiscernibles** for each value.

Once an identity link is stated between two entities, it is possible to use $\langle property, value\rangle$ pairs from one entity to another. However, it is a very strong assertion to state that two objects are the same whatever the context. From a philosophical point of view, there are multiple counterarguments to the definition of Leibniz's identity. For example, if we consider two glasses from the same set of glasses, they are indiscernible from each other and yet they are two different physical objects. Similarly, is a person the same as he or she was ten years ago?

It is also a technical problem because of the open-world assumption [6], on the one hand, and on the other hand, because of what a data publisher has in mind that could be different from what the user expects when using data. Besides, when data is published, it is "almost" impossible to know the **consensus** behind the decision of creating an *owl:sameAs* link. Several works such as [11] and [5] have demonstrated that the use of *owl:sameAs* was inadequate. Indeed, established links might be considered as true only in specific contexts.

As a first intuition, a contextual identity between two entities might be seen as a subset of properties $\mathit{\Pi}$ for which these entities share the same values for each $p \in \mathit{\Pi}$.

Example 1. Two different generic drugs *Drug1* and *Drug2* can be identical when considering the active ingredient. If a KG contains the triples ⟨ *Drug1 activeIngredient Molecule1* ⟩ and $\langle Drug2\ activeIngredient\ Molecule1\rangle$, then $Drug1 \equiv_{activeIngredient} Drug2$ when the context is $activeIngredient$.

One of the core features of *owl:sameAs* is to be able to **propagate all properties** from an entity to other identical entities. Hence, *owl:sameAs* allows to discover more knowledge and to increase completeness. In the same way, contextual identity must help to discover **more knowledge and to increase completeness**, but only under specific circumstances. So, to be useful, a contextual identity must specify what is happening with properties that are not part of the context. In other words, **an identity context must have propagating properties**.

Example 2. Following the example 1, stating only $Drug1 \equiv_{activeIngredient} Drug2$ has a limited interest, if we do not know what to do with other properties besides *activeIngredient*. Considering the context *activeIngredient*, the property

[5] https://www.w3.org/TR/owl-ref/

targetDisease is propagating, and if the statement ⟨ *Drug1 targetDisease Disease1* ⟩ exists then we can state that $\langle Drug2\ targetDisease\ Disease1\rangle$. But if we consider the property *excipient* as context, then the property *targetDisease* is not propagating.

Moreover, the ability to propagate a property between entities depends on the context, i.e., the same property might be propagating in a context C_1 and not propagating in a context C_2 as illustrated in Example 2.

Research questions: With a given identity context between two entities, how to find properties that can be propagated? Is it possible to find propagating properties (semi-)automatically?

In this paper, based on the context definition of Idrissou et al. [14], we propose an approach to **find propagating properties** to facilitate knowledge discovery for users. Instead of manually listing the propagating properties as in Idrissou et al. [14], we automatically identify the propagating properties for a given context using semantic textual similarity, significantly reducing burden to users. The semantic similarity is based on the sentence embeddings corresponding to the textual descriptions of the properties. We validated our approach through quantitative and qualitative experiments.

The rest of the paper is organized as follows. In following section, we present the related work. In Section 4, we present our approach. In Section 5, we present the experiments we have conducted. Finally, we conclude and define the next directions for our future work in Section 6.

2 Related work

In the first part of this section, we describe papers that pointed out the problems raised by the *owl:sameAs* usage. In the second part, we discuss the proposals that tackle these problems.

2.1 Identity Crisis

As early as 2002, Guarino and Welty [10] raised the issue of identity for ontologies. Especially when time is involved, stating that two things are identical became a philosophical problem. The authors proposed to involve in identity only essential properties, i.e., a property that cannot change. As described in Horrocks et al. [13], the *owl:sameAs* property purpose is to link two entities that are strictly the same, i.e., both entities are identical in every possible context. *owl:sameAs* has a strict semantics allowing to infer new information. Many existing tools produce such *owl:sameAs* links [9], and several surveys are available to this end [1, 9, 17].

However, none of these approaches consider contextual identity links. Their purpose is to discover identity links that allegedly always hold. This is, from a philosophical point of view, hard to obtain as underlined by Leibnitz's identity definition. Indeed, as stated for example in Halpin et al. [11] or Ding et al. [5], because of the strict semantic of *owl:sameAs*, the burden of data publishers might be too heavy. As a matter of fact, *owl:sameAs* links are not often adequately used. Some might be simply wrong, and, more insidiously, some might be context-dependent, i.e., the *owl:sameAs* link does not hold in every possible context because it is hard to obtain a consensus on the validity of a statement. What a data modeler means may not be what a data user expects. This misuse of *owl:sameAs* is often referred to as the "identity crisis" ([11]).

2.2 Contextual Identity

Beek et al. [2] addressed this issue by constructing a lattice of identity contexts where contexts are defined as sets of properties. All entities belonging to a context share the same values for each property of this context. Hence, a context is a set of indiscernible properties for an entity. However, the authors do not give indications about the usage of properties not belonging to such contexts. Raad et al. [19] proposed an algorithm named DECIDE to compute contexts, where identity contexts are defined as sub-ontologies. Nevertheless, as in the first work, properties of entities that are not in the sub-ontology are ignored. So, in both previous works, there is a limitation of properties that do not belong to a context. This limitation cripples the interest of using such approaches. Indeed, one of the goals of an identity context is to define an identity relation between two entities to use information about one on the other. The solution by Idrissou et al. [14] involves such propagation of properties, and thus, increases completeness of an entity according to a context. However, this proposal requires users to provide both the propagating and indiscernible properties as input. Hence, it leaves the burden to the user to identify and provide context and properties.

In this work, we propose to remove this burden partially from the user, i.e., to **semi-automatically compute the propagation set of properties given an indiscernibility set of properties**. For this, we will use sentence embedding (presented in Section 4.3) to compute the embeddings of properties using their descriptions to discover the **propagating properties** with respect to a given identity context (as defined in [14]).

3 Motivation

Sometimes, real-world entities may be close regarding their properties but not the same. For example, the French capital, Paris, is both a city and a department (an administrative subdivision of the French territory). While considering that the city and the department are the same concerning their geography, they are two distinct entities administratively (or legally) speaking, i.e., they are not considered the same per *owl:sameAs*. Now, suppose both Paris are represented in a KG as distinct entities, and both are linked to (possibly distinct) movie theaters. If one wants to retrieve movie theaters located in the city of Paris, results will not be complete if some of them are linked to the department (see Figure 1).

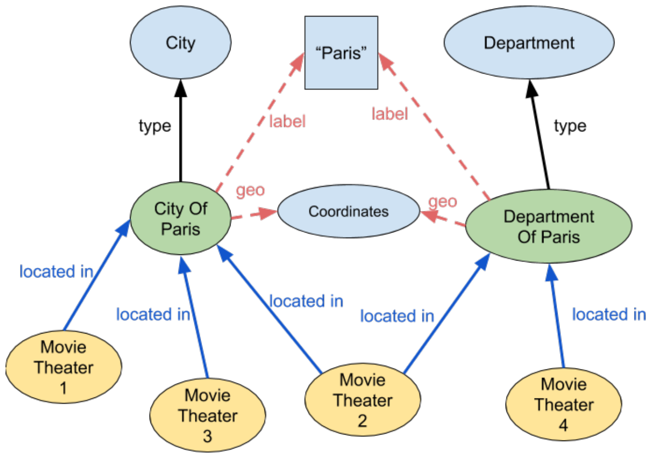

Fig. 1. Excerpt of a KG about Paris, France. The properties in red are indiscernible for both the city and the department. The properties in blue are propagating given the red properties are indiscernible.

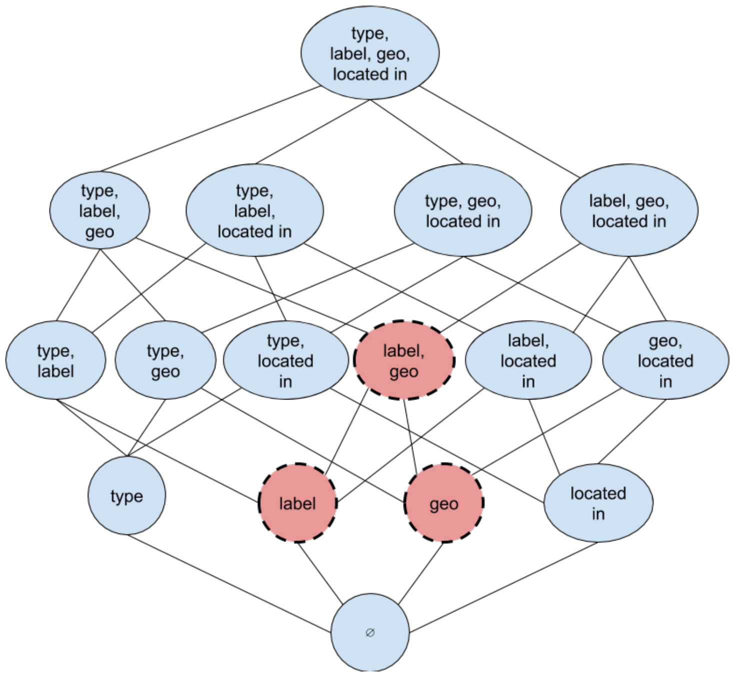

Fig. 2. Simplified identity lattice from Figure 1: each node is an indiscernible set of properties. Only the red nodes have similar entities.

A French citizen might know this ground truth, but how to allow an automated agent to discover this fact? Contextual identity is a possible answer to this question, i.e., a set of properties for which values are the same for both entities. Considering the present example, both Paris (city and department) are geographically the same and some properties related to geography might be **propagated**. In Figure 1, the dotted red properties (*geo* and *label*) are indiscernible (have the same values) and the *located in* properties are propagating. Although the two entities do not share the same values for the *located in* property, this one is related to the geographic context. Indeed, for a human agent, the *located in* property might be obviously propagated between the two entities. While we expected to have the four movie theaters located in Paris, a query on the City of Paris will only return movie theaters 1, 2 and 3 (see Figure 1).

Thus, discovering such contexts of identity between entities, might improve completeness of query results. Our intuition is inspired by Tobler's first law [23], that is: "*Everything is related to everything else, but near things are more related than distant things.*" Therefore, **we hypothesize that, from a semantic point of view, the closer a property is to the identity context, the more likely it could be a good candidate for propagation**. In the previous example, *located in* clearly refers to a geographic fact, and the context of identity is about geography since it is composed of geographical coordinates. So, the idea is to **compute a semantic distance between indiscernible properties** and **candidate properties for propagation**. Consequently, numbers, and in our case numerical vectors, are best suited to compute this distance. A numerical representation of the textual description of each property through its *rdfs:comment* or *schema:description* can provide a basis to get this vector. In most KGs, properties are described with such sentences. For example, 99% of properties in Wikidata have descriptions. Sentence embeddings of property descriptions output numerical vectors such that semantically similar descriptions appear closer in the vector space.

4 Approach

4.1 Preliminaries

As mentioned in Section 2, several proposals have been made to define an identity context. We choose the one from Idrissou et al. [14] since it is the only one that considers the propagation of properties. They give the following definition of the identity context:

Definition 1. ***(Identity Context)*** *An identity context $\mathcal{C} = (\Pi, \Psi, \approx)$ is defined by two sets of properties (Π and Ψ) and an* ***alignment procedure*** *($\approx$). Π is the* ***indiscernibility set*** *of properties (equation 1) and Ψ is the* ***propagation set*** *of properties (equation 2). In the following, x and y are entities.*

$$\begin{gathered} x =_{(\Pi,\Psi,\approx)} y \leftrightarrow \forall (p_1, p_2) \in \Pi^2 \text{ with } p_1 \approx p_2 \\ \text{and } \forall v_1, v_2 \text{ with } v_1 \approx v_2 : \langle x, p_1, v_1 \rangle \leftrightarrow \langle y, p_2, v_2 \rangle \end{gathered} \tag{1}$$

$$x =_{(\Pi,\Psi,\approx)} y \rightarrow \forall (p_1, p_2) \in \Psi^2 \textit{ with } p_1 \approx p_2 \\ \textit{and } \forall v_1, v_2 \textit{ with } v_1 \approx v_2 : \langle x, p_1, v_1\rangle \leftrightarrow \langle y, p_2, v_2\rangle \tag{2}$$

Moreover, we define the ***level of a context*** $|\Pi_{\mathcal{C}}|$ *as the number of its indiscernible properties.*

In the case where similar entities according to an identity context belong to the same KG, it is not necessary to have an alignment procedure.

An entity can have several identity contexts, depending on properties in the indiscernibility set Π. Indeed, two different combinations of properties can give different sets of similar entities. The identity lattice of all identity contexts of an entity e is defined as follow:

Definition 2. ***(Identity Lattice)*** *An identity lattice $\mathcal{L}$ is a lattice, where each element is an identity context. The set inclusion between indiscernibility set of properties of each context is the binary relation responsible for the partial order.*

The last notion is the seed of a lattice or a context that we define as follows:

Definition 3. ***(Seed of a lattice or a context)*** *Each context of a lattice is constructed from the* ***same entity*** *e. This entity e is called the seed of the lattice.*

As per Definition 2, to build an identity lattice, we need to start from a seed, despite the fact that the lattice could potentially be valid with another seed (see Figure 2).

Now that we have defined the necessary concepts, we will explain the core of our approach.

4.2 Computation of contexts

We present Algorithm 1 that computes an identity lattice. It takes as input the seed entity, the source KG to which the seed belongs, the target KG (possibly the same as the source KG) and an alignment procedure if the two KGs are distinct. The main idea is to start by computing level one identity contexts with each seed's property and finally combine those contexts to obtain upper-level identity contexts. When building a context, its first part is its indiscernibility set, from which we then get similar entities, to obtain candidate properties for propagation and, in the end, propagating properties.

The first step, line 3, is to compute all level 1 identity contexts (see Definition 1). Indeed, for each property p of the seed, there is exactly one identity context (its indiscernibility set is $\Pi = \{p\}$). Later, identity contexts with only one indiscernibility property will be merged to give identity contexts of higher-levels. Next, we retrieve similar entities $entities_p$ to the seed that have the same value(s) for the given property p. If p is multi-valuated, then entities in $entities_p$ are similar to the seed for all values o such that $\langle seed\ p\ o\rangle$. It is worth noting that, when filling $entities_p$, we search only entities that have the same type(s)

```
   Data: $\mathcal{KG}_1$: the source KG, $\mathcal{KG}_2$: the target KG, $seed$: an entity of $\mathcal{KG}_1$, $\approx$: an
         alignment procedure between $\mathcal{KG}_1$ and $\mathcal{KG}_2$
   Result: $\mathcal{L}$: a lattice of identity contexts between the seed and entities in the
           target KG
1  $\mathcal{L} = \emptyset$;
   /* Get all explicit and implicit types of the seed                    */
2  $\mathcal{T}_{seed} = \{t : \langle seed\ rdf{:}type\ t \rangle \in \mathcal{KG}_1\}$;
   /* the following will create all contexts of the level 1 (with only
      one indiscernible property)                                        */
3  for each property $p$ of seed do
4      $candidateEntities = \emptyset$;
5      for each value $o$ such as $\langle seed\ p\ o \rangle \in \mathcal{KG}_1$ do
           /* $entities_{p,o}$ is the set of indiscernible entities with seed
              with respect to the $p, o$ pair                            */
6          $entities_{p,o} = \{e : (\exists (p', o'), p' \approx p, o' \approx o, \langle e\ p'\ o' \rangle \in \mathcal{KG}_2) \wedge (\exists t \in$
            $\mathcal{T}_{seed}, t' \approx t, \langle e\ rdf{:}type\ t' \rangle \in \mathcal{KG}_2)\}$;
7          if $entities_{p,o} \neq \emptyset$ then
8              $candidateEntities = candidateEntities \cup \{entities_{p,o}\}$;
9          end
10     end
       /* $entities_p$ is the set of indiscernible entities with seed with
          respect to the property $p$                                    */
11     $entities_p = \bigcap candidateEntities$;
12     $\Psi = getPropagationSet(seed, entities_p, \{p\})$;
13     if $\Psi \neq \emptyset$ then
14         $\Pi = \{p\}$;
15         $\mathcal{C} = (\Pi, \Psi, \approx)$;
16         $\mathcal{L} = \mathcal{L} \cup \mathcal{C}$;
17     end
18 end
   /* Now we can combine contexts of the same level                      */
19 return $constructUpperLevels(\mathcal{L}, \mathcal{KG}_1, \mathcal{KG}_2, seed, \approx)$
```

Algorithm 1: createLattice: calculate identity lattice of an entity.

with the seed. This is because we want to avoid absurd results, e.g., comparing a person with an airplane. It also has the advantage of lowering the number of possible identity contexts to compute. Finally, based on $entities_p$, we compute the propagation set Ψ (line 8) as explained in the following section (Section 4.3).

The second step (see Algorithm 2) is to compute upper-level identity contexts based on those from level 1. The loop (line 2) of the algorithm calculates these upper-levels by combining contexts of the same level, and stops when it cannot construct new upper-level identity contexts. This calculation is based on an identity lattice operator, which is the set inclusion on indiscernibility sets. For example, a level 2 context is built on two contexts from level 1. Again, to lower the number of possible identity contexts to compute, if there is no similar entity to the seed for a given context C_i, there is no need to compute higher-level contexts based on C_i.

```
Data: $\mathcal{L}$: the lattice with only level one contexts, $\mathcal{KG}_1$: the source KG, $\mathcal{KG}_2$:
      the target KG, seed: an entity of $\mathcal{KG}_1$, $\approx$: an alignment procedure
      between $\mathcal{KG}_1$ and $\mathcal{KG}_2$
Result: $\mathcal{L}$: a lattice of identity contexts between the seed and entities in the
        target KG
/* $lvl$ is the current level in the lattice */
1  $lvl = 1$;
2  while $\emptyset \notin \mathcal{L}$ do
3      $contexts = \emptyset$;
4      for $(\mathcal{C}_1, \mathcal{C}_2) \in \{(\mathcal{C}_i, \mathcal{C}_j) \in \mathcal{L} \times \mathcal{L} : |\Pi_{\mathcal{C}_i}| = |\Pi_{\mathcal{C}_j}| = lvl, i > j\}$ do
5          $\Pi = \Pi_{\mathcal{C}_1} \cup \Pi_{\mathcal{C}_2}$;
           /* $getEntities$ function gives the set of entities that are
              similar under the given identity context in the given KG
              */
6          $entities = getEntities(\mathcal{C}_1, \mathcal{KG}_2) \cap getEntities(\mathcal{C}_2, \mathcal{KG}_2)$;
7          if $entities \neq \emptyset$ and $\Pi \notin \mathcal{L}$ then
8              $\Psi = getPropagationSet(seed, entities, \Pi)$;
               /* see Algo. 3 */
9              if $\Psi \neq \emptyset$ then
10                 $\mathcal{C} = (\Pi, \Psi, \approx)$;
11                 $contexts = contexts \cup \mathcal{C}$;
12             end
13         end
14     end
15     $\mathcal{L} = \mathcal{L} \cup contexts$;
16     $lvl = lvl + 1$;
17 end
18 return $\mathcal{L}$
```

Algorithm 2: constructUpperLevels: calculate upper-levels of the identity lattice of an entity.

4.3 Propagation set using sentence embedding

Our approach for computing propagation set (Line 8 in Algorithm 2) is elaborated in Algo. 3. It is based on sentence embedding which maps a sentence to a numerical vector. Ideally, semantically close sentences appear nearby in the numerical vector space.

Sentence embedding is a technique that maps a sentence to a numerical vector. Ideally, semantically close sentences are represented by close vectors in the numerical space considered. The reasons behind using sentence embedding instead of a more classical distance measures, e.g., the edit distance, RDF graph embedding like RDF2Vec [20], or an ontological alignment technique are: *(i)* classical string distances ignore sentence semantics, *(ii)* RDF graph embedding techniques are not yet adapted to such task, and *(iii)* ontological alignment techniques align pairwise properties and not sets of properties. Sentence embedding is widely used in several tasks such as computing semantic similarities between two texts. An encoder derives sentence embeddings, to capture the semantics of

```
Data: seed: the entity that generated $\Pi$,
entities: set of entities similar to seed with respect to $\Pi$,
$\Pi$: an indiscernibility set
Result: $\Psi$: a propagation set
/* computation of the embeddings of each property in $\Pi$ by using
   one of the encoder                                              */
1  $indiscernibilityEmbeddings \leftarrow getEmbeddings(\Pi)$;
2  $meanVector \leftarrow mean(indiscernibilityEmbeddings)$;
/* getCandidateProperties function returns the set of all candidate
   properties for propagation                                      */
3  $candidates \leftarrow getCandidateProperties(\Pi, \{seed\} \cup entities)$;
/* then compute their embeddings                                   */
4  $candidatesEmbeddings \leftarrow getEmbeddings(candidates)$;
5  $\Psi \leftarrow \emptyset$;
6  for $candidateVector$ in $candidatesEmbeddings$ do
7      $similarity \leftarrow cosineSimilarity(candidateVector, meanVector)$;
8      if $similarity \geq threshold$ then
9          $\Psi \leftarrow \Psi \cup \{candidateVector\}$;
10     end
11 end
12 return $\Psi$
```

Algorithm 3: getPropagationSet: calculate the propagation set.

a language, from a large text corpus. State-of-the-art encoders include Universal Sentence Encoder [3], GenSen [22] and InferSent [4]. A lot of attention has been given to sentence embeddings lately.

As presented in Section 1, our intuition, based on Tobler's first law, is that a propagation set of properties can be found given an indiscernibility set, if vectors of descriptions of those two sets are close enough. In this work, we propose to use property descriptions (e.g., *rdfs:comment* or *schema:description* as "*standard plug type for mains electricity in a country*") to find properties that are semantically related and consequently good candidates for propagation for a given indiscernibility set Π. For example, in Wikidata, the property "director" has the follow description: "director(s) of film, TV-series, stageplay, video game or similar". Descriptions are mainly composed of one sentence. Most of the properties are described with such annotations, e.g., properties of Wikidata are annotated with an english *schema:description* at 98.9%. For the embedding computation, any of the previously described encoders can be used.

Algorithm 3 presents our proposal to compute Ψ given a Π. It takes as input three parameters: a seed (an entity), a set of property built from the seed (indiscernibility set Π), and a set of entities that are similar to the seed with respect to Π. The computation of Π is presented in the previous section (see Algorithm 1).

First, for each property in the indiscernibility set Π, we calculate its representational vector. Then, we compute the mean vector that represents the indiscernibility set. Similarly, we consider each property of the seed or its similar entities, and compute their representational vectors. Therefore, on the one hand, we have one vector that represents the set of indiscernibility and, on the other hand, we have n vectors for the n properties that are candidates for propagation. Properties of similar entities (with respect to the indiscernibility set Π) are also considered as candidates since possibly one of them can have a propagating property that the seed does not have.

Then we loop on each candidate property to compute a cosine similarity [21] between each candidate vector and the mean vector representing the indiscernibility set Π. If the cosine similarity is high enough (above a specified threshold as explained in the following section) the candidate property is considered as a propagating property.

5 Experimental Results

For evaluation, we first implemented our approach, and then we present several SPARQL queries that benefited from our approach.

5.1 Implementation and set-up

We implemented our approach in Python. For the sake of reproducibility, the code is made available on a GitHub repository[6]. As mentioned earlier, we used three sentence embedding approaches, namely *InferSent*[7], *GenSen*[8] and *Universal Sentence Encoder*[9]. We used an HDT file (see [16] and [8]) that contains a dump of the last version of Wikidata[10]. The computer we used had an i7 processor and 32 GB of RAM. As an indication, the complete calculation of the identity lattice for an entity such as the city of Paris, France takes about 1396 ms. It has more than 1000 property-object pairs and, in Wikidata, the mean number of property-object pairs is about 60. Thus, it is a rather large entity and this approach could scale well.

5.2 Quantitative Study

The goal of quantity study is to evaluate how well the proposed approach can retrieve the propagating properties specified in Ψ, given the indiscernibility set of properties Π for each identity context (Π, Ψ, $\equiv$). Since there is not a prior work or dataset that we can leverage to evaluate our algorithm, we manually constructed a gold standard dataset from the Wikidata KG that is known for its high data quality [7].

The dataset consists of 100 identity contexts where each context contains the indiscernibility set of properties Π and the propagation set of properties Ψ.

[6] https://github.com/PHParis/ConProKnow
[7] https://github.com/facebookresearch/InferSent
[8] https://github.com/Maluuba/gensen
[9] https://tfhub.dev/google/universal-sentence-encoder/2
[10] http://gaia.infor.uva.es/hdt/wikidata/wikidata2018_09_11.hdt.gz

We do not need an alignment procedure ($\equiv$) specified for identity contexts since both source and target KGs are the same i.e., the Wikidata KG. To test the performance across different classes, identity contexts were constructed across five diverse class types: country, comics character, political party, literary work, and film. We randomly selected 20 entities from each class type and computed their identity lattices. One context was selected from each lattice. The propagation set of properties (Ψ) for each context was manually identified by looking at its indiscernibility set of properties Π.

Evaluation: We compared our algorithm with a baseline system that computes, using Jaccard index (JI) [15], a similarity score (range 0-1) between a candidate property and each property in Π of a given identity context. If the mean similarity score is above a specified threshold, we considered the candidate property as a propagating property for this context.

We evaluated the performance of a model using standard $Precision = \frac{tp}{tp+fp}$, $Recall = \frac{tp}{tp+fn}$, and $Fmeasure = \frac{2 \times Precision \times Recall}{Precision + Recall}$ where tp (true positive) is the number of predicted properties that are actually in Ψ, fp (false positive) is the number of predicted properties which are not in Ψ, and fn (false negative) is the number of predictive properties in Ψ not selected by the model.

We experimented with different sentence embeddings (namely InferSent, Universal Sentence Encoder, and GenSen) and with different thresholds (0 to 1). Due to space constraint, we only present the results in Table 1 corresponding to InferSent and thresholds of 0.1 and 0.9. The proposed approach outperformed the baseline for every threshold. This is expected because the baseline uses Jaccard Index, and thus it relies only on the exactly matching tokens between the property descriptions. Because our approach uses InferSent, it can obtain semantically similar descriptions even though the descriptions themselves do not contain the exact tokens. As we increased the similarity threshold, precision increased, but recall decreased. The threshold of 0.9 was a balance between precision and recall that yielded the F1 scores up to 0.69 (for film and literary work). In fact, F1 scores were above 0.60 for every class except for the country class, which had an F1 score of 0.36 due to overlapping descriptions among propagating and non-propagating properties. The overlapping descriptions for the country class appeared very close in the vector space, which reduced the precision and F1 scores. In addition, the performances were impacted by noisy descriptions, and thus, better preprocessing techniques can potentially improve these scores. In sum, we obtained an overall F1 score of 0.59 and validated our Tobler-inspired hypothesis, i.e., properties can be sorted using their descriptions obtaining the propagating properties at the top. Our result further provides a strong baseline for future research for this novel research problem.

5.3 Qualitative Study

In this section, we describe three different queries that could demonstrate the benefits of our approach by extending their results. To achieve our goal, we used *InferSent* and the threshold value equal to 0.9. All of these queries are simplified

		Precision		Recall		F1	
Class	Threshold	Baseline	Approach	Baseline	Approach	Baseline	Approach
comics character	0.1	0.40	0.39	0.1	1.00	0.16	0.54
	0.9	0.00	0.49	0.00	0.90	0.00	0.61
country	0.1	0.05	0.15	0.01	1.00	0.01	0.23
	0.9	0.00	0.32	0.00	0.81	0.00	0.36
film	0.1	0.15	0.34	0.03	1.00	0.05	0.49
	0.9	0.00	0.62	0.00	0.93	0.00	0.70
literary work	0.1	0.03	0.50	0.01	1.00	0.01	0.65
	0.9	0.00	0.60	0.00	0.90	0.00	0.69
political party	0.1	0.39	0.20	0.51	1.00	0.36	0.31
	0.9	0.00	0.55	0.00	0.87	0.00	0.62
overall	0.1	0.20	0.32	0.13	1.00	0.12	0.44
	0.9	0.00	0.51	0.00	0.88	0.00	0.59

Table 1. Baseline and InferSent results with thresholds of 0.1 and 0.9.

queries tested on Wikidata (for ease of reading). The original queries can be found on the GitHub repository[11].

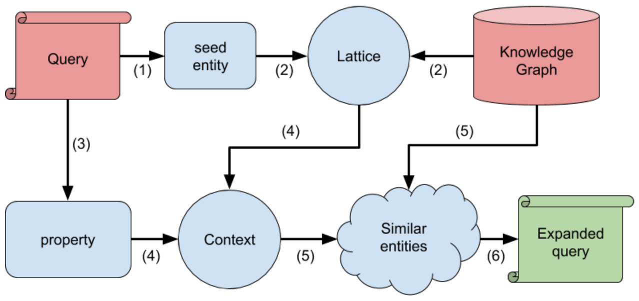

Fig. 3. Qualitative experiment workflow: the elements in red are the inputs and the element in green is the output. To simplify the diagram, we consider only one instantiated entity linked to one instantiated property in the query.

Task description: For each query, the goal is to find an identity context that will allow expanding the query with similar entities according to the user's objective. In this way, users can benefit from more complete results. The workflow is the following (see Figure 3): first, from the query, we extract the instantiated entity (or entities) that will be the seed(s) (step 1). Second, for each seed, we compute its identity lattice (step 2) that will contain in each of its nodes an indiscernible and a propagating set of properties (cf. Algorithms 1 and 3). Third, with the instantiated property (or set of properties) linked to the seed in the query, we select from the lattice, the node having this property in its propagation

[11] https://github.com/PHParis/ConProKnow

set (step 4). This node will be considered as the identity context of the query. Indeed, if multiple identity contexts are possible, the user must choose the best suited for its task purpose. Finally, based on the selected identity context, we can get similar entities (step 5) and rewrite the query with both the seed and similar entities (step 6).

Queries: We tested our approach with three queries. The first query (Listing 1.1) is to retrieve all clinical trials of the drug "Paracetamol". An interesting expansion of this query could be to find all trials of similar legal drugs in terms of medical conditions treated and physical interactions. The second query is to retrieve all persons who once lead France. However, France has a complicated history and has changed its political regime several times (for example, during World War II, or the Napoleonian period). Thus, even if the French territory was almost always the same during the past centuries, each political regime has its own entity in Wikidata. Finally, the third query is to retrieve French politicians from The Republican party that have been convicted. The peculiarity here is that this major political party changed its name several times because of either political scandal or humiliating defeats. We only give details about the first query because of space limitation, the other two are available on the GitHub repository.

```
SELECT DISTINCT ?clinicalTrial WHERE {
  ?clinicalTrial :researchIntervention :Paracetamol .
}
```

Listing 1.1. SPARQL query retrieving all studies about the painkiller named Paracetamol.

	Listing 1.1	See GitHub Repo	See GitHub Repo
Seed	Paracetamol	France	The Republicans
Ψ	research intervention	head of	member of political party
Π	condition treated, interacts with, legal status	capital, official language	country, political ideology
Similar entities	Ibuprofen Aspirin	French 2nd Republic, July Monarchy, ...	UMP, RPR, ...
# of results w/o context	586	12	2
# of results w/ context	860	99 (77)	13

Table 2. Identity context contribution to queries.

Table 2 shows the additional results brought by our approach. Each column corresponds to a query. For the first query (and also for the next ones), there is only one seed "Paracetamol" ("France" and "the Republicans" in the second and the third columns respectively) as it is the only instantiated entity in the query. To fill this table, we first computed the lattice of the seed. Then, we selected a context containing the property "research intervention" in its Ψ since this property is instantiated in the query. Moreover, as explained, our goal is to retrieve trials of similar drugs in terms of the condition treated and legal status. Finally, the query is expanded with similar entities, as shown in Listing 1.2. The results show a 47% increase in the number of clinical trials for the considered context. For the second query, it should be noted that among the 99 results, 22 persons were not head of France. 14 were head of Paris City Council, and 8 were Grand Master of Masonic obedience in France. This is because the council and the obedience are misplaced in the Wikidata ontology. These errors cannot, therefore, be attributed to our approach. The results show a 542% increase in the number of France leaders for the considered context. The results of the third query show a 550% increase in the number of convicted politicians for the considered context.

```
SELECT DISTINCT ?clinicalTrial WHERE {
  VALUES (?drug) { (:Paracetamol) (:Ibuprofen)
     (:Aspirin) }
  ?clinicalTrial :researchIntervention ?drug .
}
```

Listing 1.2. Expanded SPARQL query retrieving all studies about Paracetamol similar entities.

5.4 Discussion

As we have seen, our approach allows for discovering propagating properties for a given indiscernibility set of properties Π. An identity context with its indiscernibility and propagation sets can provide more complete answers to queries through query expansion. The results are very promising but need to be confronted to more different kinds of KGs and to combination of distinct KGs. Also, our approach does not work when the property of an entity lacks property describing it (such as *rdfs:comment* or *schema:description*). Hence, the first step for future work is to circumvent this flaw with a multifaceted approach that can include other information than the descriptions alone. Moreover, sophisticated preprocessing and textual similarity techniques can be incorporated to further improve the results.

6 Conclusion and future work

In this paper, we demonstrated that propagating properties can be discovered semi-automatically. To this end, we presented an approach based on sentence embedding. Given an indiscernible set of properties, the proposed system discovers properties that could be propagated using semantic similarities between the properties. Our approach computes, for an entity, an identity lattice that represents all its possible identity contexts, i.e., both indiscernible and propagating properties. We validated using quantitative and qualitative evaluations that the proposed approach generates promising results for both discovering propagating properties and providing complete answers to the given queries.

Future work includes using other features to improve the results, like values of properties, number of property usage, or semantic features of the property should be tried. However, capturing ontological information of a property when embedding is still an open problem. Secondly, using only sentence embedding, combined with intuition from Tober's first law, might be naïve in some cases. Therefore, there is a need to challenge our work with a combination of distinct KGs. For the time being, we only considered in lattices the case where the entity is subject to a triple, and we should also consider cases where it is the value of a triple. Moreover, using SPARQL queries to help the user to select the best-suited identity context might be an interesting starting point for later work. Finally, to explore SPARQL queries expansion (presented in Section 5.3), a prototype should be implemented to allow users selecting the proper context according to an ordered list of contexts. Also, using RDF* and/or SPARQL* [12] to represent the context as defined in this paper should be investigated.

References

1. Achichi, M., Bellahsene, Z., Todorov, K.: A survey on web data linking. Revue des Sciences et Technologies de l'Information-Série ISI: Ingénierie des Systèmes d'Information (2015)
2. Beek, W., Schlobach, S., van Harmelen, F.: A contextualised semantics for owl: sameas. In: ESWC (2016)
3. Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R.S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y., Strope, B., Kurzweil, R.: Universal sentence encoder. CoRR **abs/1803.11175** (2018)
4. Conneau, A., Kiela, D., Schwenk, H., Barrault, L., Bordes, A.: Supervised learning of universal sentence representations from natural language inference data. In: EMNLP. pp. 670–680. Association for Computational Linguistics (2017)
5. Ding, L., Shinavier, J., Finin, T., McGuinness, D.L.: owl: sameas and linked data: An empirical study (2010)
6. Drummond, N., Shearer, R.: The open world assumption. In: eSI Workshop: The Closed World of Databases meets the Open World of the Semantic Web. vol. 15 (2006)
7. Färber, M., Bartscherer, F., Menne, C., Rettinger, A.: Linked data quality of dbpedia, freebase, opencyc, wikidata, and YAGO. Semantic Web **9**(1), 77–129 (2018). https://doi.org/10.3233/SW-170275, https://doi.org/10.3233/SW-170275
8. Fernández, J.D., Martínez-Prieto, M.A., Gutiérrez, C., Polleres, A., Arias, M.: Binary rdf representation for publication and exchange (hdt). Web Semantics: Science, Services and Agents on the World Wide Web **19**, 22–41 (2013), http://www.websemanticsjournal.org/index.php/ps/article/view/328
9. Ferrara, A., Nikolov, A., Scharffe, F.: Data linking for the semantic web. International Journal on Semantic Web and Information Systems (IJSWIS) **7**(3), 46–76 (2011)
10. Guarino, N., Welty, C.A.: Evaluating ontological decisions with ontoclean. Commun. ACM **45**(2), 61–65 (2002)

11. Halpin, H., Hayes, P.J., McCusker, J.P., McGuinness, D.L., Thompson, H.S.: When owl: sameas isn't the same: An analysis of identity in linked data. In: International Semantic Web Conference. pp. 305–320. Springer (2010)
12. Hartig, O., Thompson, B.: Foundations of an alternative approach to reification in rdf. ArXiv **abs/1406.3399** (2014)
13. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible sroiq. Kr **6**, 57–67 (2006)
14. Idrissou, A.K., Hoekstra, R., van Harmelen, F., Khalili, A., den Besselaar, P.V.: Is my: sameas the same as your: sameas?: Lenticular lenses for context-specific identity. In: K-CAP (2017)
15. Jaccard, P.: Nouvelles recherches sur la distribution florale. Bull. Soc. Vaud. Sci. Nat. **44**, 223–270 (1908)
16. Martínez-Prieto, M.A., Arias, M., Fernández, J.D.: Exchange and consumption of huge rdf data. In: The Semantic Web: Research and Applications. pp. 437–452. Springer (2012)
17. Nentwig, M., Hartung, M., Ngonga Ngomo, A.C., Rahm, E.: A survey of current link discovery frameworks. Semantic Web **8**(3), 419–436 (2017)
18. Noy, N.F., Gao, Y., Jain, A., Narayanan, A., Patterson, A., Taylor, J.: Industry-scale knowledge graphs: lessons and challenges. Commun. ACM **62**(8), 36–43 (2019). https://doi.org/10.1145/3331166, https://doi.org/10.1145/3331166
19. Raad, J., Pernelle, N., Saïs, F.: Detection of contextual identity links in a knowledge base. In: K-CAP (2017)
20. Ristoski, P., Paulheim, H.: Rdf2vec: Rdf graph embeddings for data mining. In: International Semantic Web Conference (2016)
21. Singhal, A.: Modern information retrieval: A brief overview. IEEE Data Eng. Bull. **24**(4), 35–43 (2001)
22. Subramanian, S., Trischler, A., Bengio, Y., Pal, C.J.: Learning general purpose distributed sentence representations via large scale multi-task learning. CoRR **abs/1804.00079** (2018)
23. Tobler, W.R.: A computer movie simulating urban growth in the detroit region. Economic geography **46**(sup1), 234–240 (1970)