# Using Compositional Embeddings for Fact Checking

Ana Alexandra Morim da Silva[0000−0002−1614−2391], Michael Röder[0000−0002−8609−8277], and Axel-Cyrille Ngonga Ngomo[0000−0001−7112−3516]

DICE group, Department of Computer Science, Paderborn University, Germany
{michael.roeder|axel.ngonga}@uni-paderborn.de

**Abstract.** Unsupervised fact checking approaches for knowledge graphs commonly combine path search and scoring to predict the likelihood of assertions being true. Current approaches search for said metapaths in the discrete search space spanned by the input knowledge graph and make no use of continuous representations of knowledge graphs. We hypothesize that augmenting existing approaches with information from continuous knowledge graph representations has the potential to improve their performance. Our approach ESTHER searches for metapaths in compositional embedding spaces instead of the graph itself. By being able to explore longer metapaths, it can detect supplementary evidence for assertions being true that can be exploited by existing fact checking approaches. We evaluate ESTHER by combining it with 10 other approaches in an ensemble learning setting. Our results agree with our hypothesis and suggest that all other approaches can benefit from being combined with ESTHER by 20.65% AUC-ROC on average. Our code is open-sourceand can be found at at https://github.com/dice-group/esther.

## 1 Introduction

Large knowledge graphs (KGs) such as the Google Knowledge Graph [23], DBpedia [3], and WikiData [18] are now of the backend of a growing number of data-driven applications including Web search [23], community-support systems [2] and personal assistants [18] with several billion users in total. Ensuring the veracity of the assertions in such KGs has hence become mission-critical for the KG community. However, the sheer size of most KGs makes a manual verification difficile. Consequently, automated methods for ensuring the veracity of the assertions found in knowledge graphs (called *fact validation* [21] or *fact checking* [9]) are becoming indispensable.

Unsupervised fact checking approaches for KGs commonly combine path search and scoring to predict the likelihood of assertions being true [28]. To achieve this goal, several approaches rely on identifying metapaths [25,33,13] or corroborative paths [28] that are correlated with the predicate of the assertion to check. State-of-the-art approaches search for paths in the discrete search space spanned by the input KG graph and make no use of continuous representations, i.e., embeddings of KGs [4,16,24,31,7]. We hypothesize that *augmenting existing approaches with embeddings has the potential to improve their performance*. Our approach ESTHER searches for metapaths by exploiting a compositional embedding of the input KG instead of the discrete representation of the graph. By being able to explore longer metapaths, it can detect supplementary evidence for assertions being true that can be exploited by existing fact checking

approaches. We evaluate ESTHER by combining it with 10 other approaches in an ensemble learning setting. The results we obtained on the benchmark datasets FB15k-237 and WN18RR corroborate our hypothesis and suggest that nearly all other approaches benefit from being combined with ESTHER by 20.65% AUC-ROC on average.

The rest of this paper is structured as follows. In the next section, related work is described. Section 3 describes preliminaries for our approach, which is presented in Section 4. Section 5 describes our evaluation and presents our results, which are further discussed in Section 6. Section 7 concludes the paper.

## 2 Related Work

Fact checking approaches can be divided into (1) approaches that rely on unstructured *textual* sources [9,14,27] and (2) approaches that use *structured* reference knowledge [4,21,22,24]. In this work, we focus on the second category of approaches—especially on those approaches that use a knowledge graph as reference. Path-based approaches regard a given KG as a labeled directed graph with entities as nodes and relations as edges connecting these nodes. Given an assertion in the form of a triple $(s, p, o)$, Ciampaglia et al. [6] propose to search within the reference KG for the shortest paths up to length $k$ that (1) connect $s$ and $o$, and (2) are semantically similar to $p$. Their Knowledge Linker (KL) system measures this similarity based on the specificity of the path, i.e., the degree of intermediate nodes of the path. Shiralkar et al. [22] extend this idea by using the co-occurrence of properties to calculate their similarity. They propose KL-Rel as an extension of KL and Knowledge Stream (KS), which relies on multiple paths and the maximum flow between $s$ and $o$. They compare their approaches with other approaches to rank paths proposed by Jeh et al. [10], Katz [12] and Xu et al. [32], and approaches that measure the similarity between two entities proposed by Adamic et al. [1], Liben-Nowell et al. [15] and Shi et al. [20]. Syed et al. [28] propose the usage of RDF Schema information, i.e., the domain and range of $p$. Their approach COPAAL identifies metapaths between $s$ and $o$ and uses the domain and range information of $p$ to identify the set of possible subjects and objects for $p$ in the knowledge graph. Based on this information, it approximates the normalized pointwise mutual information between the metapaths and $p$ to identify paths that corroborate the given fact.

In contrast to these unsupervised approaches, several supervised approaches relying on metapaths have been proposed [25,33,13]. For example, Lao et al. [13] present PRA, which searches for metapaths in the knowledge graph and extracts features with these paths to train a classifier. While these metapaths have to be extracted manually by experts, Shi et al. [21] propose a method to automatically extract metapaths—called *anchored predicate* paths—given a set of labeled examples. Their approach PredPath relies on the `rdf:type` information contained in the input knowledge graph. All these path-based approaches are limited to shorter paths. Most of them have not been evaluated beyond a length of 3 predicates. This is caused by the large amount of longer paths that exist between $s$ and $o$ which lead to very high run times. Li et al. [14] propose Facty that combines evidence from different sources. Facty searches for single triples within the reference knowledge graph that contain $s$ and $o$ but may have a different predicate. These triples are used as pieces of evidence. In contrast to the previously mentioned

approaches, Facty takes also textual sources into account. The extracted triples are combined with evidence from other sources like web searches, query logs and web tables. The authors propose a knowledge fusion algorithm that takes the pieces of evidence and information about their sources as input to calculate a final veracity score.

A related field of research which already makes use of knowledge graph embeddings is the area of link prediction [4,16,24,31]. However, the problems of link prediction and fact checking are different. In link prediction, the goal is to compute how likely it is that any assertion whose subject, predicate and object belong to the input graph $\mathcal{G}$ should belong to a complete version of $\mathcal{G}$ [19]. Fact checking focuses on checking a single, given assertion based on the given graph [9,22,27,28]. Key difference between these two fields also include their runtimes and applications.Fact checking algorithms are typically used in online scenarios while link prediction algorithms are used offline [28].

## 3 Preliminaries

This section introduces concepts that are necessary to understand our approach. It covers the definitions of RDF knowledge graphs, corroborative paths [28] and knowledge graph embeddings [4,26,17].

**Definition 1 (RDF knowledge graph).** *Let $\mathbb{E}, \mathbb{B}, \mathbb{P}, \mathbb{L}$ be the sets of all RDF resources, blank nodes, RDF predicates, and literals, respectively. Let $\mathbb{E}, \mathbb{B}$ and $\mathbb{L}$ be mutually disjoint and $\mathbb{P} \subset \mathbb{E}$. An RDF KG $\mathcal{G}$ is defined as a set of RDF triples of the form $(s, p, o)$ with $\mathcal{G} \subset (\mathbb{E} \cup \mathbb{B}) \times \mathbb{P} \times (\mathbb{E} \cup \mathbb{B} \cup \mathbb{L})$.*

$\mathcal{G}$ can be regarded as a labeled directed graph, with triples being directed edges labeled with the property $p$, with the nodes $s$ as head and $o$ as tail. We define inverted edges by means of the inverse property $p^{-1}$ for an existing property $p$ as follows: $(s, p, o) \Leftrightarrow (o, p^{-1}, s)$.

### 3.1 Corroborative Paths

**Definition 2 (Path).** *A path of length $k$ in a knowledge graph $\mathcal{G}$ is a sequence of triples from $\mathcal{G}$ of the form $(v_0, p_1, v_1), (v_1, p_2, v_2), ..., (v_{k-1}, p_k, v_k)$ [28].*

Several paths can exist between two nodes $v_0$ and $v_k$. We use $\pi^k(v_0, v_k)$ to denote these paths. Following [28], we define $\gamma$ as a function from $\mathbb{E} \cup \mathbb{P} \cup \mathbb{B} \cup \mathbb{L}$ to the set of all RDFS classes, where $\gamma(v)$ is the set of all RDFS classes that $v$ is an instance of. For example, if the RDFS classes (also called *types*) that could be inferred from a given graph $\mathcal{G}$ for the entity `BarackObama` using RDFS semantics were exactly `Person`, `Politician` and `OfficeHolder`, we would write $\gamma$(`BarackObama`) = { `Person`, `Politician`, `OfficeHolder`}. Let $\lambda$ be a function that maps a given set of types $t_x$ to a set of resources that are instances of at least one element of $t_x$ by virtue of RDFS semantics.

**Definition 3 (Typed paths).** *The set of typed paths $\Pi^k_{(t_x, t_y)}$ of length $k$ between vertices of types $t_x$ and $t_y$ in a knowledge graph $\mathcal{G}$ are defined as follows [28]:*

$$\Pi^k_{(t_x, t_y)} = \{\pi^k(v_0, v_k) \mid t_x \subseteq \gamma(v_0) \wedge t_y \subseteq \gamma(v_k)\}. \quad (1)$$

These paths can be further restricted by using a vector of properties $\vec{q} = q_1, \ldots, q_k$:

**Definition 4 ($\vec{q}$-restricted typed paths).** *The set of $\vec{q}$-restricted typed paths $\Pi^k_{(t_x,t_y),\vec{q}} \subseteq \Pi^k_{(t_x,t_y)}$ is defined as follows [28]:*

$$\Pi^k_{(t_x,t_y),\vec{q}} = \big\{\pi^k(v_0,v_k) \;\big|\; \pi^k(v_0,v_k) \in \Pi^k_{(t_x,t_y)}, \\ \forall i \in [0,k-1] : (v_i, p_{i+1}, v_{i+1}) \in \pi^k(v_0,v_k) \rightarrow p_{i+1} = q_{i+1}\big\}. \tag{2}$$

This is the set of typed paths that have exactly the properties of $\vec{q}$ as predicates of the sequence of triples the paths consist of. These paths are used by Syed et al. [28] to identify paths that corroborate the correctness of the given fact $(s, p, o)$. To define the set of corroborative paths, we use $R(p)$ to denote the set of all types $t$ so that `p rdfs:range t` can be inferred from the input knowledge graph using RDFS semantics. We also account for the practical use of our approach by considering the set $R'(p)$, which we defined as the set of classes such that the assertion `p rdfs:range t` is explicitly stated in the input knowledge graph. $D(p)$ and $D'(p)$ are defined analogously for `rdfs:domain`.

**Definition 5 (Corroborative paths).** *The* corroborative paths *for a predicate $p$ are defined as follows [28]:*

$$\Pi^k(p) = \bigcup_{j=1}^{k} \Pi^j_{(D(p),R(p))}. \tag{3}$$

### 3.2 Knowledge Graph Embeddings

KGs are a discrete representation of knowledge. They can be embedded into a continuous space via a knowledge graph embedding (KGE). Various algorithms have been proposed to generate KGEs. Because our approach assumes that a KGE has already been generated and due to the limited space, we focus on the features of the generated KGEs and refrain from presenting much details on the single algorithms that generate them.[1] Each KGE used in this paper comes with a number of dimensions in the embedding space ($n$) and a mapping function $e(\cdot)$ that maps an RDF resource of $\mathcal{G}$ to a vector representation within the embedding space.

**Definition 6 (Compositional KGE).** *Let $p_1$,$p_2$ and $p_3$ be properties and $x, y, z$ be nodes in the KG. A KGE is compositional if the following holds:*

$$\begin{aligned} &(x,p_1,y) \wedge (y,p_2,z) \Rightarrow (x,p_3,z) \quad (\forall x,y,z) \\ \Leftrightarrow \quad &e(p_1) \oplus e(p_2) \approx e(p_3) \end{aligned} \tag{4}$$

*where $\oplus$ is an operator that combines the embedding vectors of two properties.*

We base our search for paths in an embedding space on the compositionality assumption. Hence, we work with the following compositional KGE algorithms: TransE [4], RotatE [26], DensE [17].

[1] We refer to [30] for a survey of KGE techniques.

**Table 1.** Summary of the KGE related operations used by ESTHER and their implementation in TransE, RotatE and DensE. $\circ$ denotes the Hadamard product, $\otimes$ the Hamilton product, $\overline{e}$ the complex conjugate and $e(p)^{-1}$ the inverse of a quaternion.

| | **Mapping function** | **Composition** | **Inversion** |
|---|---|---|---|
| ESTHER | $e(\cdot)$ | $e(p_1) \oplus e(p_2)$ | $e(p^{-1})$ |
| TransE | $\mathbb{E} \to \mathbb{R}^n$ | $e(p_1) + e(p_2)$ | $-e(p)$ |
| RotatE | $\mathbb{E} \to \mathbb{C}^n$ | $e(p_1) \circ e(p_2)$ | $\overline{e(p)}$ |
| DensE | $\mathbb{E} \to \mathbb{H}^n$ | $e(p_1) \otimes e(p_2)$ | $e(p)^{-1}$ |

***TransE*** represents the property $p$ in an assertion $(s, p, o)$ as a translation from $s$ to $o$. This is accomplished through the minimization of the L1 or L2 norm between the $e(s) + e(p)$ and $e(o)$ [4]. The model attempts to maximize the score function

$$\delta_{\text{TransE}} = -||e(s) + e(p) - e(o)||. \tag{5}$$

***RotatE*** models the predicate $p$ in an assertion $(s, p, o)$ as a rotation. The predicates are represented as complex numbers. Like TransE, RotatE also aims to approximate the subject and predicate vector with the object entity's vector. $\overline{e}$ the complex conjugate.

$$\delta_{\text{RotatE}} = -||e(s) \circ e(p) - e(o)|| \tag{6}$$

$$||e(\cdot)|| = \sum_{i}^{n} ||e(\cdot)_i|| = \sum_{i}^{n} \left| \sqrt{e(\cdot)_i \overline{e(\cdot)_i}} \right| \tag{7}$$

***DensE*** also represents predicates as rotations from the subject to the object entity. However, it does so by considering 3D rotations followed by a scaling factor on the subject entity. The predicates are therefore represented by quaternions. The quaternion modelling allows for non-abelian composition patterns, dependent on the operation direction. The dissimilarity function used is the L2-norm. $\mathcal{O}(\cdot)$ denotes the transformation applied on the entity such that $e(o)_i = \mathcal{O}(e(p)_i)e(s)_i$.

$$\delta_{\text{DensE}} = -\frac{1}{2}\left(||\mathcal{O}(e(p))e(s) - e(o)|| + ||\mathcal{O}(e(p)^{-1})e(o) - e(s)||\right) \tag{8}$$

# 4 Approach

## 4.1 Intuition

ESTHER is built on the assumption that existing paths between $s$ and $o$ can corroborate the existence of the triple $(s, p, o)$. Hence, it searches for corroborative paths $\Pi^k(p)$ as suggested by Syed et al. [28]. However, in contrast to the state of the art, ESTHER performs this search in a continuous space. There, the embedding of corroborative paths have a similar direction and length as the embedding of $p$. ESTHER identifies candidates for corroborative paths by utilizing the A* search algorithm. In a second step, the

identified paths are scored based on their statistical co-occurrence with $p$. Paths that corroborate the occurrence of $p$ are used as corroborative paths in the third step. This final step checks whether the given subject $s$ and object $o$ are connected with these paths. In the following, we describe the three steps in more detail.

### 4.2 Combining Properties to Paths

ESTHER's main objective is to assess the veracity of a given triple $(s, p, o)$ by leveraging a compositional embedding model of the reference graph to find corroborative paths for the property $p$. These paths are searched in the embedding space. A good candidate for a corroborative path is a $q$-restricted path that (1) connects the domain and range of $p$ and (2) has an embedding that is similar to the embedding of $p$. The embedding of the path is computed by combining the embeddings of the properties in $\vec{q}$:

$$\bigoplus_{i=1}^{|\vec{q}|} e(q_i) \approx e(p) \tag{9}$$

Previous approaches showed that it is beneficial for the path search to be able to use the directed edges between two vertices in both directions [22,28]. ESTHER leverages this idea by considering inverse properties for all $p$ with $\forall(s, p, o) \in G, \exists(o, p^{-1}, s)$. As such, a set of inverse properties is defined as $\mathbb{P}_{\mathcal{G}}^{-1} = \{p^{-1} \forall p \in \mathbb{P}_{\mathcal{G}}\}$ and the joint set $\mathbb{P}_{\mathcal{G}}^{*} = \mathbb{P}_{\mathcal{G}} \cup \mathbb{P}_{\mathcal{G}}^{-1}$ to aid in bidirectional path-finding.

When concatenating properties to create paths, the schema of the knowledge graph has to be taken into account since not all properties can be freely combined with each other. We defined an extended $|\mathbb{P}_{\mathcal{G}}^{*}| \times |\mathbb{P}_{\mathcal{G}}^{*}|$ property-adjacency matrix $\mathcal{M}$ that indicates whether two properties can be adjacent in a path. Since $\mathcal{G}$ is a directed graph, the pair of properties $(p_i, p_j)$ are adjacent if the range of the first property, $R(p_i)$, fits to the domain of the second, $D(p_j)$. The matrix expresses this as follows:

$$\mathcal{M}_{i,j} = \begin{cases} 1, & \text{if properties } p_i \text{ and } p_j \text{ can be adjacent} \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

ESTHER implements five different modes to decide whether two properties fit to each other with respect to their domain and range. They are defined as follows:

| | | |
|---|---|---|
| Strict equality (S) : | $\mathcal{M}_{i,j} = 1 \Leftrightarrow R'(p_i) = D'(p_j)$ | (11) |
| Subsumed (SU) : | $\mathcal{M}_{i,j} = 1 \Leftrightarrow R(p_i) \supseteq D(p_j)$ | (12) |
| Non-disjoint (ND) : | $\mathcal{M}_{i,j} = 1 \Leftrightarrow R'(p_i) \cap D'(p_j) \neq \emptyset$ | (13) |
| Non-disjoint subsumption (NDS) : | $\mathcal{M}_{i,j} = 1 \Leftrightarrow R(p_i) \cap D(p_j) \neq \emptyset$ | (14) |
| Irrelevant (I) : | $\forall p_i, p_j \in \mathbb{P}_{\mathcal{G}}^{*}, \mathcal{M}_{i,j} = 1$ | (15) |

It can be seen that all modes rely on the range and domain of the properties except the I mode, which allows the combination of all properties.

Syed at al. [28] exclude paths with a loop in their search, i.e., while exploring the graph, the search algorithm is not allowed to visit a node twice. However, we aimed

**Algorithm 1:** ESTHER's path search algorithm

**Input:** $p$, $N$, $k$, $e(\cdot)$, $\mathbb{P}_{\mathcal{G}}$, $\mathcal{M}$, $\mathcal{G}$
**Output:** A set of corroborative paths $\Pi^k(p)$

$\Pi^k(p) \leftarrow \{\}$;
$Q \leftarrow \{\}$;
**for** $i = 1$ *to* $|\mathbb{P}_{\mathcal{G}}|$ **do**
    // Add properties with a domain that matches the domain of $p$ according to $\mathcal{M}$
    **if** $D(p_i) = D(p)$ **then**
        // the queue takes two values: a path and its priority (i.e., its distance to $p$)
        Q.add($\{p_i\}$,$-||e(p_i) - e(p)||$);
    **end**
**end**
**while** $(|Q| > 0)$ && $(|\Pi^k(p)| < N)$ **do**
    $\vec{q} \leftarrow Q.poll()$;
    **if** $|\vec{q}| <= k$ **then**
        // If the range of the last property in the path equals $p$'s range
        **if** $R(\vec{q}_{|\vec{q}|-1}) = R(p)$ **then**
            P.add(path);
        **end**
    **end**
    **if** $|\vec{q}| < k$ **then**
        // Extend this path
        **for** $i = 1$ *to* $|\mathcal{M}|$ **do**
            **if** $\mathcal{M}_{\vec{q}_{|\vec{q}|-1}, p_i} = 1$ **then**
                Q.add($\vec{q} \cup p_i$,$-||e(\vec{q}) - e(p)||$);
            **end**
        **end**
    **end**
**end**
**return** $\Pi^k(p)$;

to quantify the effect of loops on our approach. Hence, ESTHER can be configured to allow or disallow loops. If loops are not allowed, the property $p_i$ can not be added to $\vec{q}$ when extending a path if $\vec{q}$ already contain its opposite $p_i^{-1}$.

### 4.3 Path Search

ESTHER uses the A* search algorithm to find the $N$ best corroborative path candidates for a given property $p$. Let $d$ be a distance measure in the embedding space. The A* search is configured to search for paths with a length up to $k$ that minimize the distance to the property embedding. To this end, the A* search should minimize the error $\varepsilon$:

$$\min(\varepsilon) = \min\left(d(e(\vec{q}), e(p)) + \eta|\vec{q}|\right) \tag{16}$$

where $\eta$ is a weight that allows to penalize longer paths. Algorithm 1 shows the pseudo code for the path search. A priority queue is used to sort the incomplete path candidates

according to their error $\varepsilon$. The queue is initialized with all properties that share the same domain as $p$. In each step, the best incomplete path from the queue is selected and combined with new properties based on $\mathcal{M}$. A new corroborative path is found when the newly added property has the same range as $p$. The search stops as soon as $N$ corroborative paths have been found or all possible paths with length $k$ have been checked. Table 2 shows example corroborative paths that have been identified by the search algorithm for the predicate `nationality` in the FB15k-237 dataset (see Section 5). The paths show that ESTHER will make use of information like (1) the nationality of other people that married in the place of birth of the subject, (2) the nationality of siblings of people that were born in places at which the subject lived, and (3) the countries in which the language of a subject is spoken.

### 4.4 Path Scoring

The result of the previous step is a set of corroborative paths $\mathit{\Pi}^k(p)$. The second step scores these paths by measuring their cooccurrence with $p$ within $\mathcal{G}$. Previous works [28] point out that deriving the necessary path counts is computationally expensive and provide a heuristic to compute the normalized pointwise mutual information for a $q$-restricted path and $p$. We reuse the heuristics for ESTHER but make use of the positive NPMI (PNPMI) for the path scores. Preliminary results showed that most negative NPMI values (1) were very small and, hence, statistically not reliable, and (2) reduced the performance of ESTHER. Let $\mathcal{P}(p)$ be the probability that a random triple has the property $p$ as predicate and let $\hat{\mathcal{P}}$ denote approximated probabilities. We calculate the probability of a $q$-restricted path, the probability of the cooccurrence of a $q$-restricted path and p, and the approximation of the PNPMI as follows:

$$\hat{\mathcal{P}}\left(\mathit{\Pi}^k_{(t_x,t_y),\vec{q}}\right) = \frac{|\mathit{\Pi}^k_{(t_x,t_y),\vec{q}}|}{|\lambda(t_x)| \cdot |\lambda(t_y)|} \tag{17}$$

$$\hat{\mathcal{P}}\left(\mathit{\Pi}^k_{(t_x,t_y),\vec{q}}, p\right) = \frac{|\{\pi^k(a,b) \in \mathit{\Pi}^k_{(t_x,t_y),\vec{q}} : (a,p,b) \in \mathcal{G}\}|}{|\lambda(t_x)| \cdot |\lambda(t_y)|} \tag{18}$$

$$\widehat{\text{PNPMI}}(\mathit{\Pi}^k_{(t_x,t_y),\vec{q}}, p) = \max\left(\frac{\log\left(\frac{\hat{\mathcal{P}}\left(\mathit{\Pi}^k_{(t_x,t_y),\vec{q}}, p\right)}{\hat{\mathcal{P}}\left(\mathit{\Pi}^k_{(t_x,t_y),\vec{q}}\right)\mathcal{P}(p)}\right)}{-\log\left(\hat{\mathcal{P}}\left(\mathit{\Pi}^k_{(t_x,t_y),\vec{q}}, p\right)\right)}, 0\right). \tag{19}$$

The calculation has to handle outliers which can be caused by the approximation. To this end, we define the score of the path $\mathit{\Pi}^k_{(D(p),R(p)),\vec{q}}$ dubbed $\zeta_{p,\vec{q}}$ as follows:

$$\zeta_{p,\vec{q}} = \min\left(1, \widehat{\text{PNPMI}}(\mathit{\Pi}^k_{(D(p),R(p)),\vec{q}}, p)\right) \tag{20}$$

Table 2 shows the scores for the three example paths.

### 4.5 Veracity calculation

The veracity calculation of a single fact $(s, p, o)$ is done by checking whether the subject $s$ and object $o$ of the fact are connected by corroborative paths of the previously

**Table 2.** Example paths for the predicate `nationality` found in the FB15k-237 dataset.

| $q$-restricted path | $\varepsilon$ | $\zeta_{p,\vec{q}}$ |
|---|---|---|
| $\xrightarrow{\texttt{place\_of\_birth}}\ \xleftarrow{\texttt{marriage.location\_of\_ceremony}}\ \xrightarrow{\texttt{nationality}}$ | 7.87 | 0.47 |
| $\xrightarrow{\texttt{people.place\_lived.location}}\ \xleftarrow{\texttt{place\_of\_birth}}\ \xrightarrow{\texttt{sibling}}\ \xrightarrow{\texttt{nationality}}$ | 9.30 | 0.27 |
| $\xrightarrow{\texttt{languages}}\ \xrightarrow{\texttt{countries\_spoken\_in}}$ | 10.88 | 0.07 |

determined set $\Pi^k(p)$. Let $Z$ be the set of the path scores $\zeta_{p,\vec{q}}$ of all corroborative paths $\Pi^k_{(D(p),R(p)),\vec{q}} \in \Pi^k(p)$ that connect $s$ and $o$ at least once. The final truth score $\tau$ is calculated as the cubic mean of the scores in $Z$.[2] In the special case, that no corroborative paths could be identified for $p$ 0.0 is returned. If corroborative paths have been found but none of them exists between $s$ and $o$ $-1$ is returned.

$$\tau = \begin{cases} 0.0 & \text{if } \Pi^k(p) = \emptyset \\ \text{-1} & \text{if } (\Pi^k(p) \neq \emptyset) \wedge (Z = \emptyset) \\ \sqrt[3]{\frac{1}{|Z|}\sum_{\zeta_{p,\vec{q}} \in Z} \zeta^3_{p,\vec{q}}} & \text{else} \end{cases} \tag{21}$$

It is worth noticing that only the last step of ESTHER relies on the fact to be checked. In a fact checking scenario, the search for corroborative paths and their scoring can be done in a pre-processing step. The service that checks the single facts only has to perform the veracity calculation step. This is different to approaches other approaches like KL [6], KS [22] and COPAAL [28] that have to perform their search for paths based on the given fact.

### 4.6 Complexity Analysis

The complexity of ESTHER can be derived by determining the complexity of (1) the generation of the property-adjacency matrix, (2) the path finding algorithm and (3) the calculation of the PNPMI values for the top-$N$ paths for each of the predicates. The first step is a pairwise comparison of properties and has a time and a space complexity of $O(|\mathbb{P}_\mathcal{G}|^2)$. The second step is based on the A* algorithm, which has a time complexity of $O(|\mathbb{P}_\mathcal{G}|^k)$. A single PNPMI value relies on the number of paths, the number of predicates and the number of pairs which are connected by both. Deriving the counts for the paths and the pairs that both have in common is the expensive part which grows linearly with respect to the length of the paths $k$. This has to be done for all $N$ top paths for each predicate that ESTHER should support in the fact checking step. This leads to a time and space complexity of $O(kN|\mathbb{P}_\mathcal{G}|)$. Hence, the setup of ESTHER for a given knowledge graph has a time and a space complexity of $O(|\mathbb{P}_\mathcal{G}|^k + |\mathbb{P}_\mathcal{G}|^2 + kN|\mathbb{P}_\mathcal{G}|) = O(|\mathbb{P}_\mathcal{G}|^k + kN|\mathbb{P}_\mathcal{G}|)$. It should be noted that the generation of the KGE is not part of the complexity as we assume the embedding as given.

[2] Preliminary tests showed a good performance for the cubic mean in comparison to the arithmetic mean and the quadratic mean.

**Table 3.** Data statistics of FB15k-237 and WN18RR

| | FB15k-237 | WN18RR |
|---|---|---|
| Entities | 14 541 | 40 943 |
| Relations | 237 | 11 |
| Triples | 289 650 | 89 869 |

To check a single fact, the previously identified paths for $p$ are used. In the worst case $N$ corroborative paths have to be checked. ESTHER checks whether these paths exist between $s$ and $o$ of the given fact. Hence, this check has a complexity of $O(kN)$.

## 5 Evaluation

### 5.1 Datasets

We use the datasets FB15k-237 [29] and WN18RR [8]. These datasets have a size that permits the computation of embeddings in a reasonable time and, hence, have a widespread usage in works related to KGE. Table 3 gives statistical information about the datasets. Both datasets are divided in a training, validation and test split. We generate embeddings based on the training and validation data and extend the test data to be used for fact checking. We extend the two knowledge graphs with their respective ontologies (incl. type information) to ensure that the fact checking approaches can make use of them.[3] A class hierarchy is required to make use of the SU and NDS modes of ESTHER. For WN18RR, the class hierarchy is present in its ontology. However, since Freebase does not support a class hierarchy [5], we inferred the hierarchy from the existing data in FB15k-237. Given two types $t_x$ and $t_y$, we consider $t_x$ to be a subclass of $t_y$ if all instances of $t_x$ are instances of $t_y$, i.e., $\lambda(t_x) \subseteq \lambda(t_y)$.

Each dataset's test split is a set of true facts. For a fact checking experiment, a set of false facts is needed. We adopt the approach in [9] and randomly sample 750 triples, which we then corrupt to create false triples. The false triples are generated by corrupting the subject, the object and both the subject and object, each $\frac{1}{3}$ of the time. Entities are replaced with random entities of the same type as the original entities.[4]

### 5.2 Setup

We evaluate ESTHER in three experiments. In all experiments, the effectiveness of each fact checking approach is measured using the area under ROC (AUC-ROC), the area under precision recall curve (AUC-PR) and the F1-measure. The latter needs a threshold to separate positive and negative classes. We use a threshold that maximizes each approach's F1-measure.

In our first experiment, we evaluate different configurations of ESTHER on both datasets. Current surveys present over 40 different KGE approaches [11]. We have to

[3] The ontology for FB15k-237 is available at `https://github.com/knowledgegraph/schema`. The ontology for WN18RR was adapted from

**Table 4.** Hyper-parameters used to generate TransE, RotatE and DensE embeddings.

| | FB15k-237 | | | WN18RR | | |
|---|---|---|---|---|---|---|
| | **TransE** | **RotatE** | **DensE** | **TransE** | **RotatE** | **DensE** |
| Dimensions | 1000 | 1000 | 500 | 500 | 500 | 200 |
| Learning rate | 0.00005 | 0.00005 | 0.0001 | 0.00005 | 0.00005 | 0.0001 |
| Batch size | 1024 | 256 | 512 | 512 | 128 | 256 |
| Iterations | 100 000 | 100 000 | 100 000 | 80 000 | 80 000 | 100 000 |
| Margin | 9.0 | 9.0 | 9.0 | 6.0 | 6.0 | 12.0 |
| Adversarial temperature | 1.0 | 1.0 | 1.0 | 0.5 | 0.5 | 0.3 |
| Neg. sample size | 256 | 256 | 256 | 1024 | 512 | 1024 |

choose a subset of the available algorithms due to limited resources. We use TransE, RotatE and DensE because they (1) are compositional embeddings, (2) represent diverse embedding spaces (real numbers, complex numbers and quaternions) and (3) are well cited.[5]The parameters used for the generation of the embeddings are listed in Table 4 and are taken from the respective publications since they were suggested for the two datasets. Only the batch size was reduced to make the embeddings work on our GPU. For each KGE, we run ESTHER with all different modes, a varying maximum length of paths $k = [1, 6]$ and different numbers of top paths $N = \{10, 20, 50, 100, 200, 500\}$. All runs are executed twice—with and without allowing loops in the paths. In all runs, the penalty for long paths $\eta$ is set to 1.

In our second experiment, we compare the best performing mode of ESTHER with 10 other approaches that have been used for fact checking, namely: COPAAL [28], KS [22], Katz [12], Pathent [32], Simrank [10], AdamicAdar [1], Jaccard[15], Degree product [20], PredPath [21] and PRA [13].[6] The first 8 approaches are unsupervised while PredPath and PRA are supervised. For the supervised approaches, we perform a 10-fold cross validation to get results for all facts. In addition to the effectiveness, we measure the runtime of the single systems to evaluate their efficiency.[7]

The third experiments combines each of the compared approaches with ESTHER. Let $\mathcal{A}$ be one of the approaches and let $\tau_{(s,p,o),\mathcal{A}}$ be the veracity score that it returns for a given fact $(s, p, o)$. Let $\tau_{(s,p,o),\mathcal{E}}$ be the veracity score returned by ESTHER for

https://www.w3.org/2006/03/wn/wn20/. The added information is not taken into account while generating the embeddings.

[4] The extended datasets can be found at https://hobbitdata.informatik.uni-leipzig.de/esther/.

[5] We use the implementation for TransE and RotatE of https://github.com/DeepGraphLearning/KnowledgeGraphEmbedding and the DensE implementation of https://github.com/anonymous-dense-submission/DensE.

[6] For our experiments, we used the source code provided by Shiralkar et al. [22] in the version of October 31st 2018 (see https://github.com/shiralkarprashant/knowledgestream). However, The source code of KL [6] and KL-Rel [22] did not work for us. Hence, a comparison with these approaches was not possible.

[7] The runtime experiments were conducted on a system with an Intel®Core™i5-7500 CPU @ 3.40GHz, 16 GB RAM and Ubuntu 20.04.2 LTS.

**Table 5.** Configurations of ESTHER that yield the best AUC-ROC values with the different KGEs. AUC-ROC, AUC-PR and F1-measure are shown as percentages.

| | KGE | Mode | Loops | $k$ | $N$ | AUC-ROC | AUC-PR | F1-measure |
|---|---|---|---|---|---|---|---|---|
| FB15k-237 | TransE | S | Y | 4 | 200 | 82.53 | 75.79 | 85.50 |
| | RotatE | S | Y | 4 | 200 | **83.07** | 85.54 | 76.61 |
| | DensE | S | Y | 3 | 200 | 81.82 | 84.48 | 76.60 |
| WNRR18 | TransE | I | Y | 3 | 500 | **77.55** | 88.61 | 71.06 |
| | RotatE | I | Y | 5 | 500 | 73.15 | 85.25 | 66.67 |
| | DensE | I | Y | 2 | 500 | 71.19 | 85.45 | 66.67 |

the same fact. We collect these values for each fact and use them as input for a meta-algorithm. As meta-algorithm, we use different classifiers (Random Forest and SVM) and regression algorithms (REPTree, SMO and REPTree with bagging) which return a classification or a veracity score, respectively.[8] The meta-algorithm is evaluated in a 10-fold cross validation.

### 5.3 Results

The first experiment gives a large amount of results. Due to the limited space, we focus on those results that give us a good insight into ESTHER's performance. The results of the modes S, SU, ND and NDS are comparable in nearly all configurations. Moreover, ESTHER achieves better results if loops are allowed. Hence, we only report results for the modes S and I with loops. Table 5 shows the configurations of ESTHER that achieve the highest AUC-ROC values for the different KGEs and datasets. The influence of $N$ and $k$ is visualized in Figure 1. For FB15k-237, the S mode achieves better results while the I mode yields better results for WN18RR. ESTHER performs better when using TransE or RotatE embeddings than using DensE embeddings.

The second experiment compares the performance of ESTHER with 10 other fact checking approaches. The left half of Tables 6 and 7 show the results. It can be seen that ESTHER performs better than most of the other approaches on both datasets. However, it is outperformed by KS on FB15k-237 and KS, Pathent and PrePath on WN18RR with respect to the AUC-ROC. The results of the runtime comparison are in Figure 2.

The third series of experiments evaluates the combinations of ESTHER with each of the other fact checking approaches. All meta-algorithms led to an increase of the average performance. Random Forest, SVM, SMO and REPTree led to an average improvement of the AUC-ROC of 18.5%, 13.4%, 18.8% and 20.2%. REPTree combined with Bagging led to the highest average improvement of 20.65%. Because of the limited space, only the results of this meta-algorithm are reported in detail in the right half of Tables 6 and 7 for the two datasets, respectively.

[8] We use WEKA for all meta-algorithms. `https://www.cs.waikato.ac.nz/~ml/weka/`

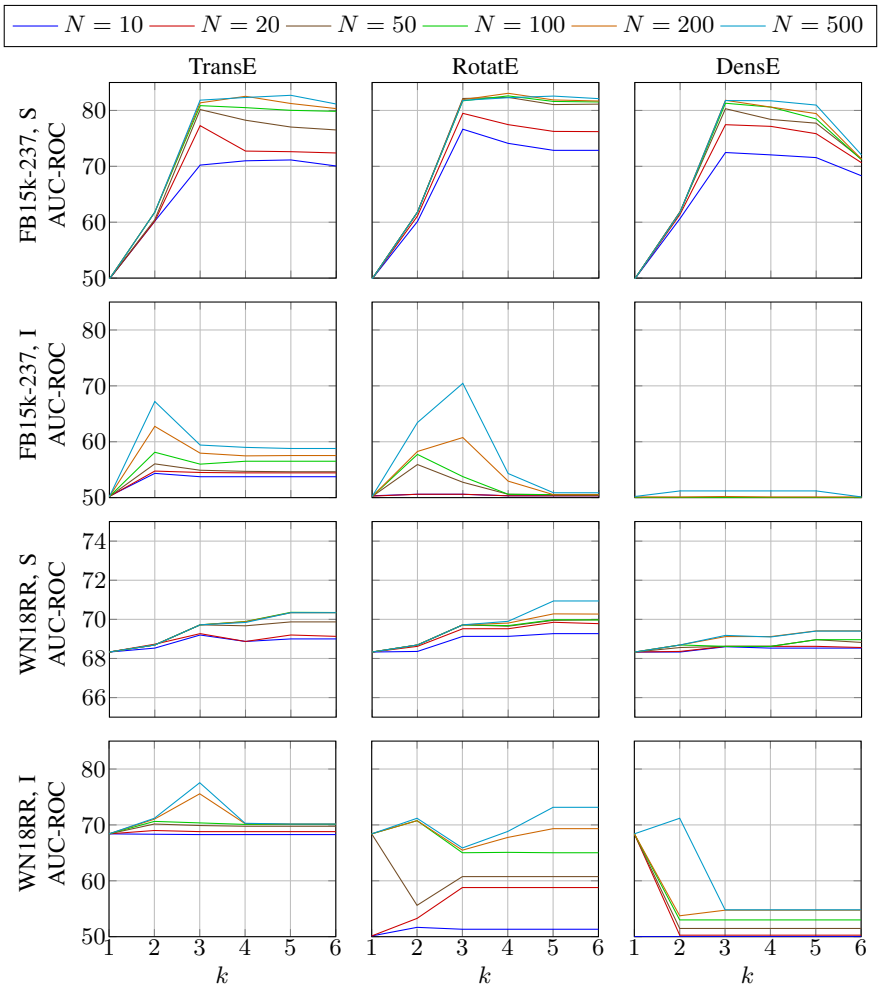


**Fig. 1.** AUC-ROC (in %) results for ESTHER (loops allowed) for both datasets and both modes on different embedding models and varying values for $k$ and $N$.

## 6 Discussion

The experimental results presented in the previous Section led us to several insights. Figure 1 shows that the S mode is more stable than the I mode with respect to the increase of path lengths. Even with a $k$ that is higher than the optimum, the performance remains high for TransE and RotatE models and high values of $N$. ESTHER also shows a robust behavior with high $N$ values, i.e., the performance might decrease only slightly if $N$ is increased while $k$ remains the same. It can be concluded that ESTHER's scoring and veracity calculation are able to filter noisy paths that have been identified by the search but are not helpful for the fact checking task. The I mode shows in most configurations a peak. This is a hint that ignoring the domain and range gives the search

**Table 6.** Performance of other fact checking approaches on FB15k-237 and their performance if they are combined with ESTHER. The number in brackets shows the performance difference.

| Approach | Without ESTHER | | | With ESTHER | | |
|---|---|---|---|---|---|---|
| | AUC-ROC | AUC-PR | F1-score | AUC-ROC | AUC-PR | F1-score |
| COPAAL | 77.42 | 70.13 | 66.67 | 87.12 (+09.70) | 85.53 (+15.40) | 81.40 (+14.73) |
| KS | **87.59** | **83.29** | **82.75** | 89.97 (+02.38) | 89.08 (+05.79) | 83.67 (+00.92) |
| Katz | 82.80 | 80.43 | 78.01 | 86.30 (+03.50) | 85.62 (+05.19) | 78.98 (+00.97) |
| Pathent | 73.46 | 63.87 | 74.68 | 84.75 (+11.29) | 84.72 (+20.85) | 77.53 (+02.85) |
| Simrank | 40.07 | 44.29 | 66.76 | 81.60 (+41.53) | 82.41 (+38.12) | 75.55 (+08.79) |
| AdamicAdar | 72.12 | 72.57 | 70.22 | 85.36 (+13.24) | 85.29 (+12.72) | 78.16 (+07.94) |
| Jaccard | 38.56 | 46.04 | 66.67 | 82.91 (+44.35) | 82.48 (+36.44) | 76.24 (+09.57) |
| Degree product | 77.11 | 76.20 | 72.87 | 83.28 (+06.17) | 83.54 (+07.34) | 78.33 (+05.46) |
| PredPath | 69.87 | 77.25 | 68.30 | 83.76 (+13.89) | 84.33 (+07.08) | 76.95 (+08.65) |
| PRA | 08.53 | 36.56 | 66.67 | **97.44** (+88.91) | **98.09** (+61.53) | **93.44** (+26.77) |

**Table 7.** Comparison of other fact checking approaches with and without ESTHER on WN18RR

| Approach | Without ESTHER | | | With ESTHER | | |
|---|---|---|---|---|---|---|
| | AUC-ROC | AUC-PR | F1-score | AUC-ROC | AUC-PR | F1-score |
| COPAAL | 68.11 | 83.68 | 66.67 | 79.38 (+11.27) | 86.14 (+02.46) | 77.99 (+11.32) |
| KS | **86.44** | **90.85** | **82.96** | **94.92** (+08.48) | **96.17** (+05.32) | **89.75** (+06.79) |
| Katz | 69.96 | 73.19 | 67.97 | 86.22 (+16.26) | 83.19 (+10.00) | 79.97 (+12.00) |
| Pathent | 79.98 | 82.67 | 75.66 | 86.94 (+06.96) | 90.43 (+07.76) | 82.30 (+06.64) |
| Simrank | 44.15 | 46.09 | 66.67 | 82.47 (+38.32) | 87.25 (+41.16) | 75.77 (+09.10) |
| AdamicAdar | 59.86 | 64.79 | 66.67 | 84.22 (+24.36) | 87.78 (+22.99) | 76.40 (+09.73) |
| Jaccard | 42.34 | 47.96 | 66.67 | 87.18 (+44.84) | 90.26 (+42.30) | 80.03 (+13.36) |
| Degree product | 65.57 | 67.80 | 66.67 | 87.43 (+21.86) | 90.39 (+22.59) | 80.39 (+13.72) |
| PredPath | 80.20 | 85.95 | 78.59 | 82.20 (+02.00) | 87.43 (+01.48) | 79.43 (+00.84) |
| PRA | 71.80 | 85.90 | 66.67 | 75.35 (+03.55) | 82.81 (−03.09) | 71.06 (+04.39) |

algorithm the ability to find a large amount of paths that are close the given property but do not exist in practice. These paths can fill the top $N$ in cases with longer $k$ and replace meaningful paths that have been identified with shorter values for $k$. Another hint for this behavior is that the I mode works better on the dataset that contains less properties. This behavior may lead to problems in practice since it will be hard to identify the correct configuration for this peak without training data. Another insight is that ESTHER works more reliable on TransE and RotatE than on DensE models. With respect to the path length, the results on FB15k-237 show that paths of length 4 can lead to better results than shorter paths. This might be an interesting result for similar approaches like COPAAL or KS. The SU, ND and NDS modes have nearly no difference to the S mode. This is caused by the ontologies of both datasets. The SU and NDS mode would show an effect if a class is used as the range of one property and has a subclass that is used as the domain of another property. The ND and NDS mode would show different results

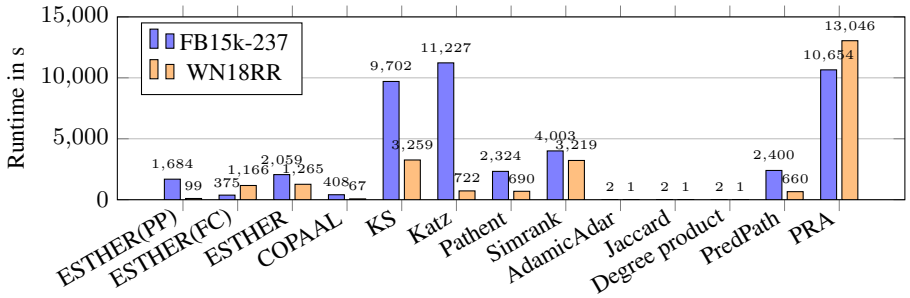


**Fig. 2.** Runtimes of the different approaches in seconds. For ESTHER, we report the runtime of the best performing configurations separated into pre-processing (PP), fact checking (FC) and the complete runtime.

for domain and range definitions that comprise more than a single class. None of the situations occur in WN18RR. In FB15K-237, the S mode allows 667 property combinations while the SU mode allows 1089. This difference does not seem to have an impact on the performance and is small compared to the I mode which allows 56169 combinations. The ND and NDS modes do not add any new combinations in comparison to the S and SU mode, respectively.

The comparison of ESTHER with other approaches shows that it is able to outperform most other approaches—including COPAAL which is based on corroborative paths as well (see Tables 6 and 7). It also shows that KS performs best while Syed et al. [28] found COPAAL to perform better than KS on a DBpedia-based dataset. This underlines that FB15k-237 and WN18RR that have been used intensively in the knowledge graph embedding research area might have different features than a DBpedia or other, larger knowledge graphs. The runtime comparison shows that ESTHER has a better efficiency on FB15k-237 than most other approaches including the better performing KS and PredPath. AdamicAdar, Jaccard and Degree product are faster than ESTHER because they only compare the direct neighbors of $s$ and $o$, i.e., they only take paths of length 2 into account. On WN18RR, the higher $k$ and $N$ values of the best performing ESTHER configuration lead to a higher runtime than most of the other approaches.

The result of the third experiment clearly show that the solution space explored by ESTHER is complementary to that explored by solutions based on discrete data. This claim is supported by the significant increase of performance for all approaches when they are combined with ESTHER.[9] Hence, a combination of approaches that are based on a discrete representation of a knowledge graph with an approach that relies on a continuous representation clearly leads to better fact checking results.

## 7 Conclusion

The goal of this paper was to measure whether the combination of information contained in continuous and discrete representations of knowledge graphs can improve

[9] We use a Wilcoxon signed rank test with $\alpha = 0.01$

state-of-the-art methods for fact checking. We presented ESTHER, the first path-based fact checking approach that makes use of a continuous graph representation by using knowledge graph embeddings. Our results suggest that ESTHER is complementary to existing approaches on the fact checking problem. In particular, ESTHER improves the performance of all other fact checking approaches if they are combined with ESTHER using a decision tree. Natural continuations of our work include using ensemble learning to combine the 11 approaches considered in this paper. Corresponding experiments will be carried out in future works. In addition, we plan to run ESTHER on larger knowledge graphs with more complex ontologies, e.g., DBpedia. However, this step depends on the development of scalable knowledge graph embedding algorithms to generate the compositional embeddings.

## Acknowledgements

This work has been supported by the German Federal Ministry of Education and Research (BMBF) within the EuroStars project E!113314 FROCKG under the grant no 01QE19418. This work has been supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 860801.

## References

1. Adamic, L.A., Adar, E.: Friends and neighbors on the web. Social Networks 25(3) (2003)
2. 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 2018. pp. 143–146 (2018)
3. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: The semantic web, pp. 722–735. Springer (2007)
4. 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)
5. Chah, N.: OK google, what is your ontology? or: Exploring freebase classification to understand google's knowledge graph. CoRR abs/1805.03885 (2018)
6. 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)
7. Demir, C., Ngonga Ngomo, A.C.: Convolutional complex knowledge graph embeddings. Proceedings of the Extended Semantic Web Conference (2020)
8. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. CoRR abs/1707.01476 (2017)
9. Gerber, D., Esteves, D., Lehmann, J., Bühmann, L., Usbeck, R., Ngonga Ngomo, A.C., Speck, R.: DeFacto—temporal and multilingual deep fact validation. Web Semantics: Science, Services and Agents on the World Wide Web 35, 85–101 (2015)
10. Jeh, G., Widom, J.: Simrank: A measure of structural-context similarity. In: Proceedings of the Eighth ACM SIGKDD Intern. Conf. on Knowledge Discovery and Data Mining (2002)
11. Ji, S., Pan, S., Cambria, E., Marttinen, P., Yu, P.S.: A survey on knowledge graphs: Representation, acquisition, and applications. IEEE Transactions on Neural Networks and Learning Systems pp. 1–21 (2021)

12. Katz, L.: A new status index derived from sociometric analysis. Psychometrika 18 (1953)
13. Lao, N., Cohen, W.W.: Relational retrieval using a combination of path-constrained random walks. Machine learning 81(1), 53–67 (2010)
14. Li, F., Dong, X.L., Langen, A., Li, Y.: Knowledge verification for long-tail verticals. Proc. VLDB Endow. 10(11), 1370–1381 (Aug 2017)
15. Liben-Nowell, D., Kleinberg, J.: The link prediction problem for social networks. In: Proceedings of the Twelfth Intern. Conf. on Information and Knowledge Management (2003)
16. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Twenty-ninth AAAI conf. on artificial intelligence (2015)
17. Lu, H., Hu, H.: Dense: An enhanced non-abelian group representation for knowledge graph embedding (2020)
18. 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: International Semantic Web Conference. pp. 376–394. Springer (2018)
19. 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 (2012)
20. Shi, B., Weninger, T.: Fact checking in large knowledge graphs - A discriminative predicate path mining approach. CoRR abs/1510.05911 (2015)
21. Shi, B., Weninger, T.: Discriminative predicate path mining for fact checking in knowledge graphs. Knowledge-based systems 104, 123–133 (2016)
22. 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)
23. Singhal, A.: Introducing the knowledge graph: things, not strings. Official google blog (May 2012), https://www.blog.google/products/search/introducing-knowledge-graph-things-not/
24. 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 (2013)
25. Sun, Y., Han, J., Yan, X., Yu, P.S., Wu, T.: Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. Proceedings of the VLDB Endowment 4(11) (2011)
26. Sun, Z., Deng, Z., Nie, J., Tang, J.: Rotate: Knowledge graph embedding by relational rotation in complex space. CoRR abs/1902.10197 (2019)
27. 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)
28. Syed, Z.H., Röder, M., Ngonga Ngomo, A.C.: Unsupervised discovery of corroborative paths for fact validation. In: The Semantic Web – ISWC 2019. pp. 630–646 (2019)
29. Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., Gamon, M.: Representing text for joint embedding of text and knowledge bases. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 1499–1509 (Sep 2015)
30. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering 29(12) (2017)
31. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Twenty-Eighth AAAI conference on artificial intelligence (2014)
32. Xu, Z., Pu, C., Yang, J.: Link prediction based on path entropy. Physica A: Statistical Mechanics and its Applications 456, 294–301 (2016)
33. Zhao, M., Chow, T.W., Zhang, Z., Li, B.: Automatic image annotation via compact graph based semi-supervised learning. Knowledge-Based Systems 76, 148–165 (2015)