Engineering Applications of Artificial Intelligence 103 (2021) 104302

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

CAFE: Knowledge graph completion using neighborhood-aware features

Agustín Borrego [a,*], Daniel Ayala [a], Inma Hernández [a], Carlos R. Rivero [b], David Ruiz [a]

[a] University of Seville, E.T.S. Ingeniería Informática, Avda. de la Reina Mercedes s/n, Seville, Spain
[b] Rochester Institute of Technology, 92 Lomb Memorial Drive, Rochester, NY, USA

ARTICLE INFO

Keywords:
Knowledge Graphs
Knowledge graph completion
Link prediction
Machine learning

ABSTRACT

Knowledge Graphs (KGs) currently contain a vast amount of structured information in the form of entities and relations. Because KGs are often constructed automatically by means of information extraction processes, they may miss information that was either not present in the original source or not successfully extracted. As a result, KGs might lack useful and valuable information. Current approaches that aim to complete missing information in KGs have two main drawbacks. First, some have a dependence on embedded representations, which impose a very expensive preprocessing step and need to be recomputed again as the KG grows. Second, others are based on long random paths that may not cover all relevant information, whereas exhaustively analyzing all possible paths between entities is very time-consuming. In this paper, we present an approach to complete KGs based on evaluating candidate triples using a set of neighborhood-based features. Our approach exploits the highly connected nature of KGs by analyzing the entities and relations surrounding any given pair of entities, while avoiding full recomputations as new entities are added. Our results indicate that our proposal is able to identify correct triples with a higher effectiveness than other state-of-the-art approaches, achieving higher average F1 scores in all tested datasets. Therefore, we conclude that the information present in the vicinities of the two entities within a candidate triple can be leveraged to determine whether that triple is missing from the KG or not.

1. Introduction

Knowledge Graphs (KGs) are vast repositories of structured information that have gained increasing popularity over the past years (Hogan et al., 2020). Large volumes of information about different domains can be found in some well-known KGs, such as DBpedia (Lehmann et al., 2015), NELL (Mitchell et al., 2018), Freebase (Bollacker et al., 2008) or the Google Knowledge Vault (Dong et al., 2014), and can be used for tasks like question answering (Bordes et al., 2014a; Huang et al., 2019).

KGs are most commonly built by extracting non-structured (Dong et al., 2014; Mitchell et al., 2018) or semi-structured (Glass and Gliozzo, 2018; Lehmann et al., 2015) information from web sources, though some smaller KGs can be manually curated by domain experts (Miller, 1995). When information extraction systems are applied to extract knowledge from online sources, that information is then semantized (Ayala et al., 2019b; Neumaier et al., 2016) and stored in a KG as triples (Schlegel and Freitas, 2019), which represent facts in the form of two entities connected by means of a certain relation. Regardless of the specific process by which a KG is constructed, the resulting structure usually lacks a certain amount of information, either because said information was not originally present in the information source, or because it was incorrectly extracted or semantized (Bordes and Gabrilovich, 2014). Because of this inherent incompleteness, KGs operate under the Open World Assumption, i.e., a piece of information that is not present in a KG is not considered to be incorrect, but rather just unknown (Galárraga et al., 2015). Therefore, it is mandatory to refine KGs after their creation in order to expand the knowledge they contain and to increase the accuracy of their information (Paulheim, 2017). Such is the example of the Google Knowledge Vault (Dong et al., 2014), a KG that is commonly used for question answering, which is under continuous expansion and refinement.

Deriving additional knowledge from an existing KG to augment it is a task known as Knowledge Graph completion (Paulheim, 2017). In KG completion, the goal is to identify triples that are missing from the KG and have a reasonable chance of being correct. This is usually done by creating, training and applying a prediction model in a task known as triple classification (Borrego et al., 2019; Lin et al., 2018). Clearly, it is desirable for these models to achieve a high precision to ensure that the triples that are considered to be missing from the KG represent correct facts in the real world (Nentwig et al., 2017).

KG completion can be approached with two complementary goals in mind: completing information about entity types (Neelakantan and

[*] Corresponding author.
E-mail addresses: borrego@us.es (A. Borrego), dayala1@us.es (D. Ayala), inmahernandez@us.es (I. Hernández), crr@cs.rit.edu (C.R. Rivero), druiz@us.es (D. Ruiz).

https://doi.org/10.1016/j.engappai.2021.104302
Received 30 July 2020; Received in revised form 23 April 2021; Accepted 11 May 2021
Available online 18 May 2021

Chang, 2015), or completing the relations between entities (Galárraga et al., 2013; Lao and Cohen, 2010; Lin et al., 2015; Socher et al., 2013). Furthermore, KG completion techniques can use both external features, i.e., features that depend on external information sources; or internal features, which are computed using only the KG itself (Drumond et al., 2012; Ji et al., 2015).

As a motivational example, Fig. 1 presents a KG that contains information about fictional works, actors, writers and characters. This KG is incomplete, as it is missing certain information from the real world. An example of completing relations between entities would be relating *Daniel Radcliffe* and *Harry Potter* by means of the relation *plays*, which represents a correct fact that is not included in this KG.

In this work, we focus in KG completion using only internal features, since this line of research has shown promising results while avoiding dependencies to external sources of information (Paulheim, 2017). There are several related proposals in the literature that use solely internal features (Galárraga et al., 2015; Gardner and Mitchell, 2015; Ji et al., 2015; Lin et al., 2015; Socher et al., 2013; Wang et al., 2014). Unfortunately, they suffer from a number of drawbacks. On one hand, some of them are embedding-based, which imposes a very expensive pre-processing step to map the entities and relations present in the KG to the embedding space. On the other hand, other proposals are based on random walks that are non-deterministic by definition and, as a consequence, they may not cover relevant information in the vicinity of an entity.

In this paper, we present CAFE, our approach for KG Completion using neighborhood-aware features, which uses a feature set that leverages information from the neighborhoods of the entities, i.e., other nearby entities and relations. This feature set transforms triples in the KG into feature vectors, which are then used to train neural prediction models. These models help to discern between correct triples that should be added to the KG, and incorrect ones that should be disregarded.

The main contributions of our work are as follows:

- **A set of neighborhood-based features.** We propose a set of features that can be applied directly to any KG, without the need for computing embedded representations. This, in addition to the fact that the proposed features do not require any pre-processing of the KG, means that KG completion can be performed in a more agile way. This is especially appealing for ever-growing KGs, since full recomputations are not needed.
- **Deterministic KG completion.** Because the proposed features do not rely on random paths, all the contextual information surrounding any given entity is always taken into account. This is especially useful in dense KGs, in which relevant information may not be covered by random paths due to mere randomness.
- **A KG completion workflow.** We propose a KG completion technique, CAFE, that leverages the previously discussed set of features to perform KG completion with a minimal amount of user intervention.
- **High effectiveness.** Our experimental results show that CAFE is able to predict which relations should be added to a KG with a comparable or higher effectiveness than other state-of-the-art approaches.

The rest of this paper is organized as follows: Section 2 analyzes the related work in KG completion proposals, Section 3 describes our terminology and the set of features that CAFE uses, Section 4 presents our proposal for training classification models for triples in a KG, Section 5 reports on our experimental results; finally, Section 6 presents our conclusions.

2. Related work

In the field of KG completion, the existing techniques can be classified into three types: rule-based, embedding-based and path-based proposals.

Rule-based proposals rely on discovering logical rules that determine whether a given triple is correct. In this regard, some authors propose using Inductive Logic Programming to find Horn rules that express the relations between entities in a KG, and then applying these rules to produce new explicit knowledge (Galárraga et al., 2013; Kolthoff and Dutta, 2015), under the assumption that all triples that match the rules are correct. The performance of the obtained rules is usually measured using metrics such as rule support or confidence, or a variant of these metrics (Galárraga et al., 2015).

Proposals that are based on embeddings aim to evaluate possible relations in a KG by learning embedded representations of its entities and relations, either by using them as inputs for a binary classification model (Socher et al., 2013), or by performing different transformations in an embedding space (Kazemi and Poole, 2018; Lin et al., 2015; Liu et al., 2017; Sun et al., 2019; Trouillon et al., 2016). The resulting embedding space – or spaces – is subsequently used to evaluate the likelihood of a candidate triple to be correct or incorrect, since entities that are supposed to be related by means of a certain relation are expected to be close to each other in the embedding space. A similar line of work includes representing a KG as a tensor, and then factorizing it to obtain latent, more compact representations of the triples contained within it (Drumond et al., 2012; Nickel et al., 2012). These proposals suffer from a performance drawback: due to the way in which the embedded representations are obtained, they need to be recomputed whenever new triples are added to the KG, which is a relatively frequent event (Dong et al., 2014; Mitchell et al., 2018). Generally, embedding-based proposals rank triples in order of decreasing likelihood, according to how close the two entities in the triples are in the generated space. Thus, they are commonly evaluated using ranking-based metrics, such as MRR or Hits@N (Wang et al., 2019); however, a likelihood threshold can be set to compute traditional classification measures such as precision, recall, or F1 (Han et al., 2018).

Finally, path-based techniques exploit the highly relational nature of KGs to learn how to predict new relations between entities. Our approach, CAFE, belongs to this category. In this line of work, Lao and Cohen (2010) introduced the Path Ranking Algorithm (PRA), a two-step process to find which paths may be useful to predict a certain relation. An evolution of PRA named Subgraph Feature Extraction (SFE) is proposed by Gardner and Mitchell (2015). SFE achieves better performance than PRA and produces more expressive results. It also requires the creation of a handmade "Alias" relation, which relates entities in the same KG that refer to the same element in the real world. Mazumder and Liu (2017) propose a random walk-based approach using neighborhood-guided path finding, where semantic similarities between entities are computed by applying a Word2vec-based embedding model on the names of the entities. Reinforcement learning has also been used to find valuable paths that can help to successfully complete a KG (Xiong et al., 2017). Unfortunately, due to the non-deterministic way in which these paths are computed, they may miss relevant information by mere chance.

As of recent, there also exist a number of KG completion proposals that combine the previous approaches. For example, Shen et al. (2019) propose computing embeddings of the entities and relations, and then combining these embeddings in the forms of paths. This can be helpful for determining whether a certain triple is correct. A different way of obtaining embedded representations of the elements in a KG is presented by Wang et al. (2019), by using graph neural networks to capture information pertaining to the structure of the graph. Entity embeddings have also been used to aid in the production of KG completion rules (Ho et al., 2018). While embedded representations can undoubtedly be helpful, CAFE avoids computing them for the sake of agility of application, and instead relies only on analyzing the structure of the KG as is.

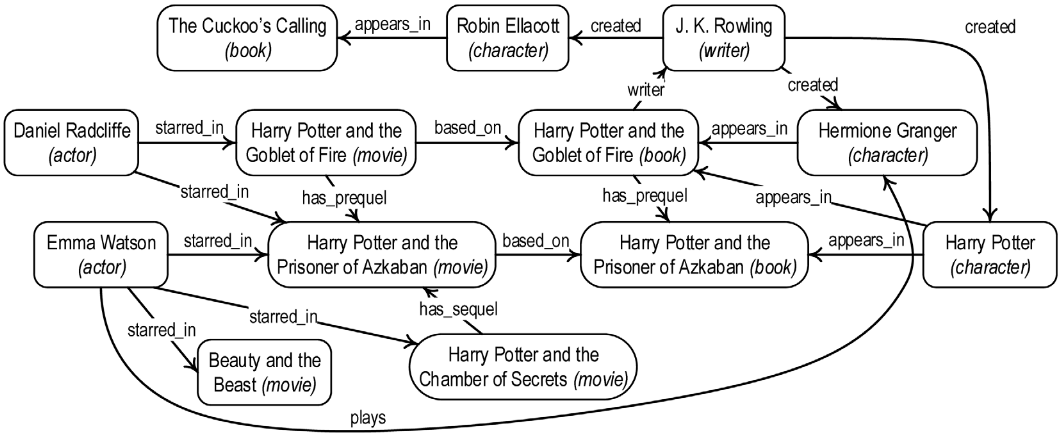

The Cuckoo's Calling (book)
appears_in
Robin Ellacott (character)
created
J. K. Rowling (writer)
created
writer
created
Daniel Radcliffe (actor)
starred_in
Harry Potter and the Goblet of Fire (movie)
based_on
Harry Potter and the Goblet of Fire (book)
appears_in
Hermione Granger (character)
starred_in
has_prequel
has_prequel
appears_in
Emma Watson (actor)
starred_in
Harry Potter and the Prisoner of Azkaban (movie)
based_on
Harry Potter and the Prisoner of Azkaban (book)
appears_in
Harry Potter (character)
starred_in
has_sequel
starred_in
Beauty and the Beast (movie)
Harry Potter and the Chamber of Secrets (movie)
plays

Fig. 1. An example of a KG describing works, actors, writers and characters.

3. Neighborhood-aware features

In this section, we first introduce some preliminary concepts that are necessary to understand our proposal, such as reachability and neighborhood subgraphs. We then define our set of neighborhood-aware features.

3.1. Preliminaries

The following definitions form the core of our proposal.

Definition 1 (*Triple*)**.** Let $\mathcal{E}$ be a set of entities, and let $\mathcal{R}$ be a set of relations. We define a triple as a 3-tuple that represents the existence of a relation $r \in \mathcal{R}$ between a source entity $s \in \mathcal{E}$ and a target entity $t \in \mathcal{E}$. We denote triples as (s, r, t).

In the sample KG depicted in Fig. 1, a sample triple is (*Emma Watson, starred_in, Beauty and the Beast*).

Definition 2 (*Knowledge Graph*)**.** Let $\mathcal{E}$ be a set of entities, let $\mathcal{R}$ be a set of relations, and let $\mathcal{T}$ be a set of triples of the form $\{(s, r, t) \mid s, t \in \mathcal{E}, r \in \mathcal{R}\}$. We define a Knowledge Graph as $\mathcal{KG} = (\mathcal{E},\ \mathcal{R},\ \mathcal{T})$.

Fig. 1 graphically represents a KG with 13 entities, 7 distinct relations and 19 triples.

Definition 3 (*Path Between Entities*)**.** Let $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph, and let $s, t \in \mathcal{E}$ be two entities in $\mathcal{KG}$. We define a path p between s and t as a sequence of triples of the form $p = \langle (e_i, r_i, e_{i+1}) \rangle$ for $i = 1..n$, where $e_1 = s$, $e_{n+1} = t$ and $(e_i, r_i, e_{i+1}) \in \mathcal{T}$ for $i = 1..n$. We define the length of a path as the number of triples it contains, i.e., $|p|$. We denote a path p of length n between s and t using the relations $r_1 \dots r_n$ as $path(s, t, r_1, r_2, \dots, r_n)$, or $path_n(s, t)$ for short. We denote the set of all possible distinct paths of the form $path(s, t, r_1, r_2, \dots, r_n)$ as $\mathcal{P}(s, t, r_1, r_2, \dots, r_n)$.

In the KG depicted in Fig. 1, an example of a path of length 2 between the entities *J.K. Rowling* and *The Cuckoo's Calling* is $\langle$(*J.K. Rowling, created, Robin Ellacott*), (*Robin Ellacott, appears_in, The Cuckoo's Calling*)$\rangle$.

Definition 4 (*Reachability*)**.** Let $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph, let $s, t \in \mathcal{E}$ be two entities in $\mathcal{KG}$, let $r \in \mathcal{R}$ be a relation in $\mathcal{KG}$, and let $n \geq 1$ be a natural number. We define reachability as a predicate that determines whether there exists a path of length n between s and t in $\mathcal{KG}$ such that the relation r appears in the last triple of the path, i.e., $Reach(\mathcal{KG},\ s,\ t,\ r,\ n) \iff \exists\ path_n(s, t) \wedge \exists\ a \in \mathcal{E} \mid last(path_n(s, t)) = (a, r, t)$. We define the set of entities that can be reached from s through a relation r at distance n as the set of entities that match the predicate Reach under such circumstances, i.e., $\{t \in \mathcal{E} \mid Reach(\mathcal{KG}, \text{s, t, r, n})\}$. We denote the previously defined set as $Reachable(s, r, n)$.

In the example KG depicted in Fig. 1,
$Reachable$(*Hermione Granger, writer, 2*) =
{*J.K. Rowling*}.

Definition 5 (*Neighborhood Subgraph*)**.** Let $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph, let $e \in \mathcal{E}$ be an entity in $\mathcal{KG}$, and let $n \geq 1$ be a natural number. We define the neighborhood subgraph of e of size n as a Knowledge Graph $\mathcal{KG}^n_e = (\mathcal{E}^n_e,\ \mathcal{R},\ \mathcal{T}^n_e)$ that contains the triples whose target entities can be reached from e at a distance of at most n through any relation, and the entity set that can be derived from such triples, where $\mathcal{T}^n_e = \{(s', r', t') \in \mathcal{T} \mid Reach(\mathcal{KG},\ \text{e},\ t',\ r',\ \text{i}),\ \text{i} = 1..\text{n}\}$ and $\mathcal{E}^n_e = \bigcup\{\{s, t\} \subseteq \mathcal{E} \mid (s, r, t) \in \mathcal{T}^n_e\}$.

Fig. 2 illustrates this definition, with two possible neighborhood subgraphs for the KG shown in Fig. 1.

3.2. Feature set

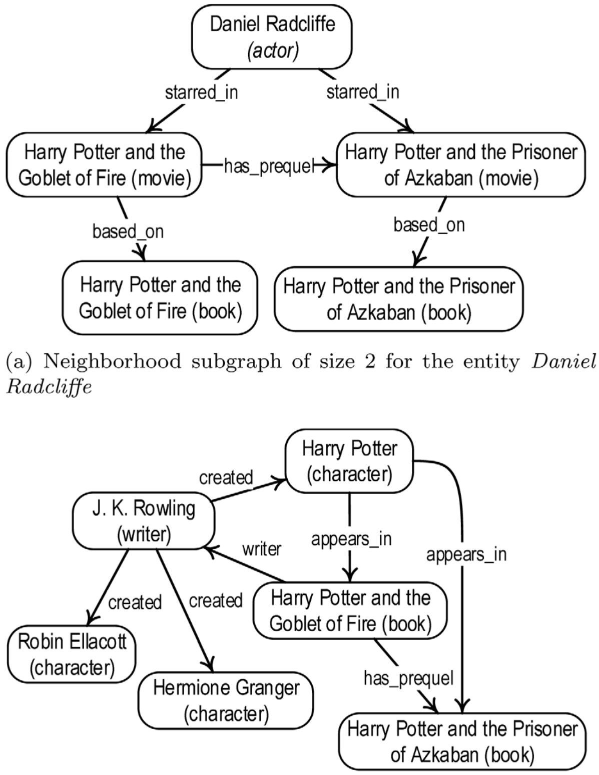
We propose a set of neighborhood-aware features that takes neighborhood subgraphs, reachable entities and paths into account. Due to the large number of possible variations of each feature, we present our feature set in terms of groups of features. Each group can be parameterized to obtain a specific feature, which we call an instance of the feature group.

Definition 6 (*Feature*)**.** Let $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph. We define a feature f as a function $f : \mathcal{T} \to \mathbb{R}$ that assigns a real number to a triple.

For example, a feature f may convert a triple into the number of entities in the neighborhood subgraph of size 2 of the source entity, i.e., $f : (s, r, t) \mapsto |\mathcal{E}^2_s|$.

Definition 7 (*Feature Group*)**.** Let $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph. We define a feature group f_n as a function $f_n : \mathcal{X} \to (\mathcal{T} \to \mathbb{R})$ that receives a set of parameters $\mathcal{X}$ and returns a feature.

For example, a feature group f_0 may return a feature that converts a triple into the number of entities in the neighborhood subgraph of size n of the source entity, i.e., $f_0(n) = f : (s, r, t) \mapsto |\mathcal{E}^n_s|$, where n is a parameter of the feature group. Thus, $f_0(2) : (s, r, t) \mapsto |\mathcal{E}^2_s|$, which is the feature shown in the previous example. Consequently, feature groups allow us to represent a set of very similar features in a more compact way, where the only distinction between said features is a given set of parameters.


Daniel Radcliffe
(actor)
starred_in
starred_in
Harry Potter and the Goblet of Fire (movie)
has_prequel
Harry Potter and the Prisoner of Azkaban (movie)
based_on
based_on
Harry Potter and the Goblet of Fire (book)
Harry Potter and the Prisoner of Azkaban (book)
Harry Potter (character)
created
J. K. Rowling (writer)
writer
appears_in
appears_in
created
created
Robin Ellacott (character)
Hermione Granger (character)
Harry Potter and the Goblet of Fire (book)
has_prequel
Harry Potter and the Prisoner of Azkaban (book)

(a) Neighborhood subgraph of size 2 for the entity *Daniel Radcliffe*

(b) Neighborhood subgraph of size 3 for the entity *Harry Potter*

Fig. 2. Two neighborhood subgraphs for the sample KG depicted in Fig. 1.

In the following, we present the feature groups that CAFE uses. For the sake of clarity, we illustrate a possible instance of every feature group and its value using the example triple $example$ = (*Daniel Radcliffe, plays, Harry Potter*), the KG shown in Fig. 1 and the neighborhood subgraphs described in Fig. 2.

Feature group f_1: Number of entities in the neighborhood subgraph of size n of the source entity in the triple.

Features in this group are computed as:

$$f_1(n) : (s,r,t) \mapsto |\mathcal{E}^n_s|$$

In the example shown in Fig. Fig. 2(a), $f_1(2)$ applied to the *example* triple is $|\{$*Daniel Radcliffe, Harry Potter and the Goblet of Fire (movie), Harry Potter and the Prisoner of Azkaban (movie), Harry Potter and the Goblet of Fire (book), Harry Potter and the Prisoner of Azkaban (book)*$\}|$ = 5.

Feature group f_2: Number of entities in the neighborhood subgraph of size n of the target entity in the triple.

Features in this group are computed as:

$$f_2(n) : (s,r,t) \mapsto |\mathcal{E}^n_t|$$

In the example shown in Fig. Fig. 2(b), $f_2(3)$ applied to the *example* triple is $|\{$*Harry Potter, J.K. Rowling, Harry Potter and the Goblet of Fire (book), Harry Potter and the Prisoner of Azkaban (book), Robin Ellacott, Hermione Granger*$\}|$ = 6.

Feature group f_3: Degree of n-path centrality of the source entity in the triple.

Features in this group are computed as:

$$f_3(n) : (s,r,t) \mapsto \frac{|\mathcal{E}^n_s|}{|\mathcal{E}| - 1}$$

In the example shown in Fig. Fig. 2(a), $f_3(1)$ applied to the *example* triple is $|\{$*Harry Potter and the Goblet of Fire (movie), Harry Potter and the Prisoner of Azkaban (movie)*$\}|/(13-1) = 2/12 \approx 0.17$.

Feature group f_4: Degree of N-path centrality of the target entity in the triple.

Features in this group are computed as:

$$f_4(n) : (s,r,t) \mapsto \frac{|\mathcal{E}^n_t|}{|\mathcal{E}| - 1}$$

In the example shown in Fig. Fig. 2(b), $f_4(1)$ applied to the *example* triple is $|\{$*Harry Potter and the Goblet of Fire (book), Harry Potter and the Prisoner of Azkaban (book)*$\}|/(13-1) = 2/12 \approx 0.17$.

Feature group f_5: Number of common entities between the neighborhood subgraph of size n of the source entity and the neighborhood subgraph of size m of the target entity in the triple.

Features in this group are computed as:

$$f_5(n,m) : (s,r,t) \mapsto |\mathcal{E}^n_s \cap \mathcal{E}^m_t|$$

In the example shown in Fig. Fig. 2, $f_5(2,3)$ applied to the *example* triple is $|\{$*Harry Potter and the Goblet of Fire (book), Harry Potter and the Prisoner of Azkaban (book)*$\}|$ = 2.

Feature group f_6: Jaccard index of similarity between the entities in the neighborhood subgraph of size n of the source entity and the neighborhood subgraph of size m of the target entity in the triple.

Features in this group are computed as:

$$f_6(n,m) : (s,r,t) \mapsto jaccard(\mathcal{E}^n_s, \mathcal{E}^m_t)$$

In the example shown in Fig. Fig. 2, $f_6(2,3)$ applied to the *example* triple is $|\{$*Harry Potter and the Goblet of Fire (book), Harry Potter and the Prisoner of Azkaban (book)*$\}|$ / $|\{$*Daniel Radcliffe, Harry Potter and the Goblet of Fire (movie), Harry Potter and the Prisoner of Azkaban (movie), Harry Potter and the Goblet of Fire (book), Harry Potter and the Prisoner of Azkaban (book), Harry Potter, J.K. Rowling, Robin Ellacott, Hermione Granger*$\}|$ = 2 / 9 = 0.22.

Feature group f_7: Adamic–adar index of closeness between the neighborhood subgraphs of size n of the source and target entities in the triple.

Features in this group are computed as:

$$f_7(n) : (s,r,t) \mapsto \sum_{e \in \mathcal{E}^n_s \cap \mathcal{E}^n_t} \frac{1}{log|\mathcal{E}^n_e|}$$

In the example shown in Fig. Fig. 2, $f_7(2)$ applied to the *example* triple is $\frac{1}{log|\mathcal{E}^2_{HarryPotterandtheGobletofFire(book)}|} = \frac{1}{log|3|} \approx 2.09$.

Feature group f_8: Number of reachable entities through the relation r at distance n from the source entity in the triple.

Features in this group are computed as:

$$f_8(r,n) : (s,r,t) \mapsto |Reachable(s,r,n)|$$

In the example shown in Fig. Fig. 2(a), f_8(*hasPrequel*, 2) applied to the *example* triple is $|\{$*Daniel Radcliffe, Harry Potter and the Prisoner of Azkaban (movie)*$\}|$ = 2.

Feature group f_9: Number of reachable entities through the relation r at distance n from the target entity in the triple.

Features in this group are computed as:

$$f_9(r,n) : (s,r,t) \mapsto |Reachable(t,r,n)|$$

In the example shown in Fig. Fig. 2(b), f_9(*created*, 2) applied to the *example* triple is $|\{$*Harry Potter, Hermione Granger, Robin Ellacott*$\}|$ = 3.

Feature group f_{10}: Number of common reachable entities through the relation r from the source entity at distance n and from the target entity at distance m.

Features in this group are computed as:

$$f_{10}(r,n,m) : (s,r,t) \mapsto |Reachable(s,r,n) \cap Reachable(t,r,m)|$$

In the example shown in Fig. Fig. 2, f_{10}(*created*, 2, 3) applied to the *example* triple is |{*Daniel Radcliffe*} ∩ {*Harry Potter, Hermione Granger, Robin Ellacott*}| = 0.

Feature group f_{11}: Jaccard index of similarity between the reachable entities through the relation r from the source entity at distance n, and those reachable through the relation r from the target entity at distance m.

Features in this group are computed as:

$$f_{11}(r,n,m) : (s,r,t) \mapsto jaccard(Reachable(s,r,n), Reachable(t,r,m))$$

In the example shown in Fig. Fig. 2, f_{11}(*created*, 2, 3) applied to the *example* triple is |∅| / |{*Harry Potter, Hermione Granger, Robin Ellacott*}| = 0 / 3 = 0.

Feature group f_{12}: Number of distinct paths of length n between the source and the target entity in the triple, using relations $r_1, \dots, r_n$.

Features in this group are computed as:

$$f_{12}(n, r_1, \dots, r_n) : (s,r,t) \mapsto |\mathcal{P}(s,t,r_1,\dots,r_n)|$$

In the example shown in Fig. Fig. 1, f_{12}(4, *starred_in, based_on, writer, created*) applied to the *example* triple is 1, as there is one path of length 4 between the entities *Daniel Radcliffe* and *Harry Potter* that matches the given relations.

3.3. Discussion of the feature set

The rationale behind this set of features is manifold. Regarding f_1 and f_2, knowing the size of the neighborhood of an entity can be helpful to determine whether said neighborhood encompasses relevant information. For instance, our hypothesis is that very large neighborhoods tend to contain a higher amount of unrelated information. The same idea is leveraged in f_3 and f_4, which provide normalized indices of centrality with respect to the total amount of entities in the KG. Following the previous reasoning, we hypothesize that entities with large indices of centrality (i.e. highly connected to other entities) will yield less useful information. Meanwhile, feature groups f_5, f_6 and f_7 measure the degree of overlap that exists in the neighborhoods of the two entities, both in absolute and in relative terms, under the assumption that correct triples have a higher degree of overlap between the neighborhoods of their entities.

The previously discussed feature groups do not take into account the specific relations involved. However, we deem it reasonable to assume that some relations may be more useful than others to determine whether a triple is correct, depending on their specific semantics. For example, having one or more children in common can be an indication of a marriage, while having the same nationality is not. In order to exploit this fine-grained information, feature groups f_8, f_9, f_{10} and f_{11} are similar to the previously discussed groups but restrict themselves to only one relation. These groups are computed for every relation in a KG.

Finally, feature group f_{12} allows CAFE to find the number of paths that exist between two entities for any given relations. It is our intuition that correct triples have more alternative paths between the two entities they contain than false ones.

4. Our proposal

Our proposal, CAFE, receives a KG and a set of relations from that KG as input, and outputs a classification model for each of the provided relations. These models are able to determine if a given triple that represents an instance of the relation is correct and should belong to the KG. Our workflow is depicted in Fig. 3 and, in the following subsections, we describe each of its steps.

4.1. Loading the KG

CAFE internally stores the input KG in the form of (s,r,t) triples, using an efficient data structure based on hashable properties, which is suitable for a high frequency of read operations. The triples that contain relations for which a predictive model does not need to be generated are still taken into account when computing features, since they may provide valuable predictive information, but they are not transformed into feature vectors in the following steps.

4.2. Generating negative examples

A KG contains only positive information, i.e., it contains examples of the occurrence of a relation r between two entities. However, it does not contain explicit information about pairs of entities for which r does not hold. Our proposal relies on a classification model that requires negative examples for training, which means that a number of negatives for each positive triple must be produced. To accomplish this, we follow the type-constrained local closed world assumption (Bansal et al., 2020), i.e., we generate negative examples from every triple (s,r,t) present in a KG by replacing their target entity t with a different one, t', such that the resulting triple (s,r,t') does not exist in the KG. Furthermore, to preserve the range of each relation, we randomly choose t' such that there exists some other triple in the KG where t' appears as the target entity for the relation r. This is known as type constraint.

In the example depicted in Fig. 1, a valid negative example is *(Hermione Granger, appears_in, The Cuckoo's Calling)*, since we know that *The Cuckoo's Calling* is a valid target for the relation *appears_in*. However, *(Hermione Granger, appears_in, Daniel Radcliffe)* would not be allowed as a negative example, because *Daniel Radcliffe* never appears as the target of the relation $appears\_in$.

It can be argued that generating negative evidence in this manner can produce false negatives by mere chance, i.e., statements that are deemed incorrect but that are true in the real world. While this is indeed plausible we, as other authors (Ji et al., 2015; Lin et al., 2015; Socher et al., 2013), consider that the chances of this happening are low and, as a consequence, the possible effects on the final results are not significant.

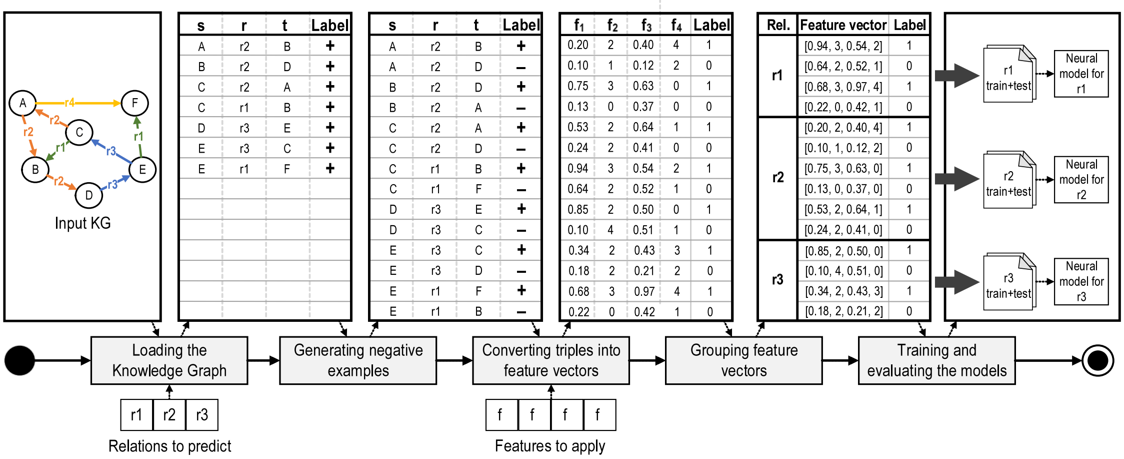
4.3. Converting triples into feature vectors

Once negative examples have been generated, our feature set is instantiated and applied to all triples. For all feature groups, we obtain all possible feature instances by applying all possible combinations of the values of their parameters. Each feature instance assigns a real number to each triple. Therefore, applying several features to a triple results in a feature vector. Each position of the feature vector represents that real number that the corresponding feature assigned to the triple.

It is important to note that, to compute features on a positive training triple, we temporarily remove it from the KG, since not doing so would result in trivial prediction models such as "a person plays a character if there exists a triple in the KG stating that the person plays that character".

4.4. Grouping feature vectors

The previous step computes feature vectors of triples. Since these triples can be either positive or negative, the feature vectors are labeled as positive or negative. Based of the labeled feature vectors, we train a classification model for each relation that predicts whether a triple should be added to the KG. We do this in order to allow the models to capture meaningful and distinctive information for every relation: even though the same set of features is applied to all triples, some features might have more predictive power for a relation, and other features may be more helpful for a different one.



s	r	t	Label
A	r2	B	+
B	r2	D	+
C	r2	A	+
C	r1	B	+
D	r3	E	+
E	r3	C	+
E	r1	F	+

s	r	t	Label
A	r2	B	+
A	r2	D	–
B	r2	D	+
B	r2	A	–
C	r2	A	+
C	r2	D	–
C	r1	B	+
C	r1	F	–
D	r3	E	+
D	r3	C	–
E	r3	C	+
E	r3	D	–
E	r1	F	+
E	r1	B	–

f_1	f_2	f_3	f_4	Label
0.20	2	0.40	4	1
0.10	1	0.12	2	0
0.75	3	0.63	0	1
0.13	0	0.37	0	0
0.53	2	0.64	1	1
0.24	2	0.41	0	0
0.94	3	0.54	2	1
0.64	2	0.52	1	0
0.85	2	0.50	0	1
0.10	4	0.51	1	0
0.34	2	0.43	3	1
0.18	2	0.21	2	0
0.68	3	0.97	4	1
0.22	0	0.42	1	0

Rel.	Feature vector	Label
r1	[0.94, 3, 0.54, 2]	1
	[0.64, 2, 0.52, 1]	0
	[0.68, 3, 0.97, 4]	1
	[0.22, 0, 0.42, 1]	0
r2	[0.20, 2, 0.40, 4]	1
	[0.10, 1, 0.12, 2]	0
	[0.75, 3, 0.63, 0]	1
	[0.13, 0, 0.37, 0]	0
	[0.53, 2, 0.64, 1]	1
	[0.24, 2, 0.41, 0]	0
r3	[0.85, 2, 0.50, 0]	1
	[0.10, 4, 0.51, 0]	0
	[0.34, 2, 0.43, 3]	1
	[0.18, 2, 0.21, 2]	0

Fig. 3. Workflow of CAFE.

4.5. Training and evaluating the models

For every relation that we predict, we create one or more neural models, where each model focuses only on the features that are obtained from a certain neighborhood size. Thus, using only neighborhood subgraphs of size 1 results in one model, using neighborhood subgraphs of size of up to 2 results in two models, and so on. This allows each model to capture the specific information that every neighborhood size may yield. To combine two or more models, we use an additional combination layer to produce a single output.

The neural models are trained using the labeled feature vectors in the training split for the desired relation, where each model receives only the features corresponding to its assigned neighborhood size, and the label or ground truth is shared among them. Prior to training our models, we first remove any individual features that have the exact same value in every feature vector and thus lack any predictive power. Such a removal helps the model focus only on a subset of potentially more useful features, which has been shown to improve classification results (Karasu et al., 2020). An example of this are path-based features (f_{12}), since only a small subset of all possible paths of fixed length occur between two given entities, and as a consequence most of them have a value of 0.

Note that we use neural classification models because they have been shown to consistently achieve satisfactory results in many different classification tasks (Aggarwal and Zhai, 2012; Altan and Karasu, 2020; Yadav and Bethard, 2019), although other classification models that make use of our features could be used in this step.

5. Evaluation

In this section, we present the experimental results that we obtained after evaluating the classification effectiveness of CAFE using several datasets, and we compare it with respect to other state-of-the-art KG completion techniques. In the following, we introduce the datasets that we used for evaluation, our experimental setting and, finally, our results.

5.1. Datasets

We evaluated our proposal using four datasets provided by the freely available AYNEC-DataGen (Ayala et al., 2019a) tool: FB13-A, WN11-AR, WN18-AR and NELL-AR. These datasets are based on the well-known FB13, WN11 (Socher et al., 2013), WN18 (Bordes et al., 2014b), and a subset of NELL proposed by Gardner and Mitchell (2015). However, they have been processed to remove reciprocal relations detected by AYNEC, i.e., relations r and r' such that, if (s, r, t) exists, then (t, r', s) also exists very frequently. Additionally, relations that amount to less than 5% of the total number of triples in the graph have been removed.

Table 1
Overview of the datasets used for evaluation.

Dataset	Training triples	Test triples	Ents.	Rels.
FB13-A-10	228,172	481,457	74,998	13
WN11-AR-10	77,948	198,231	38,195	9
WN18-AR-10	71,984	183,051	40,943	11
NELL-AR-10	86,971 (1451)	219,374 (5083)	53,934	148 (9)

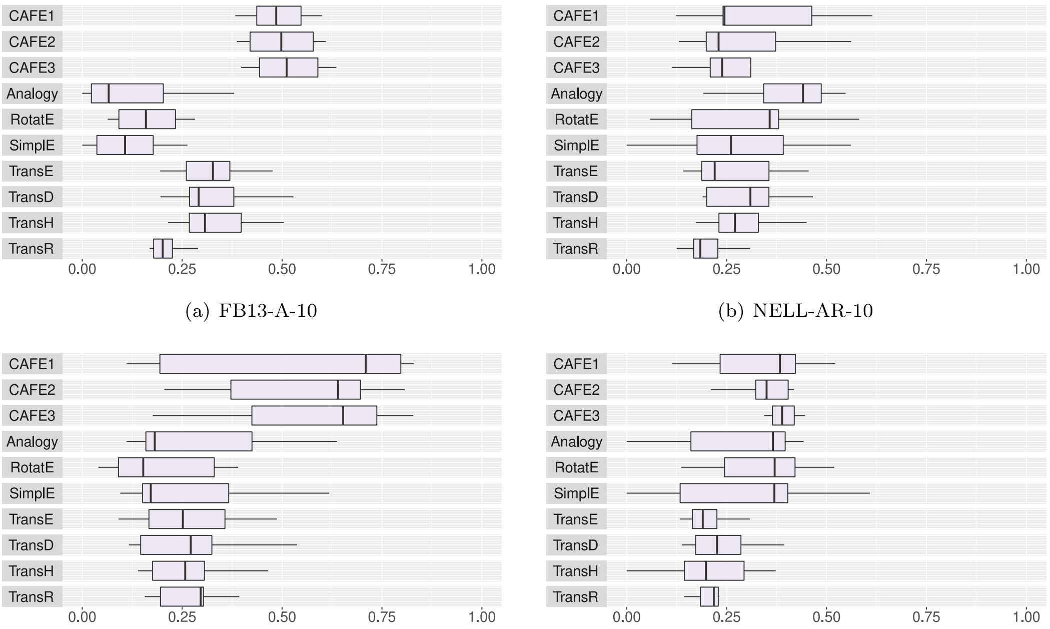
These datasets originally contained one negative example per each positive triple in both their training and testing splits. In order to study how the KG completion techniques perform when presented with a much higher volume of negative evidence, we created versions of these datasets whose testing splits contained 10 negative examples per positive, using the AYNEC-DataGen tool. We believe that this is a more realistic scenario, since a much higher number of negative examples per positive triple is typically expected in real-world KG completion tasks (Borrego et al., 2019). To avoid confusion, we denote these versions as FB13-A-10, WN11-AR-10, WN18-AR-10 and NELL-AR-10. For the sake of reproducibility, our source code and datasets are publicly available on GitHub.[1]

For FB13-A-10, WN11-AR-10 and WN18-AR-10, we aimed to predict all possible relations, and for NELL-AR-10 we focused on the same subset of 10 relations that were used to evaluate SFE (Gardner and Mitchell, 2015). However, in the latter dataset, one relation was removed by AYNEC for being the reciprocal of another relation, leaving 9 relations for evaluation. In the specific case of FB13-A-10, we transferred 25% of the training triples over to the testing set in order to provide testing examples for some relations, as they were not available in the original dataset as introduced in Socher et al. (2013). Table 1 provides an overview of the aforementioned datasets. In the case of NELL-AR-10, we show in parentheses the amount of triples and relations that were considered for evaluation, although the entire graph was used for computing features.

5.2. Evaluation method

A neural prediction model was created for every relation of interest and trained using its corresponding training set. Then, the model was

[1] https://github.com/DEAL-US/CAFE.



(a) FB13-A-10

(b) NELL-AR-10

(c) WN11-AR-10

(d) WN18-AR-10

Fig. 4. F1 scores for all relations obtained by CAFE and other state-of-the-art proposals.

applied to all feature vectors in the test set, and we compared the expected label (which denotes whether it represents a valid triple or not) against the label that was produced by our model. We report our results in terms of precision, recall and F1, in order to determine how effective our proposal is when determining the correctness of a given triple.

We evaluated three versions of CAFE, denoted CAFE$_1$ to CAFE$_3$, which were limited to using feature instances that exploited neighborhood subgraphs and paths of a maximum size of 1, 2, and 3, respectively. This was done in order to study how using larger neighborhoods affects the effectiveness of CAFE.

There exist many different KG completion proposals and, as shown in Section 2, they often use different evaluation metrics. Due to this, it is very difficult to perform a comparison across a large number of them in a manner that is fair and rigorous. For this reason, we used TransE (Bordes et al., 2013), TransD (Ji et al., 2015), TransH (Wang et al., 2014), TransR (Lin et al., 2015), Analogy (Liu et al., 2017), SimplE (Kazemi and Poole, 2018) and RotatE (Sun et al., 2019) as baselines for our evaluation, since they are some of the most well-known state-of-the-art KG completion proposals. In order to provide a common evaluation environment for these different proposals, we used the OpenKE (Han et al., 2018) framework to train and evaluate these proposals using our datasets. Additionally, since these proposals usually report metrics like MRR and Precision@N, we used the utilities provided by OpenKE to obtain binary labels for the testing triples, by setting a likelihood threshold in a way that optimized the classification results.

We selected the following values for the hyperparameters of our neural models: 3 layers with 1024, 512 and 256 neurons each, learning rate of 0.001, batch size of 16, dropout of 0.1 for all layers, 100 epochs and validation ratio of 10%. When two or more models were to be combined, we joined their results using a hidden layer with 3 neurons and an output layer with a single neuron. These values for the hyperparameters were chosen using a hold-out or “dev” set for the FB13-A-10 dataset, and all datasets were then evaluated using the same hyperparameters. We chose them because they provided satisfactory results in our empirical tests.

All our experiments were conducted on a computer equipped with an Intel Core i9-9900K CPU, 32 GB of RAM and an Nvidia RTX 2080 Ti GPU.

5.3. Results and discussion

In Fig. 4, we show the evaluation results for CAFE$_1$, CAFE$_2$, CAFE$_3$, and the related state-of-the-art proposals. For the sake of clarity, this Figure only displays the F1 values for each technique. Additionally, Table 2 shows the detailed results for all relations in every dataset and for all metrics under evaluation.

Our results show that CAFE is able to match the performance of state-of-the-art embedding-based proposals, and in many cases achieve higher values on the metrics under evaluation. In the cases of FB13-A-10 (Fig. 4(a)) and WN18-AR-10 (Fig. 4(d)), CAFE can reach or surpass the F1 scores achieved by other state-of-the-art proposals in a consistent manner. CAFE also provides better results in the WN11-AR-10 (Fig. 4(c)) dataset, although with a higher degree of variability, and matches the performance of the rest of the analyzed techniques in the NELL-AR-10 (Fig. 4(b)) dataset. These results show that CAFE can be more effective than other proposals in challenging classification scenarios.

Table 2 displays that, in general, both a satisfactory precision and recall can be achieved, and thus we consider that CAFE is generally effective. However, there exists a number of relations for which a very high precision value is obtained at the expense of a lower recall or vice-versa, resulting in a typical precision–recall trade-off. We also observe that the nature of every individual relation has a significant impact in the results, since some of them are harder to predict than others. Such is the case of the relation *cause_of_death*, since learning to predict the cause of the death of a person with a very high effectiveness would be a remarkable achievement that unfortunately falls out of the scope of this work. We further discuss these limitations in Section 5.4.

Regarding the question of how using different neighborhood sizes affects the effectiveness of CAFE, Fig. 4 shows that the metrics under evaluation are generally higher for CAFE$_2$ compared to CAFE$_1$, but the same cannot always be said for CAFE$_3$ and CAFE$_2$. Indeed, metrics appear to remain stagnant or even decrease when using larger neighborhoods in some cases. For example, in FB13-A-10 (Fig. 4(a)), we do not observe a significant increase in effectiveness when using larger neighborhood subgraphs, which suggests that the most useful information is readily available in the immediate vicinities of the

Table 2
Detailed results for all relations. The highest value for a metric in a relation is marked in bold. The highest F1 value for a relation is also highlighted in gray.

		CAFE$_1$			CAFE$_2$			CAFE$_3$			Analogy			RotatE			SimplE			TransE			TransD			TransH			TransR		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
FB13-A-10	CauseOfDeath	**0.33**	0.73	**0.45**	0.29	0.79	0.42	0.31	0.76	0.44	0.04	0.02	0.02	0.09	0.06	0.07	0.05	0.03	0.04	0.23	0.57	0.33	0.18	0.78	0.29	0.19	0.73	0.31	0.13	**0.86**	0.23
	Children	0.40	0.78	0.53	0.44	**0.88**	0.59	**0.50**	**0.88**	**0.64**	0.17	0.14	0.16	0.20	0.24	0.22	0.17	0.18	0.18	0.15	0.59	0.24	0.19	0.46	0.27	0.17	0.52	0.25	0.14	0.54	0.22
	Ethnicity	**0.60**	0.60	**0.60**	0.53	0.63	0.57	0.48	0.56	0.52	0.03	0.01	0.02	0.11	0.08	0.09	0.11	0.07	0.09	0.25	0.67	0.37	0.26	0.71	0.38	0.35	0.64	0.45	0.11	**0.94**	0.20
	Gender	0.88	0.88	0.88	**0.89**	0.89	**0.89**	**0.89**	0.89	**0.89**	0.00	0.00	0.00	0.18	0.14	0.16	0.00	0.00	0.00	0.55	**0.98**	0.70	0.71	0.81	0.76	0.82	0.88	0.85	0.74	0.84	0.79
	Institution	0.26	0.70	0.38	**0.28**	0.67	0.39	**0.28**	0.67	**0.40**	0.10	0.05	0.07	0.11	0.09	0.10	0.14	0.09	0.11	0.20	0.52	0.29	0.18	0.61	0.28	0.16	0.72	0.27	0.10	**0.95**	0.18
	Location	**0.37**	0.51	**0.43**	0.28	0.63	0.39	0.34	0.57	0.42	0.36	0.38	0.37	0.20	0.19	0.20	0.18	0.14	0.16	0.23	0.55	0.33	0.18	0.65	0.28	0.19	0.64	0.29	0.10	**0.93**	0.18
	Nationality	0.45	0.90	0.60	**0.46**	0.89	**0.61**	**0.46**	0.89	0.60	0.02	0.01	0.02	0.11	0.08	0.09	0.03	0.01	0.02	0.32	0.91	0.48	0.38	0.86	0.53	0.35	0.88	0.50	0.09	**1.00**	0.17
	Parents	0.35	0.77	0.49	0.44	**0.85**	0.58	**0.45**	**0.85**	**0.59**	0.16	0.14	0.15	0.21	0.26	0.23	0.18	0.18	0.18	0.17	0.55	0.26	0.19	0.48	0.27	0.19	0.51	0.27	0.16	0.59	0.25
	PlaceOfBirth	**0.39**	0.49	**0.44**	0.29	0.58	0.39	0.31	0.55	0.40	0.37	0.39	0.38	0.27	0.30	0.28	0.22	0.17	0.19	0.16	0.54	0.24	0.18	0.50	0.27	0.16	0.56	0.25	0.10	**0.91**	0.18
	PlaceOfDeath	0.36	0.70	0.47	**0.39**	0.67	**0.50**	0.38	0.69	0.49	0.33	0.31	0.32	0.24	0.25	0.24	0.18	0.13	0.15	0.21	0.67	0.32	0.24	0.62	0.34	0.25	0.62	0.35	0.09	**0.94**	0.17
	Profession	0.40	0.85	**0.55**	0.40	0.85	**0.55**	**0.41**	0.85	**0.55**	0.03	0.02	0.02	0.09	0.06	0.07	0.03	0.02	0.02	0.27	0.79	0.40	0.26	0.83	0.39	0.26	0.80	0.40	0.10	**1.00**	0.18
	Religion	0.37	0.77	0.50	0.32	0.83	0.46	**0.38**	0.77	**0.51**	0.03	0.02	0.02	0.08	0.05	0.06	0.08	0.04	0.06	0.24	0.69	0.35	0.25	0.73	0.38	0.24	0.73	0.36	0.12	**0.94**	0.21
	Spouse	0.27	0.89	0.41	0.32	**0.90**	0.47	**0.33**	0.89	**0.48**	0.25	0.17	0.20	0.24	0.30	0.27	0.26	0.27	0.26	0.12	0.59	0.20	0.12	0.65	0.20	0.14	0.49	0.21	0.19	0.67	0.29
NELL-AR-10	ActorStarredInMovie	0.14	0.89	0.25	0.11	**0.94**	0.19	0.11	**0.94**	0.20	0.42	0.59	0.49	**0.47**	0.76	**0.58**	0.45	0.73	0.56	0.08	0.50	0.14	0.20	0.72	0.31	0.16	0.72	0.26	0.10	0.78	0.18
	AthletePlaysForTeam	0.33	0.80	**0.46**	0.24	0.83	0.37	0.19	**0.85**	0.31	**0.39**	0.51	0.44	0.14	0.07	0.09	0.22	0.15	0.18	0.18	0.67	0.28	0.30	0.54	0.38	0.24	0.52	0.33	0.13	0.80	0.23
	CityLocatedInState	0.15	0.61	0.24	0.15	0.66	0.24	0.15	0.62	0.25	**0.43**	0.57	**0.49**	0.21	0.13	0.16	0.09	0.05	0.06	0.13	0.75	0.22	0.11	0.56	0.19	0.14	0.62	0.23	0.11	**0.92**	0.20
	JournalistWritesForPublication	**0.47**	0.89	**0.61**	0.40	0.92	0.56	0.39	0.92	0.55	0.34	0.35	0.34	0.28	0.22	0.25	0.26	0.22	0.24	0.24	0.68	0.36	0.25	0.60	0.36	0.22	0.82	0.35	0.18	**0.97**	0.30
	RiverFlowsThroughCity	0.15	0.65	0.24	0.13	**0.81**	0.23	0.12	**0.81**	0.21	0.37	0.51	**0.43**	**0.39**	0.37	0.38	**0.39**	0.40	0.39	0.12	0.23	0.16	0.12	0.42	0.19	0.10	0.79	0.17	0.10	0.47	0.17
	SportsteamPositionAthlete	0.07	0.92	0.12	0.07	**1.00**	0.13	0.06	0.85	0.11	0.22	0.17	0.19	**0.36**	0.39	**0.38**	0.32	0.22	0.26	0.12	0.46	0.19	0.12	0.69	0.20	0.11	0.46	0.18	0.06	0.08	0.06
	StadiumLocatedInCity	0.15	0.77	0.24	0.11	0.91	0.20	0.12	**0.95**	0.21	**0.42**	0.56	**0.48**	0.37	0.35	0.36	0.29	0.30	0.30	0.35	0.41	0.38	0.21	0.68	0.32	0.18	0.59	0.27	0.20	0.09	0.13
	TeamPlaysInLeague	0.47	0.85	0.60	0.55	**0.88**	0.67	**0.56**	0.86	**0.68**	0.08	0.04	0.06	0.09	0.04	0.06	0.00	0.00	0.00	0.32	0.77	0.46	0.33	0.79	0.47	0.35	0.62	0.45	0.19	0.86	0.31
	WriterWroteBook	0.13	0.86	0.23	0.12	0.89	0.21	0.14	**0.93**	0.24	**0.45**	0.71	**0.55**	**0.45**	0.68	0.54	0.44	0.69	0.54	0.13	0.34	0.19	0.15	0.58	0.23	0.21	0.46	0.28	0.10	0.77	0.18
WN11-AR-10	DomainRegion	0.22	0.14	0.17	**0.42**	0.42	0.42	0.38	0.47	0.42	0.39	0.63	**0.48**	0.30	0.40	0.34	0.39	0.62	**0.48**	0.09	0.12	0.10	0.09	0.42	0.15	0.09	0.38	0.14	0.09	**0.90**	0.16
	DomainTopic	0.16	0.09	0.11	0.26	0.17	0.21	0.27	0.13	0.18	**0.49**	0.93	**0.64**	0.30	0.37	0.33	0.47	0.89	0.62	0.11	0.33	0.17	0.10	0.19	0.13	0.11	0.45	0.18	0.11	**0.94**	0.20
	HasInstance	**0.92**	0.74	**0.82**	0.52	**0.82**	0.64	0.54	**0.82**	0.65	0.17	0.20	0.18	0.14	0.16	0.15	0.16	0.18	0.17	0.15	**0.82**	0.25	0.20	0.71	0.31	0.20	0.64	0.30	0.20	0.65	0.30
	HasPart	0.63	0.81	0.71	0.59	**0.86**	0.70	**0.65**	**0.86**	**0.74**	0.13	0.15	0.14	0.06	0.07	0.06	0.12	0.13	0.12	0.17	0.75	0.28	0.17	0.70	0.27	0.16	0.71	0.26	0.13	0.79	0.23
	MemberMeronym	0.12	0.47	0.19	0.22	0.74	0.34	0.32	**0.81**	**0.46**	**0.35**	0.53	0.42	0.19	0.23	0.21	0.31	0.44	0.37	0.08	0.10	0.09	0.07	0.32	0.12	0.08	0.50	0.14	0.09	0.77	0.16
	PartOf	**0.67**	0.78	**0.72**	0.56	0.85	0.67	0.61	**0.86**	0.71	0.16	0.19	0.17	0.09	0.09	0.09	0.14	0.16	0.15	0.27	0.66	0.38	0.21	0.74	0.32	0.20	0.71	0.31	0.19	0.68	0.30
	SimilarTo	0.24	0.90	**0.39**	0.23	**0.91**	0.37	0.24	0.90	0.38	0.22	0.27	0.24	**0.33**	0.48	**0.39**	0.22	0.28	0.25	0.15	0.38	0.22	0.12	0.59	0.21	0.14	0.51	0.22	0.23	0.82	0.35
	SubordinateInstanceOf	0.72	0.89	0.80	0.72	0.92	0.81	**0.75**	**0.93**	**0.83**	0.11	0.12	0.11	0.04	0.04	0.04	0.09	0.10	0.10	0.38	0.68	0.49	0.36	0.75	0.49	0.34	0.76	0.47	0.18	0.90	0.30
	TypeOf	**0.84**	0.82	**0.83**	0.77	0.84	0.81	0.80	0.84	0.82	0.15	0.17	0.16	0.13	0.15	0.14	0.14	0.16	0.15	0.22	**0.88**	0.36	0.41	0.79	0.54	0.38	0.78	0.51	0.26	0.80	0.39
WN18-AR-10	AlsoSee	0.25	0.79	0.38	**0.26**	0.82	**0.39**	0.25	**0.83**	0.38	0.22	0.26	0.24	0.23	0.30	0.26	0.17	0.21	0.19	0.11	0.56	0.19	0.14	0.56	0.23	0.12	0.56	0.20	0.13	0.76	0.22
	DerivationallyRelatedForm	0.28	0.87	0.43	0.41	0.90	0.57	**0.43**	**0.92**	**0.59**	0.09	0.09	0.09	0.15	0.17	0.16	0.08	0.08	0.08	0.20	0.81	0.32	0.27	0.75	0.39	0.25	0.74	0.37	0.25	0.87	0.38
	HasPart	0.16	0.69	0.26	0.21	0.70	0.33	0.24	**0.73**	0.36	**0.36**	0.53	**0.43**	0.32	0.44	0.37	0.32	0.45	0.37	0.08	0.39	0.13	0.08	0.49	0.14	0.08	0.45	0.13	0.08	0.59	0.14
	Hypernym	0.31	0.50	0.38	0.23	0.60	0.33	0.27	**0.68**	0.39	**0.37**	0.54	**0.44**	0.36	0.52	0.42	0.36	0.50	0.42	0.14	0.54	0.22	0.17	0.45	0.24	0.16	0.47	0.24	0.13	0.39	0.20
	InstanceHypernym	**0.44**	0.50	**0.47**	0.29	0.77	0.42	0.31	0.82	0.45	0.32	0.45	0.37	0.35	0.52	0.42	0.33	0.48	0.39	0.20	0.69	0.31	0.20	0.69	0.31	0.19	0.72	0.29	0.12	**0.85**	0.22
	MemberHolonym	0.17	0.21	0.19	0.19	0.69	0.30	0.26	**0.81**	**0.39**	0.26	0.33	0.29	**0.28**	0.39	0.33	0.23	0.29	0.26	0.12	0.40	0.19	0.12	0.53	0.19	0.13	0.38	0.20	0.11	0.45	0.18
	MemberOfDomainRegion	0.27	0.17	0.21	0.17	0.28	0.21	0.19	0.26	0.22	0.33	0.41	0.37	0.43	0.66	0.52	**0.47**	**0.87**	**0.61**	0.12	0.25	0.17	0.11	0.77	0.20	0.12	0.22	0.16	0.14	0.53	0.23
	MemberOfDomainUsage	0.15	0.09	0.11	0.44	0.29	0.35	**0.48**	0.30	0.37	0.39	0.46	0.42	0.40	0.57	**0.47**	0.39	0.57	0.46	0.10	0.53	0.16	0.09	0.47	0.15	0.08	0.23	0.11	0.11	**0.87**	0.19
	SimilarTo	**0.26**	**1.00**	**0.42**	0.19	**1.00**	0.32	0.21	**1.00**	0.34	0.00	0.00	0.00	0.21	0.25	0.23	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	SynsetDomainTopicOf	0.30	0.60	**0.40**	0.26	0.76	0.39	0.27	0.75	0.39	0.32	0.43	0.37	**0.33**	0.45	0.38	0.32	0.44	0.37	0.13	**0.79**	0.23	0.16	0.71	0.26	0.20	0.53	0.29	0.14	0.69	0.23
	VerbGroup	0.36	0.92	0.52	**0.40**	0.94	**0.56**	0.38	**0.96**	0.54	0.04	0.04	0.04	0.13	0.14	0.14	0.04	0.04	0.04	0.14	0.61	0.23	0.26	0.44	0.33	0.21	0.57	0.30	0.36	0.61	0.46

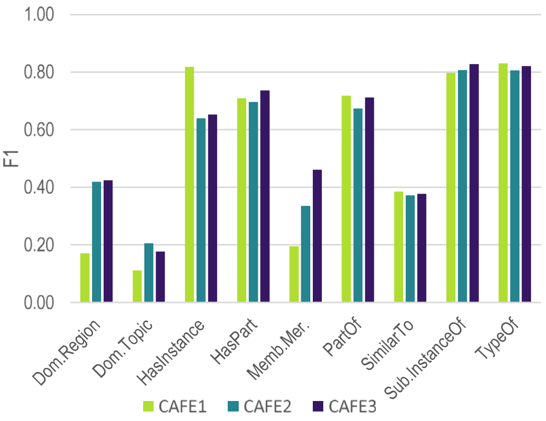


Fig. 5. F1 scores in the WN11-AR-10 dataset.

relevant entities. In the cases of WN11-AR-10 (Fig. 4(c)) and WN18-AR-10 (Fig. 4(d)), an improvement is observed when using neighborhoods of size 2, but further expanding the neighborhood size does not seem to have a significant impact. A possible conclusion for this is that, at a certain point, larger neighborhood subgraphs do not provide additional value or predictive power over smaller ones. In a worst-case scenario, the number of features with little to no predictive power would greatly increase by using larger neighborhood sizes, negatively affecting the results. This effect actually occurs in the case of the NELL-AR-10 dataset (Fig. 4(b)), where effectiveness decreases when increasing the size of the neighborhood subgraphs for which we compute features. A plausible explanation for this is that it has been noted that NELL is a noisier dataset (Paulheim and Bizer, 2014), and thus taking larger neighborhoods into account significantly increases the amount of noise that our classification model has to deal with, reducing its effectiveness.

5.4. Limitations and performance considerations

Despite our proposal being generally effective, it has some limitations. CAFE does not work well for relations for which useful predictive information cannot always be found in the neighborhoods of the entities in a triple. In this regard, we identify two types of relations: those that can be predicted using other information present in the KG, and those for which the entities and relations in their neighborhoods do not provide useful information, and thus are much harder to predict. This dichotomy can be observed in the results shown in Table 2, and it is especially notable in the WN11-AR-10 dataset. For the sake of visualization, we display the results of applying CAFE to this dataset in Fig. 5.

This difference is particularly visible in the *Similar to* and *Domain topic* relations, which have low F1 scores. Upon manual inspection, we found that *Similar to* tends to link words that are generally isolated within the KG and share very little common context. Also, *Domain topic* is extremely broad, causing the two entities in a triple to have few or no relevant common elements in their neighborhoods, e.g., (*Britain*, *Domain topic*, *Surgery*).

In other cases, even if some information is present in the neighborhoods under consideration, it may be not successfully captured by CAFE due to the large amount of irrelevant data surrounding it. In this regard, the NELL-AR-10 dataset shows that larger neighborhood sizes are not always better for predictive purposes, since the amount of noise they may include can be detrimental for the performance of CAFE. This can be due to some source or target entities being present in many triples (for example, countries). Therefore, larger neighborhoods of these entities introduce many other entities that are not relevant to

Table 3
Time in seconds (average ± standard deviation) needed to compute features for all training and testing triples in every dataset.

Dataset	CAFE$_1$	CAFE$_2$	CAFE$_3$
FB13-A-10	17.21 ± 0.52	67.98 ± 2.87	183.38 ± 7.71
NELL-AR-10	0.56 ± 0.06	2.41 ± 0.23	6.95 ± 0.62
WN11-AR-10	8.69 ± 0.26	50.85 ± 0.83	195.50 ± 16.77
WN18-AR-10	5.70 ± 0.12	18.90 ± 0.29	55.57 ± 0.91

the triple under evaluation. In these cases, it is up to the users to decide which maximum neighborhood size best caters to their interests.

Additionally, due to the fact that feature instances are obtained from feature groups by considering all possible combination of the values of their parameters, as explained in Section 3.2, the number of individual feature instances associated with every neighborhood size has a theoretically exponential growth with respect to the number of relations in the dataset. However, several optimizations can be leveraged to reduce the amount of time needed to compute the required features for all triples in a KG. Given that this computation can be done independently for each triple, our implementation uses multi-threading to parallelize this task, and could theoretically benefit from further performance improvements under a distributed computing architecture. Furthermore, since our features are deterministic and always return the same value for the same input, we use a caching strategy to minimize the amount of feature computations needed. To illustrate this, we display the average CPU times required by CAFE to transform all triples in a dataset into feature vectors, across 10 different executions for all datasets and context sizes, in Table 3. It can be noted that, in all cases, the computation times were quite reasonable, which suggests that our proposal is suitable to be applied in a real-world scenario.

6. Conclusions

In this paper, we have introduced CAFE, a proposal that classifies triples for Knowledge Graph completion by using a feature set based upon neighborhoods and neural prediction models. This feature set, and by extension our proposal, is applicable to all KGs, and it is completely deterministic. Additionally, CAFE does not need any manually supplied or external information, nor does it need any previous preprocessing of the KG, and it is able to operate using a KG as its only input.

We have performed a number of experiments to evaluate the effectiveness of CAFE in four well-known Knowledge Graphs, identifying candidate triples that should be added to each KG. To provide a more realistic evaluation scenario, we have performed our experiments using testing splits that contain a much lower amount of true triples than false ones. Our results show that applying the proposed feature set yields an effectiveness that is comparable to state-of-the-art techniques, and in many cases even higher. CAFE achieves higher average F1 values in all datasets under study, thus leading to a more trustworthy KG completion process. We also show that using information present in the neighborhoods of the two entities in a triple provides satisfactory results, and thus local information existing around any given entity appears to be of great value.

CRediT authorship contribution statement

Agustín Borrego: Conceptualization, Methodology, Software, Investigation, Writing - original draft. **Daniel Ayala:** Conceptualization, Methodology, Software, Investigation, Writing - original draft. **Inma Hernández:** Conceptualization, Writing - review & editing, Supervision, Funding acquisition, Project administration. **Carlos R. Rivero:** Conceptualization, Writing - review & editing, Supervision. **David Ruiz:** Conceptualization, Writing - review & editing, Supervision, Funding acquisition, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the Spanish Ministry of Science and Innovation under grants TIN2016-75394-R and PID2019-105471RB-I00.

References

Aggarwal, Charu C., Zhai, ChengXiang, 2012. A survey of text classification algorithms. In: Mining Text Data. Springer, pp. 163–222. http://dx.doi.org/10.1007/978-1-4614-3223-4_6.

Altan, Aytaç, Karasu, Seçkin, 2020. Recognition of COVID-19 disease from X-ray images by hybrid model consisting of 2D curvelet transform, chaotic salp swarm algorithm and deep learning technique. Chaos Solitons Fractals 140, 110071. http://dx.doi.org/10.1016/j.chaos.2020.110071.

Ayala, Daniel, Borrego, Agustín, Hernández, Inma, Rivero, Carlos R., Ruiz, David, 2019a. AYNEC: All you need for evaluating completion techniques in knowledge graphs. In: ESWC. pp. 397–411. http://dx.doi.org/10.1007/978-3-030-21348-0_26.

Ayala, Daniel, Hernández, Inma, Ruiz, David, Toro, Miguel, 2019b. TAPON: A two-phase machine learning approach for semantic labelling. Knowl.-Based Syst. 163, 931–943. http://dx.doi.org/10.1016/j.knosys.2018.10.017.

Bansal, Iti, Tiwari, Sudhanshu, Rivero, Carlos R., 2020. The impact of negative triple generation strategies and anomalies on knowledge graph completion. In: CIKM '20: The 29th ACM International Conference on Information and Knowledge Management. ACM, pp. 45–54. http://dx.doi.org/10.1145/3340531.3412023.

Bollacker, Kurt, Evans, Colin, Paritosh, Praveen, Sturge, Tim, Taylor, Jamie, 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In: SIGMOD. ACM, pp. 1247–1250. http://dx.doi.org/10.1145/1376616.1376746.

Bordes, Antoine, Chopra, Sumit, Weston, Jason, 2014a. Question answering with subgraph embeddings. In: EMNLP. ACL, pp. 615–620.

Bordes, Antoine, Gabrilovich, Evgeniy, 2014. Constructing and mining web-scale knowledge graphs. In: SIGKDD. ACM, p. 1967. http://dx.doi.org/10.1145/2623330.2630803.

Bordes, Antoine, Glorot, Xavier, Weston, Jason, Bengio, Yoshua, 2014b. A semantic matching energy function for learning with multi-relational data - Application to word-sense disambiguation. Mach. Learn. 94 (2), 233–259. http://dx.doi.org/10.1007/s10994-013-5363-6.

Bordes, Antoine, Usunier, Nicolas, García-Durán, Alberto, Weston, Jason, Yakhnenko, Oksana, 2013. Translating embeddings for modeling multi-relational data. In: NIPS. pp. 2787–2795.

Borrego, Agustín, Ayala, Daniel, Hernández, Inma, Rivero, Carlos R., Ruiz, David, 2019. Generating rules to filter candidate triples for their correctness checking by knowledge graph completion techniques. In: K-CAP. pp. 115–122. http://dx.doi.org/10.1145/3360901.3364418.

Dong, Xin, Gabrilovich, Evgeniy, Heitz, Geremy, Horn, Wilko, Lao, Ni, Murphy, Kevin, Strohmann, Thomas, Sun, Shaohua, Zhang, Wei, 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In: SIGKDD. ACM, pp. 601–610. http://dx.doi.org/10.1145/2623330.2623623.

Drumond, Lucas, Rendle, Steffen, Schmidt-Thieme, Lars, 2012. Predicting RDF triples in incomplete knowledge bases with tensor factorization. In: SAC. ACM, pp. 326–331. http://dx.doi.org/10.1145/2245276.2245341.

Galárraga, Luis Antonio, Teflioudi, Christina, Hose, Katja, Suchanek, Fabian, 2013. AMIE: Association rule mining under incomplete evidence in ontological knowledge bases. In: WWW. ACM, pp. 413–422. http://dx.doi.org/10.1145/2488388.2488425.

Galárraga, Luis, Teflioudi, Christina, Hose, Katja, Suchanek, Fabian M., 2015. Fast rule mining in ontological knowledge bases with AMIE+. VLDB J. 24 (6), 707–730. http://dx.doi.org/10.1007/s00778-015-0394-1.

Gardner, Matt, Mitchell, Tom, 2015. Efficient and expressive knowledge base completion using subgraph feature extraction. In: EMNLP. The Association for Computational Linguistics, pp. 1488–1498.

Glass, Michael, Gliozzo, Alfio, 2018. A dataset for web-scale knowledge base population. In: ESWC. Springer, pp. 256–271. http://dx.doi.org/10.1007/978-3-319-93417-4_17.

Han, Xu, Cao, Shulin, Lv, Xin, Lin, Yankai, Liu, Zhiyuan, Sun, Maosong, Li, Juanzi, 2018. OpenKE: An open toolkit for knowledge embedding. In: EMNLP. pp. 139–144.

Ho, Vinh Thinh, Stepanova, Daria, Gad-Elrab, Mohamed Hassan, Kharlamov, Evgeny, Weikum, Gerhard, 2018. Learning rules from incomplete kgs using embeddings. In: The 17th International Semantic Web Conference. ceur. ws. org.

Hogan, Aidan, Blomqvist, Eva, Cochez, Michael, d'Amato, Claudia, de Melo, Gerard, Gutierrez, Claudio, Gayo, José Emilio Labra, Kirrane, Sabrina, Neumaier, Sebastian, Polleres, Axel, Navigli, Roberto, Ngomo, Axel-Cyrille Ngonga, Rashid, Sabbir M., Rula, Anisa, Schmelzeisen, Lukas, Sequeda, Juan F., Staab, Steffen, Zimmermann, Antoine, 2020. Knowledge graphs. CoRR, abs/2003.02320.

Huang, Xiao, Zhang, Jingyuan, Li, Dingcheng, Li, Ping, 2019. Knowledge graph embedding based question answering. In: WSDM'19. pp. 105–113. http://dx.doi.org/10.1145/3289600.3290956.

Ji, Guoliang, He, Shizhu, Xu, Liheng, Liu, Kang, Zhao, Jun, 2015. Knowledge graph embedding via dynamic mapping matrix. In: ACL (1). The Association for Computer Linguistics, pp. 687–696.

Karasu, Seçkin, Altan, Aytaç, Bekiros, Stelios, Ahmad, Wasim, 2020. A new forecasting model with wrapper-based feature selection approach using multi-objective optimization technique for chaotic crude oil time series. Energy 212, 118750. http://dx.doi.org/10.1016/j.energy.2020.118750.

Kazemi, Seyed Mehran, Poole, David, 2018. Simple embedding for link prediction in knowledge graphs. In: Advances in Neural Information Processing Systems.

Kolthoff, Kristian, Dutta, Arnab, 2015. Semantic relation composition in large scale knowledge bases. In: LD4IE@ISWC, Vol. 1467. pp. 34–47.

Lao, Ni, Cohen, William W., 2010. Relational retrieval using a combination of path-constrained random walks. Mach. Learn. 81 (1), 53–67. http://dx.doi.org/10.1007/s10994-010-5205-8.

Lehmann, Jens, Isele, Robert, Jakob, Max, Jentzsch, Anja, Kontokostas, Dimitris, Mendes, Pablo N., Hellmann, Sebastian, Morsey, Mohamed, Van Kleef, Patrick, Auer, Sören, et al., 2015. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. Semant. Web 6 (2), 167–195. http://dx.doi.org/10.3233/SW-140134.

Lin, Yankai, Liu, Zhiyuan, Sun, Maosong, Liu, Yang, Zhu, Xuan, 2015. Learning entity and relation embeddings for knowledge graph completion. In: AAAI, Vol. 15. AAAI Press, pp. 2181–2187.

Lin, Peng, Song, Qi, Wu, Yinghui, 2018. Fact checking in knowledge graphs with ontological subgraph patterns. Data Sci. Eng. 3, 341–358. http://dx.doi.org/10.1007/s41019-018-0082-4.

Liu, Hanxiao, Wu, Yuexin, Yang, Yiming, 2017. Analogical inference for multi-relational embeddings. In: International Conference on Machine Learning. PMLR, pp. 2168–2178.

Mazumder, Sahisnu, Liu, Bing, 2017. Context-aware path ranking for knowledge base completion. In: IJCAI. AAAI Press, pp. 1195–1201. http://dx.doi.org/10.24963/ijcai.2017/166.

Miller, George A., 1995. WordNet: A lexical database for english. Commun. ACM 38 (11), 39–41. http://dx.doi.org/10.1145/219717.219748.

Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Yang, B., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohamed, T., Nakashole, N., Platanios, E., Ritter, A., Samadi, M., Settles, B., Wang, R., Wijaya, D., Gupta, A., Chen, X., Saparov, A., Greaves, M., Welling, J., 2018. Never-ending learning. Commun. ACM 61 (5), 103–115. http://dx.doi.org/10.1145/3191513.

Neelakantan, Arvind, Chang, Ming-Wei, 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. In: HLT-NAACL. The Association for Computational Linguistics, pp. 515–525.

Nentwig, Markus, Hartung, Michael, Ngonga Ngomo, Axel-Cyrille, Rahm, Erhard, 2017. A survey of current link discovery frameworks. Semant. Web 8 (3), 419–436. http://dx.doi.org/10.3233/SW-150210.

Neumaier, Sebastian, Umbrich, Jürgen, Parreira, Josiane Xavier, Polleres, Axel, 2016. Multi-level semantic labelling of numerical values. In: ISWC, Vol. 9981. pp. 428–445. http://dx.doi.org/10.1007/978-3-319-46523-4_26.

Nickel, Maximilian, Tresp, Volker, Kriegel, Hans-Peter, 2012. Factorizing YAGO: scalable machine learning for linked data. In: WWW. ACM, pp. 271–280. http://dx.doi.org/10.1145/2187836.2187874.

Paulheim, Heiko, 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. Semant. Web 8 (3), 489–508. http://dx.doi.org/10.3233/SW-160218.

Paulheim, Heiko, Bizer, Christian, 2014. Improving the quality of linked data using statistical distributions. Int. J. Semant. Web Inf. Syst. 10 (2), 63–86. http://dx.doi.org/10.4018/ijswis.2014040104.

Schlegel, Viktor, Freitas, André, 2019. DBee: A database for creating and managing knowledge graphs and embeddings. In: Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing, TextGraphs@EMNLP 2019. pp. 177–185. http://dx.doi.org/10.18653/v1/D19-5322.

Shen, Ying, Wen, Desi, Li, Yaliang, Du, Nan, Zheng, Hai-tao, Yang, Min, 2019. Path-based attribute-aware representation learning for relation prediction. In: Proceedings of the 2019 SIAM International Conference on Data Mining. SIAM, pp. 639–647.

Socher, Richard, Chen, Danqi, Manning, Christopher D., Ng, Andrew, 2013. Reasoning with neural tensor networks for knowledge base completion. In: NIPS. pp. 926–934.

Sun, Zhiqing, Deng, Zhi-Hong, Nie, Jian-Yun, Tang, Jian, 2019. RotatE: Knowledge graph embedding by relational rotation in complex space. In: International Conference on Learning Representations.

Trouillon, Théo, Welbl, Johannes, Riedel, Sebastian, Gaussier, Éric, Bouchard, Guillaume, 2016. Complex embeddings for simple link prediction. In: ICML, Vol. 48. pp. 2071–2080.

Wang, Zihan, Ren, Zhaochun, He, Chunyu, Zhang, Peng, Hu, Yue, 2019. Robust embedding with multi-level structures for link prediction. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence. AAAI Press, pp. 5240–5246.

Wang, Zhen, Zhang, Jianwen, Feng, Jianlin, Chen, Zheng, 2014. Knowledge graph embedding by translating on hyperplanes. In: AAAI, Vol. 14. AAAI Press, pp. 1112–1119.

Xiong, Wenhan, Hoang, Thien, Wang, William Yang, 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In: EMNLP.

Yadav, Vikas, Bethard, Steven, 2019. A survey on recent advances in named entity recognition from deep learning models. CoRR.