Unsupervised Discovery of Corroborative Paths for Fact Validation

Zafar Habeeb Syed[1(✉)], Michael Röder[1,2], and Axel-Cyrille Ngonga Ngomo[1,2]

[1] Data Science Group, Paderborn University, Paderborn, Germany
zsyed@mail.uni-paderborn.de, {michael.roeder,axel.ngonga}@upb.de
[2] Institute for Applied Informatics, Leipzig, Germany

Abstract. Any data publisher can make RDF knowledge graphs available for consumption on the Web. This is a direct consequence of the decentralized publishing paradigm underlying the Data Web, which has led to more than 150 billion facts on more than 3 billion things being published on the Web in more than 10,000 RDF knowledge graphs over the last decade. However, the success of this publishing paradigm also means that the validation of the facts contained in RDF knowledge graphs has become more important than ever before. Several families of fact validation algorithms have been developed over the last years to address several settings of the fact validation problems. In this paper, we consider the following fact validation setting: Given an RDF knowledge graph, compute the likelihood that a given (novel) fact is true. None of the current solutions to this problem exploits RDFS semantics—especially domain, range and class subsumption information. We address this research gap by presenting an *unsupervised approach* dubbed COPAAL, that extracts paths from knowledge graphs to corroborate (novel) input facts. Our approach relies on a mutual information measure that takes the RDFS semantics underlying the knowledge graph into consideration. In particular, we use the information shared by predicates and paths within the knowledge graph to compute the likelihood of a fact being corroborated by the knowledge graph. We evaluate our approach extensively using 17 publicly available datasets. Our results indicate that our approach outperforms the state of the art *unsupervised* approaches significantly by up to 0.15 AUC-ROC. We even outperform *supervised* approaches by up to 0.07 AUC-ROC. The source code of COPAAL is open-source and is available at https://github.com/dice-group/COPAAL.

1 Introduction

The participatory paradigm underlying the Data Web has led to more than 150 billion facts on more than 3 billion things being published on the Web in more than 10,000 RDF knowledge graphs.[1] For example, DBpedia [2], YAGO [20] and WikiData [13] contain information about millions of entities and comprise billions of facts about these entities. These facts are used in the backend of a growing number of applications including in-flight applications [13], community-support

[1] http://lodstats.aksw.org/.

C. Ghidini et al. (Eds.): ISWC 2019, LNCS 11778, pp. 630–646, 2019.
https://doi.org/10.1007/978-3-030-30793-6_36

systems [1] and even personal assistants such as Apple's Siri [13]. Ensuring the veracity of the facts contained in knowledge graphs is hence of critical importance for an increasing number of end users and applications. Manual solutions to the computation of the veracity of facts are clearly an impractical feat due to the volume and the velocity of the data of the Data Web.[2] Consequently, automated solutions to this computation, dubbed *fact validation* [11,17] (also called *fact checking* in some of the literature, e.g., [8]) have been devised over the last years.

The goal of fact validation can be summarized as follows: *Given a fact, compute the likelihood that the given fact is true.* Two main families of approaches have been devised to address this problem (see Sect. 2 for more details). The first family of approaches encompasses solutions which verbalize the input fact and use textual evidence (e.g., large corpora such as the Web or Web crawls) to find statements which support or refute the input fact [8,22,24]. We focus on the second family of approaches. These approaches use a knowledge graph $\mathcal{G}$ as background knowledge and use the facts contained therein to evaluate the likelihood that the given fact is true [5,9,18]. These approaches use sets of facts as evidence to compute the likelihood of a given fact. For example, when using DBpedia version 2016-10 as background knowledge, they might use facts such as (`Barack_Obama`, `birthPlace`, `Hawaii`) and (`Hawaii`, `country`, `United_States_of_America`) to conclude that (`Barack_Obama`, `nationality`, `United_States_of_America`) holds—a fact which is not to be found in the background knowledge base.

Our work is based on the following *observation*: While most approaches which use a knowledge graph as background knowledge have been deployed on RDF knowledge graphs, none has made use of the semantics of the accompanying schema in RDFS to the full. In particular, none of the state-of-the-art approaches makes use of the combination of domain, range and subsumption hierarchy expressed in the schema of most RDF datasets in RDFS. However, the RDFS schema contains crucial information (e.g., type information) necessary to detect facts which can be used to validate or invalidate other facts.

In this paper, we address this research gap by presenting an unsupervised fact validation approach for RDF knowledge graphs which identifies paths that support a given fact (s, p, o). This approach is based on the insight that the predicate p (e.g., `nationality`) carries mutual information with a set of other paths (e.g., paths pertaining to `birthPlace` and `country`) in the background knowledge graph $\mathcal{G}$. Hence, the presence of certain sets of paths in $\mathcal{G}$ that begin in s and end in o can be regarded as evidence which *corroborates the veracity* of (s, p, o). Our approach is the first to take the *domain* and *range* information of p, the type of s and o as well as the subsumption relations between types in the RDFS schema of $\mathcal{G}$ into consideration while identifying these paths. Our results show conclusively that using this information leads to significantly higher AUC-ROC results on 17 benchmark datasets.

Our approach has several advantages over the state of the art: (i) It uses data which can be directly queried via SPARQL from $\mathcal{G}$, i.e., there is no *need to*

[2] See https://lod-cloud.net/ for data on the growth of the Linked Open Data Cloud.

alter the representation mechanism of $\mathcal{G}$ or to use an internal representation of $\mathcal{G}$ in our implementation. Moreover, our approach can exploit the large body of work on scaling up triple stores to competitive runtimes. (ii) The proposed co-occurrence measure for the similarity calculation between predicates and paths is *not bound to path lengths and can hence be exploited to detect paths of any finite length.* (iii) Our approach is *completely unsupervised* and neither training nor labeled data is required.

The rest of the paper is organized as follows: Sect. 2 present details pertaining to related fact validation approaches. In Sect. 3, we present a brief overview of the formal notation used in this paper. We also introduce the formal specification we use throughout this work. Section 4 details the formal model underlying our approach. In particular, it gives a formal specification of corroborative paths and how they can be used to measure the likelihood of a fact being true. Section 5 provides the details of our implementation. We present our experimental setup in Sect. 6 and discuss our results in Sect. 7. Finally, we conclude in Sect. 8.

2 Related Work

Approaches to fact validation can be broadly classified into two categories: (i) approaches that use unstructured *textual* sources [8,22,24] and (ii) approaches that use *structured* information sources [3,17–19]. The latter—in particular approaches that use a given knowledge graph for fact validation—are more relevant to the work presented herein. Several approaches view a given knowledge graph as labeled graph connecting nodes (entities) and edges (relations). Given an input triple (s, p, o), the goal is then to search for paths of length up to a given threshold k and use them to validate/invalidate the given input triple. For instance, in [5,18] a knowledge graph is viewed as undirected network of paths. The task is then to find shortest paths that connect s and o and are semantically related to p. These approaches are *unsupervised* and do not require prior training data. However, these approaches do not take into consideration the *terminological information* (in particular the semantics of RDFS) of the input knowledge graph while defining semantic proximity metrics. Other approaches view KBs as graphs and search for *metapaths* to extract features [9,21,25]. These features are then used to train a classification model to label unseen facts as true or false. However, these approaches require training data in the form of labeled *metapaths* and hence required significantly more human effort that the approach presented herein. In PredPath [17], the authors propose a novel method to automatically extract metapaths—called *anchored predicate* paths—given a set of labeled examples. To achieves this goal, PredPath uses the `rdf:type` information contained in the input knowledge graph. However, the *anchored predicate paths* used for learning features are selected based on the type information of *subject* and *object* irrespective of the *predicate* connecting them. This means that they do not consider the domain, range and class subsumption provided by the RDFS schema of the given knowledge graph. Consequently, their ability to generalize over paths is limited as shown in Sect. 7 of this paper. Additionally, PredPath requires labeled training data. Hence, porting it to previously unseen

predicates is significantly more demanding that porting our approach, which is fully unsupervised.

Alternative to graph models, several approaches encode the entities and relations in a KB using vector embeddings [3,12,19,23]. The fact validation problem is then formulated as calculating the similarity between the entities and predicate of a given input triple. Embedding-based methods for link prediction address a related but different problem. Given a KG $\mathcal{G}$, they compute a score function, which expresses how likely it is that any triple whose subject, predicate and object belong to the input graph $\mathcal{G}$ should belong to $\mathcal{G}$ [14]. Fact validation approaches addresses a different but related goal: Given a graph $\mathcal{G}$ and a triple t, they aim to compute the likelihood that t is true [8,18,22]. A core repercussion of these two different problem formulations are the runtimes and the applications of link prediction and fact checking. While fact validation algorithms are used in online scenarios embedding-based algorithms are often used offline. Approaches such as [6,7] mine Horn rules that can be used for knowledge base completion tasks. However, they often fail to scale to large knowledge graphs.

Our approach, is inspired by approaches that discover metapaths. We propose a novel approach for finding paths which corroborate a given triple (s, p, o). In addition, we present a novel measure to calculate association strength these paths and the input triple. In contrast to approaches based on metapaths, our approach does not need training examples and does not require any supplementary effort to deployed to previously unseen relations.

3 Preliminaries

Throughout this paper, we consider RDF knowledge graphs with RDFS semantics. We use the notation presented in Table 1.

Table 1. List of symbols

Notation	Description
$\mathcal{G}$	A knowledge graph
$\mathbb{B}, \mathbb{C}, \mathbb{E}, \mathbb{L}, \mathbb{P}$	Set of all blank nodes, RDFS classes, RDF resources, Literals and RDF predicates, respectively
$\pi^k(v_0, v_k)$	*Directed path* of length k between nodes v_0 and v_k in $\mathcal{G}$
$\mu^k(v_0, v_k)$	*Undirected path* of length k between nodes v_0 and v_k in $\mathcal{G}$
$\Pi^k(p)$	Set of *corroborative paths* for a predicate p
$\Pi^k_{(t_x,t_y)}$	Set of *typed directed paths* of length k between nodes v_0 and v_k in $\mathcal{G}$
$M^k_{(t_x,t_y)}$	Set of *typed undirected paths* of length k between nodes v_0 and v_k in $\mathcal{G}$
$\vec{q}$	Vector of k predicates in $\mathcal{G}$
$\Pi^k_{(t_x,t_y),\vec{q}}$	Set of $\vec{q}$-restricted typed *directed paths* of length k between nodes v_0 and v_k in $\mathcal{G}$
$M^k_{(t_x,t_y),\vec{q}}$	Set of $\vec{q}$-restricted typed *undirected paths* of length k between nodes v_0 and v_k in $\mathcal{G}$
$\gamma(x)$	Function mapping each element of $\mathbb{E} \cup \mathbb{P} \cup \mathbb{B} \cup \mathbb{L}$ to its type
$\lambda(t_x)$	Function mapping the type t_x to a set of resources that are instances of this type
$D(p)$	The domain of the predicate p
$R(p)$	The range of the predicate p

3.1 Knowledge Graph

Definition 1. *An RDF knowledge graph $\mathcal{G}$ is a set of RDF triples, i.e.,*

$$\mathcal{G} = \{(s, p, o) | s \in \mathbb{E} \cup \mathbb{B}, p \in \mathbb{P}, o \in \mathbb{E} \cup \mathbb{B} \cup \mathbb{L}\}, \tag{1}$$

where $\mathbb{E}$ is the set of all RDF resources, $\mathbb{B}$ is the set of all blank nodes, $\mathbb{P} \subseteq \mathbb{E}$ is the set of all RDF predicates and $\mathbb{L}$ represents the set of all literals.

Intuitively, an RDF knowledge graph can be understood as an edge-labeled directed graph in which the node s is connected to the node o via an edge with the label p iff the triple $(s, p, o) \in \mathcal{G}$. This is the approach we use to display knowledge graphs graphically (see, e.g., Fig. 1). We use the notation $s \xrightarrow{p} o$ to denote that $(s, p, o) \in \mathcal{G}$. We denote the set of all RDFS classes as $\mathbb{C}$ (with $\mathbb{C} \subseteq \mathbb{E}$). For $A \in \mathbb{C}$ and $B \in \mathbb{C}$, we write $A \sqsubseteq B$ to signify that $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ for any interpretation $\cdot^{\mathcal{I}}$.

Example 1. An excerpt of an example RDF knowledge graph—which we will use as a running example—is displayed in Fig. 1. The example shows a subgraph extracted from DBpedia[3] consisting of nodes (resources) (e.g., `Barack Obama` and `United States`) and edges (relations) connecting these entities either directly or via intermediate nodes (e.g., `birthplace`).

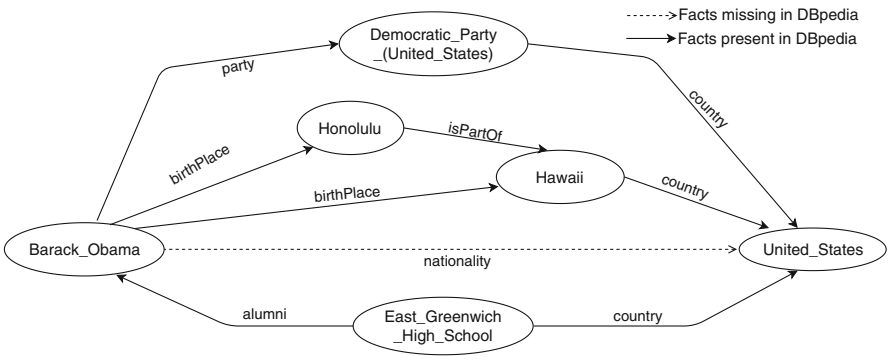

Fig. 1. A subgraph of DBpedia version 10-2016.

Definition 2. *Path: A path of length k in a knowledge graph $\mathcal{G}$ is a cycle-free sequence of triples from $\mathcal{G}$ of the form $(v_0, p_1, v_1), (v_1, p_2, v_2), ..., (v_{k-1}, p_k, v_k)$.*

This means in particular that $\forall i, j \in [0, k], i \neq j \rightarrow v_i \neq v_j$. We use $\pi^k(v_0, v_k)$ to denote paths between v_0 and v_k. For the sake of legibility, we use the notation $v_0 \xrightarrow{p_1} \ldots \xrightarrow{p_{k-1}} v_k$ to denote paths. Note that several paths can exist between v_0 and v_k. For example, `BarackObama` $\xrightarrow{\texttt{birthPlace}}$ `Hawaii` $\xrightarrow{\texttt{country}}$ `USA` and `BarackObama` $\xrightarrow{\texttt{party}}$ `DemocraticParty` $\xrightarrow{\texttt{country}}$ `USA` are both paths of length 2 between the resources `BarackObama` and `USA` in our running example.

[3] http://downloads.dbpedia.org/2016-10/.

Definition 3. *Undirected path: An undirected path of length k in a graph $\mathcal{G}$ is a cycle-free sequence of triples of the form $(v_0, p_1, v1), (v_1, p_2, v_2), ..., (v_{k-1}, p_k, v_k)$ where $\forall i \in [0, k-1]$ $(v_i, p_{i+1}, v_{i+1}) \in \mathcal{G} \vee (v_{i+1}, p_{i+1}, v_i) \in \mathcal{G}$.*

Again, this means that $\forall i, j \in [0, k], i \neq j \rightarrow v_i \neq v_j$. We denote undirected paths with $\mu^k(v_0, v_k)$. For example, `BarackObama` $\xleftarrow{\texttt{alumni}}$ `GreenwichHighSchool` $\xrightarrow{\texttt{country}}$ `USA` is an undirected path of length 2 between `BarackObama` and `USA` in our example.

4 Corroborative Paths

4.1 Intuition

In this paper, we address the *following problem*: Given an RDF knowledge graph $\mathcal{G}$ and a triple (s, p, o), compute the likelihood that (s, p, o) is true. For example, we would have good reasons to believe that `BarackObama` is a citizen of the `USA` given that `BarackObama` was born in `Hawaii` and `Hawaii` is located in the `USA`. Clearly, we cannot formally infer that x is a national of z by virtue of the existence of $x \xrightarrow{\texttt{birthplace}} y \xrightarrow{\texttt{country}} z$. Still, this path is a strong indicator (i.e., strongly corroborates) triples of the form $x \xrightarrow{\texttt{nationality}} z$. The *basic intuition behind our work* is correspondingly that the existence of certain paths $\pi^k(s, o)$ between s and o is a strong indicator for the correctness (i.e., corroborate the existence) of (s, p, o) and can hence be used to compute its likelihood.

4.2 Formal Model

Let γ be a function which maps each element of $\mathbb{E} \cup \mathbb{P} \cup \mathbb{B} \cup \mathbb{L}$ to its type. For example, $\gamma(\texttt{BarackObama}) = \texttt{Person} \sqcap \texttt{Agent} \sqcap \texttt{Politician} \sqcap \texttt{President}$ and $\gamma(\texttt{UnitedStates}) = \texttt{Place} \sqcap \texttt{Location} \sqcap \texttt{Country} \sqcap \texttt{PopulatedPlace}$ in our running example.[4] Further, let λ be a function which maps a given type t_x to a set of resources that are instances of t_x by virtue of RDFS semantics. Extending the formal model in [17], we now define the set $\Pi^k_{(t_x, t_y)}$ of typed paths of length k between pairs of resources of type t_x and t_y in a knowledge graph $\mathcal{G}$ as follows:

$$\Pi^k_{(t_x, t_y)} = \{\pi^k(v_0, v_k) \mid \gamma(v_0) \sqsubseteq t_x \wedge \gamma(v_k) \sqsubseteq t_y\}. \tag{2}$$

For $t_x = \{\texttt{Person}\}$ and $t_y = \{\texttt{Place}\}$, the path `BarackObama` $\xrightarrow{\texttt{birthPlace}}$ `Hawaii` $\xrightarrow{\texttt{country}}$ `USA` is an element of the set $\Pi^2_{(t_x, t_y)}$ in our running example. We define the set $M^k_{(t_x, t_y)}$ of typed undirected paths analogously.

[4] We use $\sqcap$ to denote the conjunction of classes. Note that given that `President` $\sqsubseteq$ `Person` $\sqsubseteq$ `Agent`, we could write the type `BarackObama` in an abbreviated form. Similar considerations holds for the type of `UnitedStates`. We chose to write the types out to remain consistent with the output of our example knowledge graph, DBpedia 2016-10.

Let $\vec{q} = q_1, \ldots, q_k$ be a vector of properties of length k. We define the set of $\vec{q}$-restricted typed paths $\Pi^k_{(t_x,t_y),\vec{q}} \subseteq \Pi^k_{(t_x,t_y)}$ as follows:

$$\Pi^k_{(t_x,t_y),\vec{q}} = \big\{\pi^k(v_0,v_k) \mid \pi^k(v_0,v_k) \in \Pi^k_{(t_x,t_y)}, \\ \forall i \in [0,k-1] : (v_i, p_{i+1}, v_{i+1}) \in \pi^k(v_0,v_k) \rightarrow p_{i+1} = q_{i+1}\big\}. \tag{3}$$

Put simply, this is the set of typed paths such that the sequence of properties in each path is exactly $\vec{q}$. For example, let t_x={`Person`}, t_y = {`Place`} and $\vec{q}$ = (`birthPlace`, `country`). Then the path `BarackObama` $\xrightarrow{\texttt{birthPlace}}$ `Hawaii` $\xrightarrow{\texttt{country}}$ `USA` is the only element of $\Pi^2_{(t_x,t_y),\vec{q}}$ in our running example. We call the elements of $\Pi^k_{(t_x,t_y),\vec{q}}$ *similar* as they share a sequence of predicates (i.e., $\vec{q}$). We define sets of $\vec{q}$-restricted undirected typed paths $M^k_{(t_x,t_y),\vec{q}}$ analogously to $\Pi^k_{(t_x,t_y),\vec{q}}$.

We can now use restricted typed paths to compute how well a predicate is corroborated in a knowledge graphs as follows: Let $D(p)$ be the domain of p and $R(p)$ be its range. Given that we assume RDF knowledge graphs, we can safely assume the existence of an RDFS class hierarchy for the said graph (defined via the `rdf:type` predicate). Consequently, we can derive the following important condition on paths $\pi^k(s,o)$ which are to corroborate the correctness of (s,p,o): *Only typed paths in* $\Pi^k_{(D(p),R(p))}$ *can corroborate facts with the predicate* p. This particular insight is one of the major differences between this and previous works (see Sect. 2), in which the consequences of RDFS semantics were not taken into consideration. In particular, while previous approaches [17] used at most $\gamma(s)$ and $\gamma(o)$ to measure the strength of the association between paths and predicates, we use $D(p)$ and $R(p)$ as well as the RDFS class hierarchy in the input knowledge graph $\mathcal{G}$ to determine the degree to which a path $\pi^k(s,o)$ corroborates a predicate p.

Given an RDF knowledge graph $\mathcal{G}$, we hence define the *corroborative paths* for a predicate p formally as follows:

$$\Pi^k(p) = \bigcup_{j=1}^{k} \Pi^j_{(D(p),R(p))}. \tag{4}$$

Simply put, *corroborative paths* in $\Pi^k(p)$ are paths of length at most k that carry similar information to p.

4.3 Association Strength

We base our computation of the strength of the association between $\Pi^j_{(t_x,t_y),\vec{q}}$ and p on their normalized pointwise mutual information [4]. To this end, we define probability $\mathcal{P}(\Pi^j_{(t_x,t_y),\vec{q}})$ of pairs of instances of t_x resp. t_y being connected via a $\vec{q}$-restricted path of length j is as follows:

$$\frac{\left|\{(a,b) : \gamma(a) \sqsubseteq t_x \wedge \gamma(b) \sqsubseteq t_y \wedge (\exists \pi^j(a,b) \in \Pi^j_{(t_x,t_y),\vec{q}})\}\right|}{|\lambda(t_x)| \cdot |\lambda(t_y)|}. \tag{5}$$

The probability $\mathcal{P}(p)$ of the predicate p linking resources of type t_x and t_y is

$$\frac{|\{(a,p,b) : \gamma(x) \sqsubseteq t_x \wedge \gamma(y) \sqsubseteq t_y \wedge (a,p,b) \in \mathcal{G}\}|}{|\lambda(t_x)| \cdot |\lambda(t_y)|} \tag{6}$$

Finally, the joint probability $\mathcal{P}(\Pi^j_{(t_x,t_y),\vec{q}}, p)$ is defined as

$$\frac{\left|\{(a,b) : \gamma(a) \sqsubseteq t_x \wedge \gamma(b) \sqsubseteq t_y \wedge (\exists \pi^j(a,b) \in \Pi^j_{(t_x,t_y),\vec{q}}) \wedge (a,p,b) \in \mathcal{G}\}\right|}{|\lambda(t_x)| \cdot |\lambda(t_y)|}. \tag{7}$$

We could now compute the NPMI of $\Pi^j_{(t_x,t_y),\vec{q}}$ and p as defined in [4]. However, a direct implementation of the original definition of the NPMI would be expensive as it would require deduplicating the sets of pairs (a,b) connected by the paths in $\Pi^j_{(t_x,t_y)}$.[5] Hence, our approach implements an approximation of the NPMI based on counting the number of paths which connect pairs (a,b) instead of the pairs themselves. We hence end up with the following approximations (note that these values are not probabilities):

$$\widehat{\mathcal{P}}(\Pi^j_{(t_x,t_y),\vec{q}}) = \frac{|\Pi^j_{(t_x,t_y),\vec{q}}|}{|\lambda(t_x)| \cdot |\lambda(t_y)|} \tag{8}$$

$$\widehat{\mathcal{P}}(\Pi^j_{(t_x,t_y),\vec{q}}, p) = \frac{|\{\pi^j(a,b) \in \Pi^j_{(t_x,t_y),\vec{q}} : (a,p,b) \in \mathcal{G}\}|}{|\lambda(t_x)| \cdot |\lambda(t_y)|}. \tag{9}$$

These approximations can be computed by using SPARQL queries without `DISTINCT` clause, which makes the computation an order of magnitude faster (see Table 7 for some of the scores returned by this function). Note that $\mathcal{P}(p)$ remains unchanged and the number of paths $a \xrightarrow{p} b$ is exactly equal to the number of pairs (a,b) connected by p. Based on these approximations we can now approximate the NPMI of $\Pi^j_{(t_x,t_y),\vec{q}}$ and p as follows:

$$\widehat{\text{NPMI}}(\Pi^j_{(t_x,t_y),\vec{q}}, p) = \frac{\log\left(\frac{\widehat{\mathcal{P}}\left(\Pi^j_{(t_x,t_y),\vec{q}}, p\right)}{\widehat{\mathcal{P}}\left(\Pi^j_{(t_x,t_y)}\right) \cdot \mathcal{P}(p)}\right)}{-\log\left(\widehat{\mathcal{P}}\left(\Pi^j_{(t_x,t_y),\vec{q}}, p\right)\right)} \tag{10}$$

5 Method and Implementation

This section presents our implementation of the formal model presented above in detail. In particular, we show how some of the core computations of our model can be implemented using SPARQL queries, ensuring practicable runtimes for our approach. As above, we explain the approach using directed paths for the sake of legibility. The approach was also implemented using undirected paths. An evaluation of the performance of the approach with directed and undirected paths is presented in Sect. 7.

[5] Preliminary experiments suggest a 20-fold increase in runtime without any significant increase in AUC-ROC.

5.1 Algorithm

Given an input triple $t = (s, p, o)$, a knowledge graph $\mathcal{G}$ and a maximum path length k, our implementation begins by identifying a set of *paths* of varying lengths connecting s and o, respectively. For each path, it calculates a score, which explicates the degree to which the path corroborate t. Finally, the scores are amalgamated to a single score τ which expresses the veracity of t. The complete algorithm is shown in Algorithm 1 and can be separated into the 4 steps (i) Initialization, (ii) Path discovery, (iii) Path scoring and (iv) Veracity calculation.

Algorithm 1. COPAAL - Corroborative Fact Validation

Input : The input triple $t = (s, p, o)$, the knowledge graph $\mathcal{G}$ and the maximum path length k
Output: A veracity score τ for t

```
// Initialization
prune(G)
c_{D(p)} ← countInstances(D(p))
c_{R(p)} ← countInstances(R(p))
c_p ← countTriples(p)
List Z ←{}; List Q ←{}
// Path Discovery
for j = 1 to k do
    QT ← generateQueryTemplates(j)
    for qt ∈ QT do
        sq ← qt(v_o = s, v_k = o)
        Q ← execute(sq)
        prune(Q)
        for q⃗ ∈ Q do
            Q.add((qt, q⃗))
        end
    end
end
// Path scoring
for (qt, q⃗) ∈ Q do
    sq ← generatePathCountQuery (qt, π^j(s, o), D(p), R(p))
    c_Π ← execute(sq)
    sq ← generateCoocCountQuery (sq, t)
    c_{Π,p} ← execute(sq)
    Z.add(calcNPMI(c_{Π,p}, c_Π, c_p, c_{D(p)}, c_{R(p)}))
end
// Veracity calculation
τ ← 1
for ζ ∈ Z do
    τ ← τ × (1 − ζ)
end
return 1 − τ
```

Initialization. Firstly, we prune $\mathcal{G}$ by removing all nodes from domain outside the union of (i) base namespace(s) of $\mathcal{G}$, (ii) the namespace for RDF, RDFS and OWL. We carry out this preprocessing because we are interested in relations (edges) that are defined by the ontology of the given $\mathcal{G}$ (line 1). Thereafter, the domain $D(p)$ and range $R(p)$ of the given triple's predicate p are determined.[6] The number of instances of these two types as well as the number of triples containing p as predicate are retrieved via SPARQL count queries (lines 2–4).

Path Discovery. In the second step, the properties of all paths $\pi^j(s, o)$ (i.e., their $\vec{q}$ restrictions) of length $j \in [1, k]$ between s and o are retrieved. To this end, we generate SPARQL[7] query templates (line 7). The query template which retrieves directed paths of length $j = 2$ between `?v0` and `?v2` from an RDF graph is:

```
SELECT ?p1 ?p2
WHERE {
    ?v0 ?p1 ?v1 .
    ?v1 ?p2 ?v2 .
}
```

Note that the query has to be modified with `UNION` to cover *undirected paths*. Still, a single query can be used to detect paths of any length. Hence, our approach generates k queries in this step.

After generating all necessary query templates up to the given length k, we replace the variables `?v0` and `?vj` with s and o respectively in the query (line 9). We prune the results of the query (line 11) by removing results containing predicates which define the terminology (e.g., class membership through `rdf:type`, class hierarchy through `rdfs:subClassOf`).[8] The remaining $\vec{q}$-restrictions are stored as pairs together with the template which was used to retrieve them in the list $\mathcal{Q}$ (line 12–14). We store the template to ensure that we can reconstruct the direction of the predicates in case undirected paths are used.

Path Scoring. Pairs in $\mathcal{Q}$ are used to define the $\vec{q}$-restricted typed path sets $\Pi^j_{(D(p),R(p)),\vec{q}}$. For each of these pairs, a score ζ is calculated based on the NPMI approximation in Eq. 10 (lines 16–23). For the sake of efficiency, we use SPARQL queries to obtain the necessary counts of typed paths which are generated based on the query template and $\vec{q}$ (line 18). However, as pointed out in [7], a direct translation of the needed counts into queries leads to time-consuming computa-

[6] If $D(p)$ or $R(p)$ are not available, the types of the given subject or object will be used, respectively.

[7] https://www.w3.org/TR/rdf-sparql-query/.

[8] We are aware that the terminology (especially concept similarity scores) used in `G` can potentially inform the fact validation process further. Studying the integration of assertional and terminological information will be the object of future work and is out of the scope of this paper.

tions[9] which require optimization. Therefore, we generate the SPARQL queries needed for our counts with a recursive structure. Listing 1.1 shows a sample query used to count the number of paths `?vo` $\xrightarrow{birthPlace}$ `?v1` $\xrightarrow{country}$ `?v2` between entities with the types `Person` and `Country`, respectively.

Listing 1.1. SPARQL query to count all paths of an example $\vec{q}$

```
SELECT SUM(?b1*?b2) as ?sum WHERE {
  SELECT COUNT(?v1) as ?b2, ?b1 WHERE {
    ?v0 <http://dbpedia.org/ontology/birthPlace> ?v1 .
    ?v0 a <http://dbpedia.org/ontology/Person> .
    {
      SELECT COUNT(?v2) as ?b1, ?v1 WHERE {
        ?v1 <http://dbpedia.org/ontology/country> ?v2 .
        ?v2 a <http://dbpedia.org/ontology/Country> .
      } GROUP BY ?v1
    }
  } GROUP BY ?v0 ?b1
}
```

Veracity Calculation. We treat the association strength of each $\vec{q}$-restricted typed path as the confidence with which the path supports the existence of the input predicate p. We hence combine the ζ values by checking whether at least one path supports p. Let Z be the set of scores of all single paths, the veracity score τ can be calculated with the following equation (see lines 23–28):

$$q\tau = 1 - \prod_{\zeta \in Z} (1 - \zeta)\,. \tag{11}$$

6 Experiments and Results

In this section, we provide details of the data and hardware we used in our experiments. We compare the results of our approach with those achieve by state-of-the-art approaches in the subsequent section.

6.1 Setup

Knowledge Graph. For our experiments, we chose DBpedia version 2016-10 as background knowledge. We chose this dataset because it is the reference dataset of a large number of fact validation benchmarks. We used the latest dumps[10] of `ontology`, `instance types`, `mapping-based objects` and `infobox properties`. We filtered out triples that (i) contain literals and datatypes or

[9] We used Virtuoso and Fuseki for our experiments and our runtime findings support [7].

[10] http://downloads.dbpedia.org/2016-10/.

(ii) link the entities in DBpedia to external sources. The final graph contains 44 million triples, which we stored using an instance of Openlink Virtuoso v7.2.5.1 hosted on VM with 16GB memory and 256GB disk space. To ensure the comparability of our results, we ran our evaluation using GERBIL [16]—a benchmarking platform that facilitates the evaluation of fact validation systems across different datasets.[11] We used the AUC-ROC as an evaluation metric and set $k = 2$ for the sake of comparability with previous works.

Competing Approaches. We compare our approach (COPAAL) to three state-of-the-art graph-based fact validation approaches: (i) Knowledge Stream (KS), (ii) its variant Relational Knowledge Linker (KL-REL) [18] and (iii) Discriminative Path Mining (PredPath) [17]. For all these approaches, we use the implementation provided by the authors [18].[12] We considered the configuration suggested in the original paper: (i) PredPath [17] uses the top-100 features while learning positive and negative facts. (ii) KS [18] and KL-REL [18] use the top-5 paths and single best path, respectively, for validating input triples.

6.2 Benchmarks

We evaluated all the approaches using two publicly available sets of benchmarks: (i) the *Real-World* and (ii) *Synthetic* datasets[13] made available by the authors of the literature [18]. In addition, we generated a new set of benchmarks dubbed *FactBench-DBpedia* from the *FactBench*[14] dataset. All the facts in *FactBench* are automatically extracted from DBpedia and Freebase for 10 different relations[15] and stored in the form of RDF models. In *FactBench*, the positive facts are generated by querying DBpedia and Freebase and selecting top 150 results returned for each relation. The negative facts are generated by modifying the positive facts while still following domain and range restrictions. The positive and negative facts are collected into 6 different benchmarks dubbed *Domain, Range, Domain-Range, Mix, Random, Property*. *FactBench-DBpedia* restricts the generation process of *FactBench* to DBpedia by extracting all facts belonging to DBpedia and facts from Freebase whose resources can be mapped to resources in DBpedia. Table 2 shows the stats for the different datasets.

7 Results

7.1 Comparison of Directed and Undirected Paths

We first aimed to determine the type of paths for which our approach performs best. We hence compared the AUC achieved by both variations of our approach

[11] All the datasets and result files can be found at https://hobbitdata.informatik.uni-leipzig.de/COPAAL/.

[12] https://github.com/shiralkarprashant/knowledgestream.

[13] https://github.com/shiralkarprashant/knowledgestream/tree/master/datasets.

[14] https://github.com/DeFacto/FactBench.

[15] *award*, *birthPlace*, *deathPlace*, *foundationPlace*, *leader*, *team*, *author*, *spouse*, *starring*, *subsidiary*.

Table 2. Summary of benchmark datasets

Dataset	*FactBench-DBpedia*						*Real-World*				*Synthetic*						
Subset	Domain	Domain-Range	Range	Mix	Property	Random	Birth-Place	Death-Place	Education	Nationality	US-CAP	NBA-Team	Oscars	CEO	US-WAR	US-VP	FLOTUS
Positive	1,124	1,124	1,124	1,124	1,124	1,124	273	126	466	50	50	41	78	201	126	47	16
Negative	1,119	1,006	1,123	1,014	1,153	511	819	378	1,395	150	250	123	4,602	1,007	584	227	240
Total	2,243	2,130	2,247	2,138	2,277	1,635	1,092	504	1,861	200	300	164	4,680	1,208	710	274	256

Table 3. Comparison of AUC-ROC achieved using directed and undirected paths

	Domain	Domain-Range	Range	Mix	Random	Property
Undirected paths	**0.9348**	**0.9389**	**0.8937**	**0.8561**	**0.9411**	**0.7307**
Directed paths	0.7741	0.7824	0.7416	0.5914	0.6411	0.4713

on FactBench-DBpedia (see Table 3). The results are clear: Using undirected paths (average AUC-ROC = 0.87) always outperforms using directed paths (avg. AUC-ROC = 0.66) and are 0.21 better on average w.r.t. the AUC-ROC they achieve. We studied the results achieved using the two types of paths. It became quickly evident that using undirected paths allows to detect significantly more corroborative evidence. Therewith, undirected paths achieve a better approximation of the probability of a triple being true (see Table 7 for examples). Consequently, we only consider our approach with undirected paths in the following.

7.2 Comparison with Other Approaches

Tables 4 and 5 show the AUC-ROC results of all the approaches on the benchmarks contained in the *Real-World* and *Synthetic* datasets, respectively. Our approach outperforms other approaches on most of these datasets. In the best case, we are roughly 4.5% (absolute value, Birth Place benchmark) better than PredPath and more than 20% (absolute value, Birth Place benchmark) better than KS on real data. A careful study of our results reveals that the *anchored predicate paths* used by PredPath for learning features are restricted by the types of subject and object irrespective of predicate of the input triple. Hence they

Table 4. AUC-ROC results of all approaches on *Real-World* datasets

	Birth place	Death place	Education	Nationality
COPAAL	**0.9441**	0.8997	**0.8731**	**0.9831**
PredPath	0.8997	0.8054	0.8644	0.9520
KL-REL	0.9254	**0.9095**	0.8547	0.9692
KS	0.7197	0.8002	0.8651	0.9789

Table 5. ROC-AUC results of all approaches on *Synthetic* datasets

	US-CAP	NBA-Team	Oscars	CEO	US-WAR	US-VP	FLOTUS
COPAAL	**1.000**	**0.999**	0.995	**0.912**	**0.999**	**0.953**	**1.000**
PredPath	0.996	0.923	**0.999**	0.897	0.995	0.944	**1.000**
KL-REL	**1.000**	0.999	0.976	0.898	0.873	0.891	0.983
KS	**1.000**	0.999	0.950	0.811	0.865	0.798	0.980

someetimes fail to generalize well. On the other hand, KL-REL uses single best paths, which sometimes limits its ability to validate facts if it is not able to rank the path which conveys the most evidence for the input triple to the first position. This is made evident by the examples shown in Table 7: We computed the union of the top-3 paths identified by our approach and all other approaches on the three datasets for which the difference in AUC values were the largest. We also computed the weights assigned by each of the approaches (i.e., $\widetilde{\text{NPMI}}$ for our approach, average flow values of paths for KS and KL-REL [18] and weights learned by the classifier for PredPath [17]). While our approach finds all paths and allocated them weights, the other approach sometimes fail to detect relevant paths (marked by dashes in Table 7) and are hence not able to use them in their evidence computation. Having a large number of paths available however also means that our scores are (even if rarely) overoptimistic w.r.t. evidence for a triple, which explain the marginally lower scores we achieve on Death Place and Oscars.

The results on FactBench-DBpedia (see Table 6) confirm the insight we gained on the previous two datasets. Our approach outperforms the state of the art and achieve a better AUC-ROC on most datasets. We ran a Wilcoxon signed ranked test (significance = 99%) on all results we collected. The results state that our approach is significantly better than the state of the art.

One could assume that our approach is slower than the state of the art due to the larger amount of evidence it collects. Hence, we measured the average throughput of all the approaches including all phases of the processing. The average throughput of our approach was 21.02 triples/min. KS, which follows an approach similar to ours, achieves an average throughput of 10.05 triples/min while its counterpart KL-REL achieves 29.78 triples/min. PredPath's average throughput was 21.67 triples/min. Overall, our results show that our approach scales as well as the state of the art while achieving significantly better results.

Table 6. ROC-AUC results of all approaches on *FactBench-DBpedia* datasets

	Domain	DomainRange	Mix	Property	Random	Range
COPAAL	**0.9348**	0.9389	**0.8561**	**0.7307**	**0.9411**	0.8937
PredPath	0.9301	**0.9447**	0.8408	0.7154	0.9354	**0.8992**
KL-REL	0.8453	0.8619	0.7721	0.6154	0.8547	0.8219
KS	0.8019	0.8124	0.7215	0.6047	0.7911	0.8047

Table 7. Union of the top-3 paths identified by the different approaches and their weighting. The weights allocated by each of the approaches are given in the corresponding column. A dash (-) means that the approach was not able to find the said path.

Dataset	Path	COPAAL	KS/KL-REL	PredPath
BirthPlace	hometown	0.65	0.28	–
	birthPlace isPartOf	0.65	0.23	26
	highSchool city	0.62	0.21	–
	parent birthPlace	0.63	0.08	29
	child birthPlace	0.60	0.04	21
CEO	foundedBy	0.72	0.28	3
	owningCompany	0.70	–	–
	owner	0.70	–	–
	parentCompany keyPerson	0.70	0.08	7
	employer	0.64	0.23	9
US-VP	successor	0.62	0.19	5
	predecessor	0.61	0.12	7
	vicePresident president	0.55	–	–
	associate president	0.49	–	2
	predecessor successor	0.48	0.02	13

8 Conclusion and Future Work

In this paper, we present a novel unsupervised approach for the validation of facts using an RDF knowledge graph $\mathcal{G}$ as background knowledge. Our approach uses domain, range and class subsumption information found in the schema of $\mathcal{G}$ to outperform both supervised and unsupervised fact validation approaches. We evaluated our results on 17 datasets against three state-of-the-art approaches. Our results show that our approach outperforms the state of the art significantly (Wilcoxon signed ranked test, $p < 0.01$). We studied the difference between the approaches and concluded that our approach performs better because it is able to score corroborative paths more accurately as it uses more information from the schema of $\mathcal{G}$. These results point to the importance of using the semantics of the data contained in RDF knowledge graphs when aiming to validate them. Another advantage of our approach is that it allows to verbalize the evidence found to support a given input triple.

The main limitation of our approach lies in its relying on the existence of type information. Well-defined ontologies are not always given in real world datasets and therefore our approach cannot be applied on them. Previous works have aimed at improving type information in noisy knowledge graphs [15]. We will evaluate whether combining our approach with such algorithms leads to bet-

ter corroborative paths in future works. Additionally, the approaches evaluated herein are limited to evidence found in one RDF graph. In future work, we will consider performing fact validation at a larger scale. In particular, we will use the linked nature of Linked Data sets to detect paths across several knowledge graphs. We will focus on the scalability and the distributed execution of this novel solution. Moreover, we will consider relaxing the requirements to types used in the definition of $\Pi^k_{(t_x,t_y),\vec{q}}$ by using well-defined semantic similarities [10].

Acknowledgements. This work has been supported by the BMVI projects LIMBO (project no. 19F2029C) and OPAL (project no. 19F20284), the BMBF project SOLIDE (project no. 13N14456) and the EU project KnowGraphs (project no. 860801).

References

1. Athreya, R.G., Ngonga Ngomo, A.C., Usbeck, R.: Enhancing community interactions with data-driven chatbots-the DBpedia chatbot. In: Companion of the the Web Conference 2018 on The Web Conference, pp. 143–146 (2018). International World Wide Web Conferences Steering Committee (2018)
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
3. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems, pp. 2787–2795 (2013)
4. Bouma, G.: Normalized (pointwise) mutual information in collocation extraction. In: Proceedings of GSCL, pp. 31–40 (2009)
5. Ciampaglia, G.L., Shiralkar, P., Rocha, L.M., Bollen, J., Menczer, F., Flammini, A.: Computational fact checking from knowledge networks. PloS One **10**(6), e0128193 (2015)
6. d'Amato, C., Fanizzi, N., Esposito, F.: Inductive learning for the semantic web: what does it buy? Semant. Web **1**(1, 2), 53–59 (2010)
7. Galárraga, L.A., Teflioudi, C., Hose, K., Suchanek, F.: Amie: association rule mining under incomplete evidence in ontological knowledge bases. In: Proceedings of the 22nd International Conference on World Wide Web, pp. 413–422. ACM (2013)
8. Gerber, D., et al.: DeFacto–temporal and multilingual deep fact validation. Web Semant. Sci. Serv. Agents World Wide Web **35**, 85–101 (2015)
9. Lao, N., Cohen, W.W.: Relational retrieval using a combination of path-constrained random walks. Mach. Learn. **81**(1), 53–67 (2010)
10. Lehmann, K., Turhan, A.-Y.: A framework for semantic-based similarity measures for $\mathcal{ELH}$-concepts. In: del Cerro, L.F., Herzig, A., Mengin, J. (eds.) JELIA 2012. LNCS (LNAI), vol. 7519, pp. 307–319. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33353-8_24
11. Lin, P., Song, Q., Wu, Y.: Fact checking in knowledge graphs with ontological subgraph patterns. Data Sci. Eng. **3**(4), 341–358 (2018)
12. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)

13. Malyshev, S., Krötzsch, M., González, L., Gonsior, J., Bielefeldt, A.: Getting the most out of wikidata: semantic technology usage in wikipedia's knowledge graph. In: Vrandečić, D., et al. (eds.) ISWC 2018. LNCS, vol. 11137, pp. 376–394. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00668-6_23
14. Nickel, M., Tresp, V., Kriegel, H.P.: Factorizing yago: scalable machine learning for linked data. In: Proceedings of the 21st International Conference on World Wide Web, pp. 271–280. ACM (2012)
15. Paulheim, H., Bizer, C.: Type Inference on noisy RDF data. In: Alani, H., et al. (eds.) ISWC 2013. LNCS, vol. 8218, pp. 510–525. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41335-3_32
16. Röder, M., Usbeck, R., Ngonga Ngomo, A.: GERBIL - benchmarking named entity recognition and linking consistently. Semant. Web **9**(5), 605–625 (2018)
17. Shi, B., Weninger, T.: Discriminative predicate path mining for fact checking in knowledge graphs. Knowl.-Based Syst. **104**, 123–133 (2016)
18. Shiralkar, P., Flammini, A., Menczer, F., Ciampaglia, G.L.: Finding streams in knowledge graphs to support fact checking. In: 2017 IEEE International Conference on Data Mining (ICDM), pp. 859–864. IEEE (2017)
19. Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: Advances in Neural Information Processing Systems.,pp. 926–934 (2013)
20. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th InternationalCconference on World Wide Web, pp. 697–706. ACM (2007)
21. Sun, Y., Han, J., Yan, X., Yu, P.S., Wu, T.: Pathsim: meta path-based top-k similarity search in heterogeneous information networks. Proc. VLDB Endow. **4**(11), 992–1003 (2011)
22. Syed, Z.H., Röder, M., Ngonga Ngomo, A.C.: Factcheck: Validating rdf triples using textual evidence. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 1599–1602. ACM (2018)
23. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Twenty-Eighth AAAI Conference on Artificial Intelligence (2014)
24. Yin, X., Han, J., Philip, S.Y.: Truth discovery with multiple conflicting information providers on the web. IEEE Trans. Knowl. Data Eng. **20**(6), 796–808 (2008)
25. Zhao, M., Chow, T.W., Zhang, Z., Li, B.: Automatic image annotation via compact graph based semi-supervised learning. Knowl.-Based Syst. **76**, 148–165 (2015)