Semantic Web 0 (2020) 1–0
IOS Press

CAFE: Fact Checking in Knowledge Graphs using Neighborhood-Aware Features

Agustín Borrego [a,*], Daniel Ayala [a], Inma Hernández [a], Carlos R. Rivero [b] and David Ruiz [a]
[a] *University of Seville, Av. Reina Mercedes s/n, Seville, Spain*
E-mails: borrego@us.es, dayala1@us.es, inmahernandez@us.es, druiz@us.es
[b] *Rochester Institute of Technology, 92 Lomb Memorial Drive, Rochester NY, USA*
E-mail: crr@cs.rit.edu

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 potentially lack useful and valuable information. Current approaches that aim to complete missing information in KGs either have a dependence on embedded representations, which hinders their scalability and applicability to different KGs; or are based on long random paths that may not cover relevant information by mere chance, since exhaustively analyzing all possible paths of a large length between entities is very time-consuming. In this paper, we present an approach to completing KGs based on evaluating candidate triples using a novel set of features, which exploits the highly relational nature of KGs by analyzing the entities and relations surrounding any given pair of entities. Our results show that our proposal is able to identify correct triples with a higher effectiveness than other state-of-the-art approaches (up to 60% higher precision or 20% higher recall in some datasets).

Keywords: Knowledge Graph Completion, Fact Checking, Machine Learning

1. Introduction

Knowledge Graphs (KGs) are vast repositories of structured information whose growth over the past years can be related to that of the Web of Data [1]. Large volumes of information about different domains can be found in some well-known KGs such as DBpedia [2], NELL [3], Freebase [4] or the Google Knowledge Vault [5], which can then be used for tasks such as question answering [6].

KGs are most commonly built by extracting non-structured [3, 5] or semi-structured [2, 7] information from web sources, though some smaller KGs can be manually curated by domain experts [8]. After information extraction systems have been applied to extract knowledge from online sources, said information is then semantized [9, 10] and stored in a KG as entities and relations between those entities. Regardless of the specific process by which a KG is constructed, said KG usually lacks a certain amount of information, either because said information was not originally present in the original information source, or because it was not extracted or semantized correctly [11]. Because of this inherent incompleteness, KGs operate under the Open World Assumption, i.e., a piece of information that is not contained in a KG is not considered to be incorrect, but rather just unknown [12]. Therefore, it is mandatory to refine KGs after their creation in order to expand the knowledge they contain, so that they contain more accurate information [13]. Such is the example of the Google Knowledge Vault [5], a KG that is commonly used for question answering and that is under constant expansion and refinement.

As a motivational example, Figure 1 presents a KG that contains information about fictional works, actors, writers and characters. This KG is incomplete, as it is missing certain information that is true in the real world, such as the fact that *Daniel Radcliffe* played the character *Harry Potter*.

*Corresponding author. E-mail: borrego@us.es.

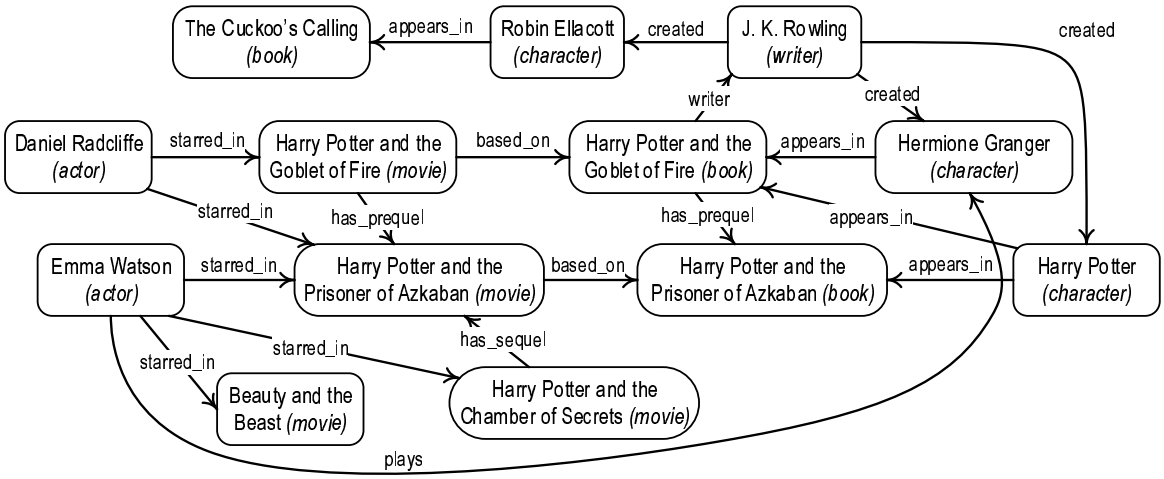


Fig. 1. Example KG describing works, actors, writers and characters

Deriving additional knowledge from an existing KG to augment it is a task known as Knowledge Graph completion [13]. KG completion can be approached with two complementary goals in mind: completing information about entity types [14], or completing the relations between entities [15–18]. Furthermore, these techniques can be based either on external features, i.e., features that depend on external information sources, or on purely internal features, i.e., features that are computed using only the KG itself [19, 20]. In this work, we focus in KG completion using only internal features, since this line of work has shown promising results without dependencies to external sources of information.

When completing a KG, one has to produce a set of facts that are not contained in said KG but that have a reasonable chance of being true. This is usually done through a prediction model that has been trained beforehand. The task of creating, training and applying such a model is known as fact checking [21, 22]. Thus, it is desirable for these models to achieve a high precision to ensure that the facts that are considered to be correct are actually true in the real world [23].

There are several fact checking proposals in the literature that use solely internal features [12, 15, 18, 20, 24, 25], which we discuss in more detail in Section 2. Unfortunately, these approaches can suffer from a number of drawbacks. Some of them rely on embedded representations of the entities and/or relations, and both these representations and the associated neural models need to be completely recomputed whenever a few new triples, which is a relatively frequent event [3, 5]. Additionally, there exists a number of different, random walk-based proposals that may miss relevant information due to their non-deterministic nature.

In this paper we present CAFE, our approach for Fact **C**hecking using Neighborhood-**A**ware **FE**atures, which uses a novel feature set that extracts information from the neighborhoods of the entities, i.e., their nearby entities and relations. This feature set is used to transform triples in the KG into feature vectors, which are fed to neural prediction models that discern between correct triples that should be added to the KG, and incorrect ones that should be disregarded. Different than embedding-based proposals, the features CAFE is based on can be directly applied to any KG without the need of computing embedded representations, and do not require to be recomputed as the KG grows. They are also not based on random paths, which ensures that all contextual information is always taken into account. Furthermore, the experimental results show that CAFE is able to predict which relations should be added to a KG with a higher precision 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 CAFEuses, Section 4 presents our proposal for training predictive models for triples in a KG, Section 5 reports our experimental results; finally, Section 6 presents our conclusions.

2. Related Work

In the field of Fact Checking, there are rule-based, embedding-based and path-based proposals:

Rule-based proposals rely on discovering logical rules that determine when a given triple is correct. In this regard, some authors propose using Inductive Logic Programming to mine Horn rules that express the relations between entities in a KG, and then applying said rules to produce new explicit knowledge [16, 26], 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 [12].

Those 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 feature vectors for a prediction model [15], or by performing different transformations in an embedding space [18, 20, 25, 27, 28]. 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 [19, 29]. These proposals usually rank triples by decreasing order of likelihood, according to how close the two entities in the triples are in the generated space. Thus, they are commonly evaluated using metrics that are based on some ranking of likely correct candidate triples [30], however, a likelihood threshold can be set to obtain classical classification measures such as precision, recall and F1 [31].

Finally, those that are path-based exploit the highly relational nature of KGs to learn how to predict new relations between entities. Our approach, CAFE, falls under this category. In this line of work, Lao and Cohen [17] introduce 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 in [24] by Gardner and Mitchell. SFE achieves a better performance than PRA and produced more expressive results while running faster than PRA. It also introduced a handmade "Alias" relation, which relates entities in the same KG that have different labels but refer to the same element in the real world. In [32], Mazumder and Liu 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 [33]. The evaluation metrics used by these proposals are similar to those used by the proposals based on embeddings.

Nowadays, there exist a number of KG completion proposals that combine different approaches. For example, in [34] the authors propose obtaining embeddings of the entities and the relations, and then combining these embeddings in the forms of paths, which 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 in [30], where they use graph neural networks to capture information pertaining the structure of the graph. Entity embeddings have also been used to aid in the production of KG completion rules [35].

3. Neighborhood-Aware Features

In this Section, we first introduce some preliminary concepts that are necessary to understand our proposal, and then we define our set of neighborhood-aware features.

3.1. Preliminaries

We now define some preliminary concepts and the notation that is used throughout the paper.

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 example KG depicted in Figure 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 a 3-tuple $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$.*

Figure 1 graphically represents a KG, with 13 entities and 7 distinct relations.

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 as* $path(s, t, r_1, r_2, \ldots, r_n)$, *or* $path_n(s, t)$ *for short. We denote the set of all possible distinct paths of the form* $path(s, t, r_1, r_2, \ldots, r_n)$ *as* $\mathcal{P}(s, t, r_1, r_2, \ldots, r_n)$.

In the KG depicted in Figure 1, an example 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* $n \geqslant 1$ *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.,* $\{t \in \mathcal{E} \mid Reach(\mathcal{KG}, s, t, r, n)\}$. *We denote the previously defined set as* $Reachable(s, r, n)$.

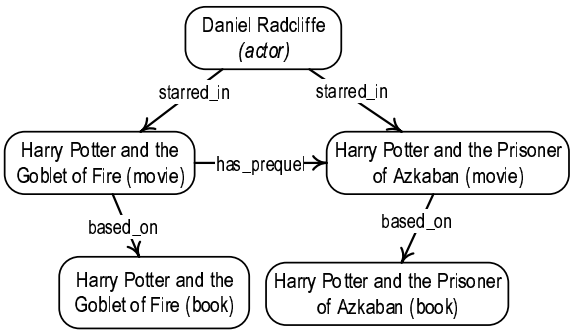
In the example KG depicted in Figure 1, $Reachable(Hermione\ Granger, writer, 2) = \{Hermione\ Granger, J.K.\ Rowling\}$.

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* $n \geqslant 1$ *be a natural number. We define the neighborhood subgraph of e of size n as a Knowledge Graph* $\mathcal{KG}_e^n = (\mathcal{E}_e^n, \mathcal{R}, \mathcal{T}_e^n)$ *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* $\mathcal{T}_e^n = \{(s', r', t') \in \mathcal{T} \mid Reach(\mathcal{KG}, e, t', r', i),\ i = 1..n\}$ *and* $\mathcal{E}_e^n = \bigcup \{\{s, t\} \subseteq \mathcal{E} \mid (s, r, t) \in \mathcal{T}_e^n\}$.

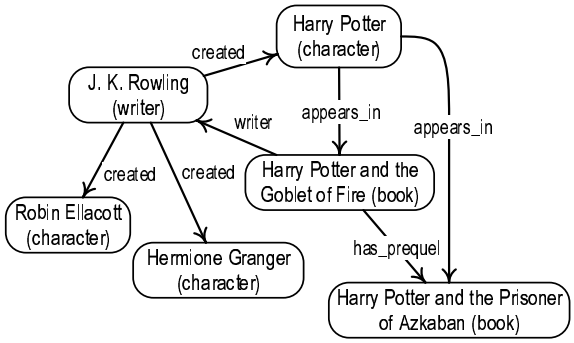
Figure 2 illustrates this definition, with two possible neighborhood subgraphs for the KG shown in Figure 1.

3.2. Feature set

We devised a neighborhood-aware set of features which 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, where each group can be parameterized to obtain a specific feature, which we call an instance of the feature group.



(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 example KG depicted in Figure 1

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., $f : (s, r, t) \mapsto |\mathcal{E}_s^2|$.

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., $f_0(n) = f : (s, r, t) \mapsto |\mathcal{E}_s^n|$, where n is a parameter of the feature group. Thus, $f_0(2) : (s, r, t) \mapsto |\mathcal{E}_s^2|$, 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.

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 Figure 1 and the neighborhood subgraphs described in Figure 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 can be computed as $f_1(\mathrm{n}) : (s,r,t) \mapsto |\mathcal{E}_s^n|$.
In the example shown in 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 can be computed as $f_2(\mathrm{n}) : (s,r,t) \mapsto |\mathcal{E}_t^n|$.
In the example shown in 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: 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 can be computed as $f_3(\mathrm{n}, \mathrm{m}) : (s,r,t) \mapsto |\mathcal{E}_s^n \cap \mathcal{E}_t^m|$.
In the example shown in Fig. 2, $f_3(2,\ 3)$ 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_4: Jaccard index of similarity between the entity sets for 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 can be computed as $f_4(\mathrm{n}, \mathrm{m}) : (s,r,t) \mapsto jaccard(\mathcal{E}_s^n, \mathcal{E}_t^m)$.
In the example shown in Fig. 2, $f_4(2,\ 3)$ 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_5: Number of reachable entities through the relation r at distance n from the source entity in the triple.

Features in this group can be computed as $f_5(\mathrm{r}, \mathrm{n}) : (s,r,t) \mapsto |Reachable(s,r,n)|$.
In the example shown in Fig. 2(a), $f_5($*2, hasPrequel*$)$ applied to the *example* triple is $|\{$*Daniel Radcliffe, Harry Potter and the Prisoner of Azkaban (movie)*$\}| = 2$.

Feature group f_6: Number of reachable entities through the relation r at distance n from the target entity in the triple.

Features in this group can be computed as $f_6(\mathrm{r}, \mathrm{n}) : (s,r,t) \mapsto |Reachable(t,r,n)|$.
In the example shown in Fig. 2(b), $f_6($*2, created*$)$ applied to the *example* triple is $|\{$*Harry Potter,Hermione Granger, Robin Ellacott*$\}| = 3$.

Feature group f_7: 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 can be computed as $f_7(\mathrm{r}, \mathrm{n}, \mathrm{m}) : (s,r,t) \mapsto |Reachable(s,r,n) \cap Reachable(t,r,m)|$.
In the example shown in Fig. 2, $f_7($*2, 3, created*$)$ applied to the *example* triple is $|\{$*Daniel Radcliffe*$\} \cap \{$*Harry Potter, Hermione Granger, Robin Ellacott*$\}| = 0$.

Feature group f_8: 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 can be computed as $f_8(\mathrm{r}, \mathrm{n}, \mathrm{m}) : (s,r,t) \mapsto jaccard(Reachable(s,r,n), Reachable(t, r, m))$.
In the example shown in Fig. 2, $f_8($*2, 3, created*$)$ applied to the *example* triple is $|\emptyset|$ / $|\{$*Harry Potter, Hermione Granger, Robin Ellacott*$\}| = 0 \,/\, 3 = 0$.

Feature group f_9: Number of distinct paths of length n between the source and the target entity in the triple, using relations $r_1, \ldots, r_n$.

Features in this group can be computed as $f_9(\mathrm{n}, \mathrm{r}_1, \ldots, \mathrm{r}_\mathrm{n}) : (s, r, t) \mapsto |\mathcal{P}(s, t, r_1, \ldots, r_n)|$.
In the example shown in Fig. 1, f_9(*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.

The rationale behind this set of feature groups is mani: first, computing the common entities and indices of similarities of the neighborhood subgraphs for the source and target entities allows CAFE to find out the degree of overlap that exists in the neighborhoods of the two relations, along with detailed information on whether this overlap occurs for different relations. Second, studying all kinds of paths that exist between the entities in a triple allows CAFE to also benefit from path-based information in a deterministic manner. Finally, they can be applied at any time as a KG grows with new entities and relations, without the need of a complete recomputation in opposition to embedding-based approaches.

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 selected relations. These models are able to determine if a given triple that represents an instance of the relation is correct, i.e., if it should belong to the KG. Our workflow is visually depicted in Figure 3 and, in the following subsections, we describe each one of its steps.

4.1. Loading the KG

We are interested only in the triples contained in a KG, as both the entities and the relations sets can be derived from them. CAFE internally stores the KG it is given in the form of (s, r, t) triples, using an efficient graph-like structure. The triples that contain relations for which a predictive model does not need to be generated will still be taken into account when computing features, since they may provide valuable predictive information.

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, but it does not contain explicit information about pairs of entities for which r does not hold. Since our proposal relies on a neural prediction model that requires negative examples during its training, a number of them has to be produced. To do this, we follow the same approach as other authors and consider any triple not present in the KG to be a negative example [15, 25, 32]. To train our models, we generate one negative example for each triple existing in the KG by replacing its target entity t with a different one, t', such that the resulting triple (s, r, t') does not exist in the KG. To preserve the range of each relation, we randomly choose a t' such that there exists some other triple in the KG where t' appears as the target entity for the relation r. A positive or negative label is then added to every triple, denoting whether it represents positive or negative evidence.

In the example depicted in Figure 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*, while *(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 false because they do not exist in the KG, but that are true in the real world. While this is indeed plausible, we as other authors [15, 18, 20] consider that the chances of this happening are low enough for it to not have a noticeable outcome in our results. Furthermore, it may even prevent our prediction models from overfitting to the available evidence.

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 the Cartesian product of its possible parameters but, for efficiency reasons, we limit parameters that represent neighborhood sizes to a maximum value of 3. Each feature instance then assigns each triple a real number, and thus applying several features to a triple results in a feature vector, in which

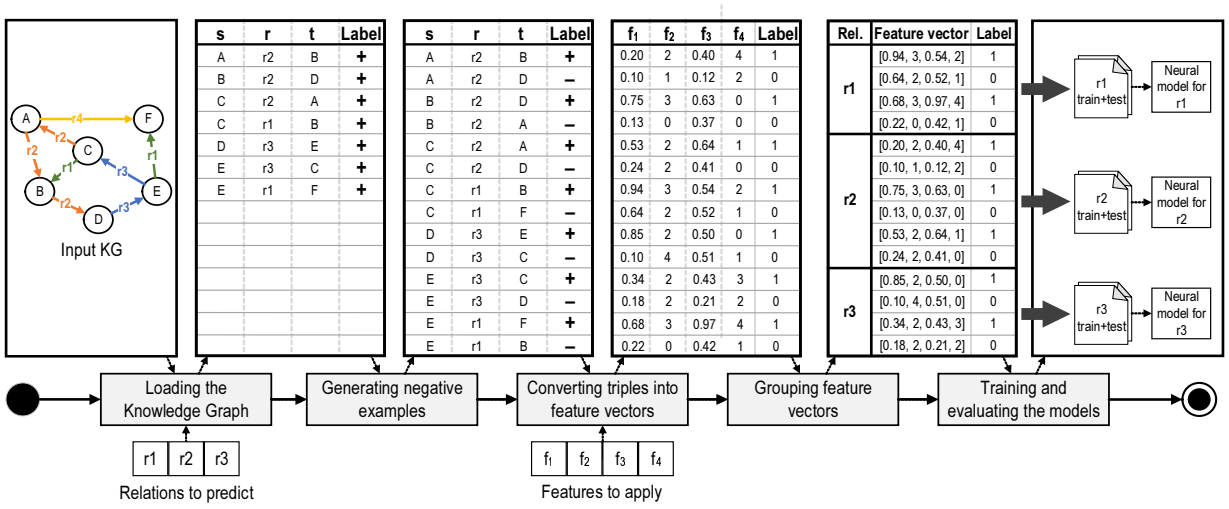


Fig. 3. Workflow of CAFE

each position represents the value that the corresponding feature assigns to the triple. The labels that were added in the previous step, which indicate if the triple is a positive or a negative example, are converted into a numeric label.

It is important to note that, to compute features on a triple, we temporarily remove said triple from the KG (in case it is a positive example and thus exists in 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 labeled feature vectors that result from the previous step are grouped by the relation that was present in the original triples, resulting in a group of labeled vectors for every relation for which we train a prediction model. For example, the feature vector produced by the triple *(Daniel Radcliffe, plays, Harry Potter)* is added to the set of feature vectors for the relation *plays* alongside its label, which denotes whether it is a positive or a negative example. These groups are then split into training and test sets, resulting in two sets of labeled feature vectors per relation.

4.5. Training and evaluating the models

For every relation that we predict, we create one or more neural models (whose hyperparameters can be found in Section 5.2), where each model is fed 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 layer of three neurons and an output layer of one neuron.

Said 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. This is due to some features being relatively sparse (especially path-based features, since only a small subset of all possible paths of fixed length occur between two given entities), and thus helps the model focus only on potentially useful features.

5. Evaluation

In this section, we present our experimental results for several datasets, and we compare our proposal against other state-of-the-art fact checking techniques. In the following, we introduce the datasets that we used for evaluation, our experimental settings and, finally, our results.

5.1. Datasets

We evaluated our proposal using four datasets for KG completion provided by the freely available AYNEC-DataGen [36] tool: FB13-A, WN11-AR, WN18-AR and NELL-AR. These datasets are based on the well-known FB13, WN11 [15], WN18 [37] and a subset of NELL proposed in [24]. However, they have been processed to remove reciprocal relations detected by AYNEC (that is, relations r and r' where if (s, r, t) exists then (t, r', s) also exists very frequently). Additionally, relations that make up for less than 5% of the total amount of triples in the graph have been removed.

These datasets originally contained one negative example per each positive one in both their training and testing splits. In order to study how the KG completion techniques perform when presented with a much higher number of negative evidence, we created versions of these datasets whose testing splits contain, whenever possible, 10 negative examples per each positive one using the AYNEC-DataGen tool. Exceptions may apply for relations where the set of possible target entities is very small, e.g., the relation *gender*. In these cases we generated as many negative examples as possible, up to 10 per positive example. We believe that this is a more realistic scenario, since a real-world KG completion task would be presented with a much higher number of false candidate triples than true ones [21]. 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 is 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 [24]. However, in the later 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 some 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 [15]. This was done in a random manner, and 1 in 4 such triples was transferred to the testing set. Table 1 provides an overview on the aforementioned datasets.

[1] https://github.com/tdg-seville/CAFE

5.2. Evaluation method

A neural prediction model is created for every relation of interest and trained using its corresponding training set. Then, the model is applied to all feature vectors in the test set, and we compare 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 it determines that a triple should be added to the KG. We computed these metrics using the AYNEC-ResTest tool [36].

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 performance of CAFE. In the specific case of the NELL dataset, we only used CAFE_1 and CAFE_2. The reason for this is that the larger amount of total relations made it impractical to reach neighborhoods of size 3, as the number of features that would have to be computed was large.

We use TransE [38], TransD [20], TransH [25], TransR [18] and ComplEx [39] as baselines for our evaluation, since they are well-known state-of-the-art KG completion proposals. In order to evaluate these different embedding-based proposals in a fair and level manner, we use the OpenKE [31] framework to train and evaluate these proposals using our datasets. Additionally, since these baselines proposals usually employ metrics such as MRR and Precision@N, we used the utilities provided by the OpenKE tool to obtain binary labels for the testing triples by allowing it to set an automatic likelihood threshold. The overall metrics evaluation was carried out using AYNEC-ResTest.

To measure how using different neighborhood sizes influences the performance of CAFE, we evaluated CAFE_1, CAFE_2 and CAFE_3 independently against each other in all proposed metrics. Also, to determine if CAFE is comparable to the previously mentioned state-of-the-art proposals, we used the best-performing variant of CAFE (in terms of better average F1) for comparison against other proposals.

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 8, dropout of 0.1 for all layers, 50 epochs and validation ratio of 10%. When two or more models were to be combined, we joined their results using a

Dataset	Training triples	Test triples	Entities	Relations
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	219,374	53,934	148 (9)

Table 1
Metadata for the datasets we used for evaluation

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 these hyperparameters. We chose them because they provided satisfactory results in our empirical tests and, to the best of our knowledge, there are no proposals for automatically choosing the optimal parameters for a neural network. Our experiments were conducted on a computer equipped with an Intel Core i9-9900K CPU, 64GB of RAM and an Nvidia RTX 2080 Ti GPU.

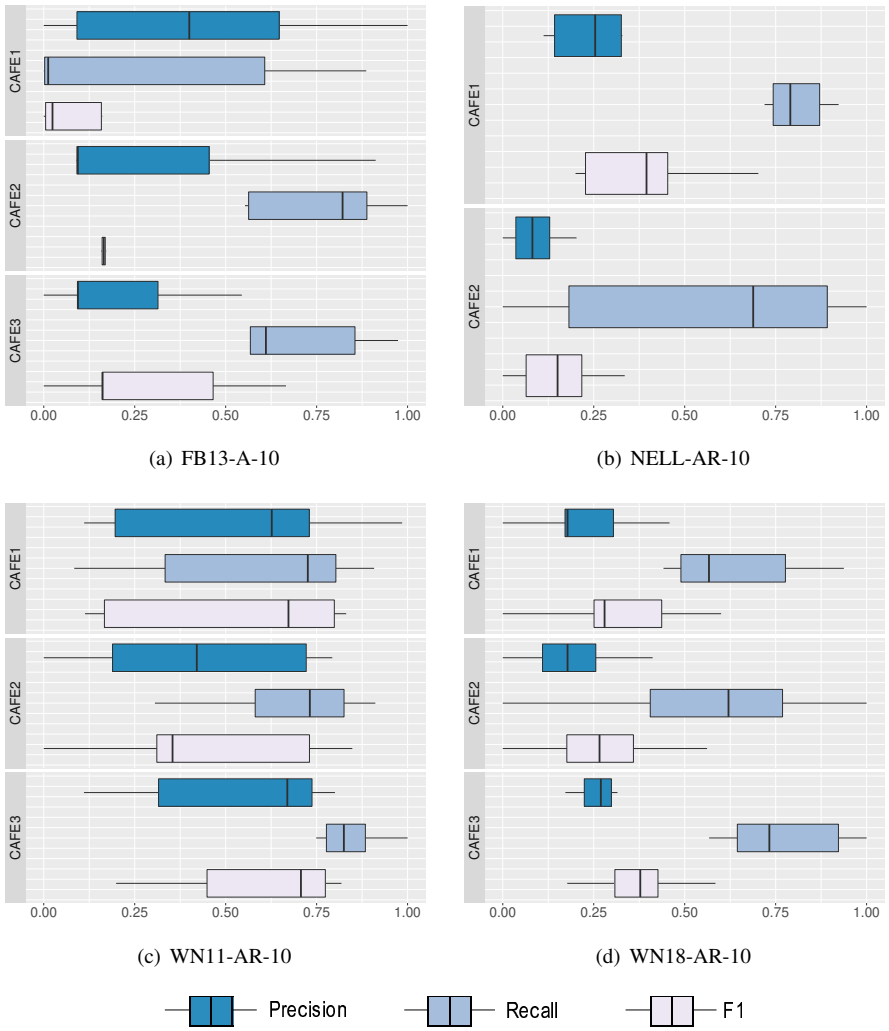
5.3. Results and discussion

In Figure 4 we show the obtained results in terms of precision, recall and F1 for CAFE_1 through CAFE_3. Those obtained when comparing the embedding-based proposals to the best-performing variant of CAFE for every dataset are shown in Figure 5. This was done by selecting the CAFE variant that achieved a higher average F1 in each dataset. Additionally, in Figure 6 we show the detailed results per every relation in all datasets.

Regarding the question of how using different neighborhood sizes affects the performance of CAFE, Fig. 4 shows that performance metrics generally seem to improve for CAFE_2 with respect to CAFE_1, but the same cannot 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 neighborhood subgraphs of size 3 in contrast of those of size 2. Similar remarks can be made for WN11-AR-10 (Fig. 4(c)) and WN18-AR-10 (Fig. 4(d)). 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 performance of CAFE. This effect actually occurs in the case of the NELL-AR-10 dataset, where performance worsens when increasing the size of the neighborhood subgraphs for which we compute features. We believe that a possible explanation for this is that NELL is a noisier dataset [40], and thus taking larger neighborhoods into account significantly increases the amount of noise that our classification model has to deal with, effectively hindering its performance.

Our results also 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 case of FB13-A-10 (Fig. 4(a)), CAFE can achieve a wide range of values for the different metrics, however, it is able to reach higher performance values than the embedding-based proposals under evaluation for all metrics (Fig. 5(a)). In other datasets, such as WN11-AR-10 (Fig. 5(c)) and WN18-AR-10 (Fig. 5(d)), CAFE obtains a more satisfactory performance in a more consistent manner. These results show that CAFE can be more effective than other proposals when presented with a high amount of negative evidence to evaluate.

Figure 6 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, resulting in a typical precision-recall trade-off. We have also observed that, in the specific case of the FB13-A-10 dataset (Fig. 5(a)), one finds an unusually higher number of classification results with a low precision value. We hypothesize that this is due to the dataset in question having relations that are hard to predict using only other information present in it. 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.



(a) FB13-A-10 (b) NELL-AR-10

(c) WN11-AR-10 (d) WN18-AR-10

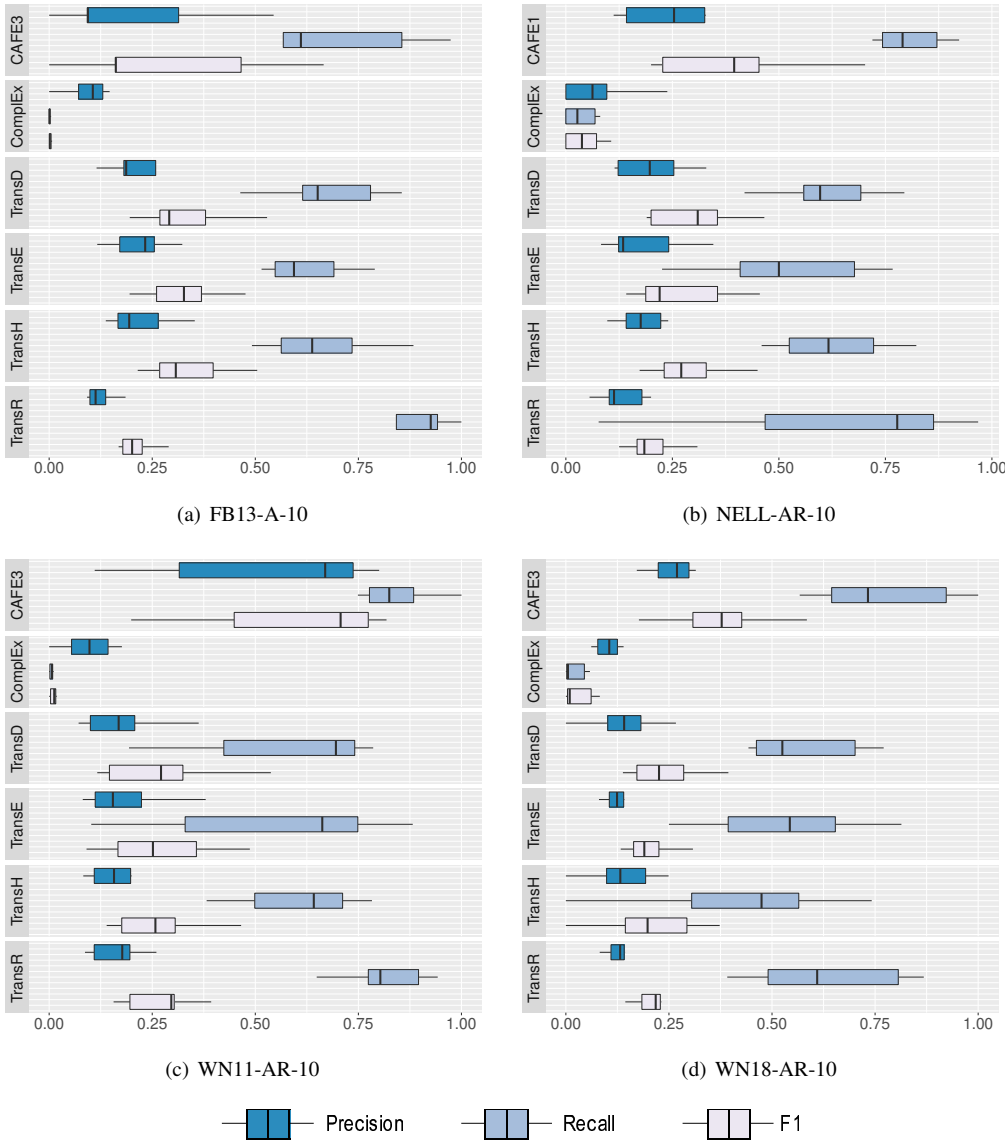
Fig. 4. Performance comparison between the different CAFE variants

5.4. Limitations

Despite CAFE being generally effective, it has limitations. Since the number of features associated with every neighborhood size does not increase linearly, but rather at least quadratically, the computational cost associated with using larger neighborhoods to compute features can escalate quickly. Additionally, CAFE does not work well for relations where useful predictive information cannot always be found in the neighborhoods of the entities in a triple (for example, certain values of 0.00 in Fig. 6). In these cases, even if some information is present in the neighborhoods under consideration, it may be not successfully captured by CAFE due to the high amount of irrelevant data surrounding it. Also, 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), and thus larger neighborhoods of these entities introduce many other entities that are not relevant at all to the triple under evaluation. In these cases, it is up to the user to decide which maximum neighborhood size best caters to their interests.

6. Conclusions

In this paper we introduced CAFE, a proposal for fact checking in Knowledge Graphs that uses a novel



(a) FB13-A-10 (b) NELL-AR-10

(c) WN11-AR-10 (d) WN18-AR-10

Fig. 5. Performance comparison between CAFE and other state-of-the-art proposals

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 performance of CAFE on four well-known Knowledge Graphs, identifying candidate triples that should be added to each KG. To provide a more realistic evaluation scenario, we 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 a performance result that is comparable to state-of-the-art embedding-based techniques, and in many cases even higher. In all datasets under study, CAFE achieves higher average precision values, thus leading to a more trustworthy fact checking 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.

		CAFE$_1$			CAFE$_2$			CAFE$_3$			ComplEx			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
FB13-A-10	CauseOfDeath	**0.40**	0.00	0.00	0.09	0.80	0.16	0.16	0.02	0.04	0.00	0.00	0.00	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	**0.76**	0.70	**0.73**	0.56	0.84	0.68	0.54	**0.86**	0.67	0.12	0.00	0.00	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	**0.65**	0.01	0.03	0.09	**1.00**	0.17	0.00	0.00	0.00	0.11	0.00	0.00	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	0.00	0.00	0.00	0.78	0.03	0.05	0.51	0.97	0.67	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	**1.00**	0.00	0.00	0.09	0.70	0.16	0.09	0.70	0.16	0.05	0.00	0.00	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	**0.81**	0.01	0.02	0.09	0.55	0.16	0.09	0.57	0.16	0.50	0.00	0.00	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	0.09	0.11	0.10	**0.91**	0.02	0.05	0.09	0.90	0.17	0.13	0.00	0.00	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	**0.59**	0.70	**0.64**	0.45	0.82	0.59	0.47	**0.84**	0.61	0.11	0.01	0.02	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	0.09	0.00	0.01	0.09	0.56	0.16	0.09	0.60	0.16	0.07	0.00	0.00	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	**0.42**	0.00	0.01	0.09	**1.00**	0.16	0.09	0.58	0.16	0.04	0.00	0.00	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	0.09	0.61	0.16	0.09	**1.00**	0.17	0.09	0.61	0.16	0.08	0.00	0.00	**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	0.14	0.01	0.02	0.09	0.86	0.17	0.00	0.00	0.00	0.15	0.01	0.01	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	0.28	0.89	0.42	**0.33**	0.89	**0.48**	0.31	**0.90**	0.47	0.10	0.02	0.03	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	**0.25**	0.89	**0.40**	0.08	**1.00**	0.15	-	-	-	0.00	0.00	0.00	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**	0.72	**0.45**	0.04	0.30	0.06	-	-	-	0.09	0.06	0.07	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	0.14	0.49	0.22	0.13	0.69	0.22	-	-	-	**0.33**	0.01	0.03	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	**0.76**	0.87	**0.81**	0.20	**0.97**	0.34	-	-	-	0.00	0.00	0.00	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	**0.13**	**0.79**	**0.23**	0.10	0.77	0.18	-	-	-	0.10	0.08	0.09	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	0.11	**0.92**	**0.20**	0.00	0.00	0.00	-	-	-	0.03	0.08	0.04	**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	0.32	**0.77**	**0.45**	0.04	0.18	0.07	-	-	-	0.00	0.00	0.00	**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	**0.62**	0.81	**0.70**	0.00	0.00	0.00	-	-	-	0.24	0.07	0.11	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	**0.24**	0.74	**0.37**	0.14	**0.89**	0.24	-	-	-	0.06	0.03	0.04	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	0.21	0.12	0.15	0.42	0.31	0.35	**0.61**	0.37	**0.46**	0.05	0.01	0.01	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	**0.18**	0.08	0.11	0.09	0.58	0.15	0.11	**1.00**	**0.20**	0.15	0.04	0.07	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	0.63	0.73	0.67	**0.72**	0.73	**0.73**	0.67	0.75	0.71	0.12	0.00	0.00	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	**0.98**	0.71	**0.82**	0.67	0.80	0.73	0.74	**0.81**	0.77	0.00	0.00	0.00	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	0.11	0.33	0.17	0.21	0.67	0.32	**0.32**	**0.78**	**0.45**	0.18	0.01	0.02	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	0.70	0.75	0.72	0.00	0.00	0.00	**0.71**	**0.82**	**0.76**	0.10	0.01	0.01	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	0.20	**0.91**	0.32	0.19	0.87	0.31	0.21	0.88	0.34	0.08	0.04	0.05	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	0.73	0.88	0.80	**0.79**	**0.91**	**0.85**	0.74	**0.91**	0.82	0.03	0.00	0.00	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	**0.86**	0.80	**0.83**	0.78	0.83	0.80	0.80	0.83	0.81	0.14	0.00	0.00	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	0.27	0.74	0.39	0.00	0.00	0.00	**0.32**	**0.77**	**0.45**	0.14	0.06	0.08	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	0.34	0.82	0.48	0.41	0.88	0.56	**0.43**	**0.91**	**0.58**	0.11	0.00	0.00	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	0.18	0.50	0.26	0.25	0.62	0.35	**0.27**	**0.65**	**0.38**	0.06	0.01	0.01	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	0.17	0.44	0.24	0.18	**0.57**	0.27	**0.23**	**0.57**	**0.32**	0.13	0.00	0.00	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	0.23	0.55	0.32	0.27	0.66	0.39	**0.28**	0.72	**0.41**	0.00	0.00	0.00	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	0.10	0.48	0.17	0.17	0.62	0.27	**0.22**	**0.73**	**0.34**	0.06	0.00	0.00	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	0.18	0.65	**0.28**	0.20	0.24	0.22	**0.27**	0.19	0.22	0.09	0.05	0.06	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	0.00	0.00	0.00	**0.11**	0.06	0.08	0.10	**1.00**	0.18	**0.11**	0.04	0.06	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	**0.44**	0.94	**0.60**	0.11	0.88	0.19	0.17	**1.00**	0.29	0.10	0.50	0.16	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	0.17	0.57	0.27	0.26	0.59	0.37	**0.27**	0.64	**0.38**	0.13	0.00	0.00	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	**0.46**	0.86	**0.60**	0.09	**1.00**	0.16	0.42	0.94	0.58	0.13	0.02	0.03	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. 6. Detailed results for all relations. The highest value for a metric in a relation is marked in **bold**.

Acknowledgments

This work was supported by the Spanish R&D&I program under grant TIN2016-75394-R.

References

[1] C. Bizer, T. Heath and T. Berners-Lee, Linked data - The story so far, *Int. J. Semantic Web Inf. Syst.* **5**(3) (2009), 1–22. doi:10.4018/jswis.2009081901.

[2] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer et al., DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia, *Semantic Web* **6**(2) (2015), 167–195. doi:10.3233/SW-140134.

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

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge and J. Taylor, Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge, in: *SIGMOD*, ACM, 2008, pp. 1247–1250. doi:10.1145/1376616.1376746.

[5] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun and W. Zhang, Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion, in: *SIGKDD*, ACM, 2014, pp. 601–610. doi:10.1145/2623330.2623623.

[6] A. Bordes, S. Chopra and J. Weston, Question Answering with Subgraph Embeddings, in: *EMNLP*, ACL, 2014, pp. 615–620. http://aclweb.org/anthology/D/D14/D14-1067.pdf.

[7] M. Glass and A. Gliozzo, A Dataset for Web-Scale Knowledge Base Population, in: *ESWC*, Springer, 2018, pp. 256–271. doi:10.1007/978-3-319-93417-4_17.

[8] G.A. Miller, WordNet: A Lexical Database for English, *Commun. ACM* **38**(11) (1995), 39–41. doi:10.1145/219717.219748.

[9] D. Ayala, I. Hernández, D. Ruiz and M. Toro, TAPON: A two-phase machine learning approach for semantic labelling, *Knowledge-Based Systems* **163** (2019), 931–943. doi:10.1016/j.knosys.2018.10.017.

[10] S. Neumaier, J. Umbrich, J.X. Parreira and A. Polleres, Multi-level Semantic Labelling of Numerical Values, in: *ISWC*, Vol. 9981, 2016, pp. 428–445. doi:10.1007/978-3-319-46523-4_26.

[11] A. Bordes and E. Gabrilovich, Constructing and Mining Web-scale Knowledge Graphs, in: *SIGKDD*, ACM, 2014, pp. 1967–1967. doi:10.1145/2623330.2630803.

[12] L. Galárraga, C. Teflioudi, K. Hose and F.M. Suchanek, Fast rule mining in ontological knowledge bases with AMIE+, *VLDB J.* **24**(6) (2015), 707–730. doi:10.1007/s00778-015-0394-1.

[13] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, *Semantic Web* **8**(3) (2017), 489–508. doi:10.3233/SW-160218.

[14] A. Neelakantan and M.-W. Chang, Inferring Missing Entity Type Instances for Knowledge Base Completion: New Dataset and Methods, in: *HLT-NAACL*, The Association for Computational Linguistics, 2015, pp. 515–525. https://aclweb.org/anthology/N/N15/N15-1054.pdf.

[15] R. Socher, D. Chen, C.D. Manning and A. Ng, Reasoning with neural tensor networks for knowledge base completion, in: *NIPS*, 2013, pp. 926–934.

[16] L.A. Galárraga, C. Teflioudi, K. Hose and F. Suchanek, AMIE: Association rule mining under incomplete evidence in ontological knowledge bases, in: *WWW*, ACM, 2013, pp. 413–422. doi:10.1145/2488388.2488425.

[17] N. Lao and W.W. Cohen, Relational retrieval using a combination of path-constrained random walks, *Machine learning* **81**(1) (2010), 53–67. doi:10.1007/s10994-010-5205-8.

[18] Y. Lin, Z. Liu, M. Sun, Y. Liu and X. Zhu, Learning entity and relation embeddings for knowledge graph completion., in: *AAAI*, Vol. 15, AAAI Press, 2015, pp. 2181–2187. https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571.

[19] L. Drumond, S. Rendle and L. Schmidt-Thieme, Predicting RDF triples in incomplete knowledge bases with tensor factorization, in: *SAC*, ACM, 2012, pp. 326–331. doi:10.1145/2245276.2245341.

[20] G. Ji, S. He, L. Xu, K. Liu and J. Zhao, Knowledge graph embedding via dynamic mapping matrix, in: *ACL (1)*, The Association for Computer Linguistics, 2015, pp. 687–696. https://aclweb.org/anthology/P/P15/P15-1067.pdf.

[21] A. Borrego, D. Ayala, I. Hernández, C.R. Rivero and D. Ruiz, Generating Rules to Filter Candidate Triples for their Correctness Checking by Knowledge Graph Completion Techniques, in: *K-CAP*, 2019, pp. 115–122. doi:10.1145/3360901.3364418.

[22] P. Lin, Q. Song and Y. Wu, Fact Checking in Knowledge Graphs with Ontological Subgraph Patterns, *Data Science and Engineering* **3** (2018), 341–358. doi:10.1007/s41019-018-0082-4.

[23] M. Nentwig, M. Hartung, A.-C. Ngonga Ngomo and E. Rahm, A survey of current link discovery frameworks, *Semantic Web* **8**(3) (2017), 419–436. doi:10.3233/SW-150210.

[24] M. Gardner and T. Mitchell, Efficient and expressive knowledge base completion using subgraph feature extraction, in: *EMNLP*, The Association for Computational Linguistics, 2015, pp. 1488–1498. https://aclweb.org/anthology/D/D15/D15-1173.pdf.

[25] Z. Wang, J. Zhang, J. Feng and Z. Chen, Knowledge Graph Embedding by Translating on Hyperplanes, in: *AAAI*, Vol. 14, AAAI Press, 2014, pp. 1112–1119. https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531.

[26] K. Kolthoff and A. Dutta, Semantic relation composition in large scale knowledge bases, in: *LD4IE@ISWC*, Vol. 1467, CEUR-WS.org, 2015, pp. 34–47. http://ceur-ws.org/Vol-1467/LD4IE2015_Kolthoff.pdf.

[27] B. Shi and T. Weninger, Open-World Knowledge Graph Completion, in: *AAAI*, 2018, pp. 1957–1964. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16055.

[28] B. Shi and T. Weninger, ProjE: Embedding Projection for Knowledge Graph Completion, in: *AAAI*, 2017, pp. 1236–1242. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14279.

[29] M. Nickel, V. Tresp and H.-P. Kriegel, Factorizing YAGO: scalable machine learning for linked data, in: *WWW*, ACM, 2012, pp. 271–280. doi:10.1145/2187836.2187874.

[30] Z. Wang, Z. Ren, C. He, P. Zhang and Y. Hu, Robust embedding with multi-level structures for link prediction, in: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, AAAI Press, 2019, pp. 5240–5246.

[31] X. Han, S. Cao, X. Lv, Y. Lin, Z. Liu, M. Sun and J. Li, OpenKE: An Open Toolkit for Knowledge Embedding, in: *EMNLP*, 2018, pp. 139–144.

[32] S. Mazumder and B. Liu, Context-aware Path Ranking for Knowledge Base Completion, in: *IJCAI*, AAAI Press, 2017, pp. 1195–1201. doi:10.24963/ijcai.2017/166.

[33] W. Xiong, T. Hoang and W.Y. Wang, Deeppath: A reinforcement learning method for knowledge graph reasoning, in: *EMNLP*, 2017.

[34] Y. Shen, D. Wen, Y. Li, N. Du, H.-t. Zheng and M. Yang, Path-based attribute-aware representation learning for relation prediction, in: *Proceedings of the 2019 SIAM International Conference on Data Mining*, SIAM, 2019, pp. 639–647.

[35] V.T. Ho, D. Stepanova, M.H. Gad-Elrab, E. Kharlamov and G. Weikum, Learning rules from incomplete kgs using embeddings, in: *The 17th International Semantic Web Conference*, ceur. ws. org, 2018.

[36] D. Ayala, A. Borrego, I. Hernández, C.R. Rivero and D. Ruiz, AYNEC: All You Need for Evaluating Completion Techniques in Knowledge Graphs, in: *ESWC*, 2019, pp. 397–411. doi:10.1007/978-3-030-21348-0_26.

[37] A. Bordes, X. Glorot, J. Weston and Y. Bengio, A semantic matching energy function for learning with multi-relational data - Application to word-sense disambiguation, *Machine Learning* **94**(2) (2014), 233–259. doi:10.1007/s10994-013-5363-6.

[38] A. Bordes, N. Usunier, A. García-Durán, J. Weston and O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: *NIPS*, 2013, pp. 2787–2795.

[39] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier and G. Bouchard, Complex Embeddings for Simple Link Prediction, in: *ICML*, Vol. 48, 2016, pp. 2071–2080. http://proceedings.mlr.press/v48/trouillon16.html.

[40] H. Paulheim and C. Bizer, Improving the quality of linked data using statistical distributions, *Int. J. Semantic Web Inf. Syst.* **10**(2) (2014), 63–86. doi:10.4018/ijswis.2014040104.