Generating Rules to Filter Candidate Triples for their Correctness Checking by Knowledge Graph Completion Techniques

Agustín Borrego
borrego@us.es
University of Seville
Seville, Spain

Daniel Ayala
dayala1@us.es
University of Seville
Seville, Spain

Inma Hernández
inmahernandez@us.es
University of Seville
Seville, Spain

Carlos R. Rivero
crr@cs.rit.edu
Rochester Institute of Technology
Rochester, NY, USA

David Ruiz
druiz@us.es
University of Seville
Seville, Spain

ABSTRACT

Knowledge Graphs (KGs) contain large amounts of structured information. Due to their inherent incompleteness, a process known as KG completion is often carried out to find the missing triples in a KG, usually by training a fact checking model that is able to discern between correct and incorrect knowledge. After the fact checking model has been trained and evaluated, it has to be applied to a set of candidate triples, and those that are considered correct are added to the KG as new knowledge. However, this process needs a set of candidate triples of a reasonable size that represents possible new knowledge, in order to be evaluated by the fact checking task and, if considered to be correct, added to the KG, enriching it. Current approaches for selecting candidate triples for their correctness checking either use the full set possible missing candidate triples (and thus provide no filtering) or apply very basic rules to filter out unlikely candidates, which may have a negative effect on the completion performance as very few candidate triples are filtered out. In this paper we present CHAI, a method for producing more complex rules that are able to filter candidate triples by combining a set of criteria to optimize a fitness function. Our experiments show that CHAI is able to generate rules that, when applied, yield smaller candidate sets than similar proposals while still including promising candidate triples.
•

KEYWORDS

Knowledge Graphs, Knowledge Graph completion, Candidate filtering

1 INTRODUCTION

Over the recent years, Knowledge Graphs (KGs) have enjoyed increasing popularity as a means to store and represent information pertaining the relations that exist between entities, thus representing real-world facts. Consequently, large-scale KGs such as DBpedia [16], NELL [21], Freebase [3] or the Google Knowledge Vault [8] have become widely used for tasks such as question answering [4].

Large-scale KGs are generally constructed through unsupervised processes which extract semi-structured [13, 29] or non-structured [8, 21] information, and then semantize the extracted information [2] in order to store it in the form of entities and relations between these entities, which are known as triples. Thus, it is likely for the resulting KG to be incomplete, either because the information extraction step failed to extract a relevant piece or information, or because it was altogether missing from the original source [5]. As a result of this incompleteness, KGs are governed by the Open World Assumption, which dictates that a piece of information does not necessarily have to be false if it is not present in a KG. This results in the need to refine existing KGs so as to complete the knowledge that they contain, a task commonly known as KG Completion [24]. Figure 1 illustrates the typical workflow of a KG Completion process.

A KG completion process consists in identifying missing triples that should be added to it, since they represent correct knowledge. Usually, it involves several tasks: first, it is necessary to learn a classification model to predict whether a candidate triple should be considered correct or not regarding a particular KG, a task commonly known as fact checking [17]. The creation of a fact checking model usually requires to pre-process the KG, by splitting it into training and testing sub-graphs, which are then be used to train and evaluate the fact checking model. A variable amount of negative evidence is also commonly added to these splits, because KGs

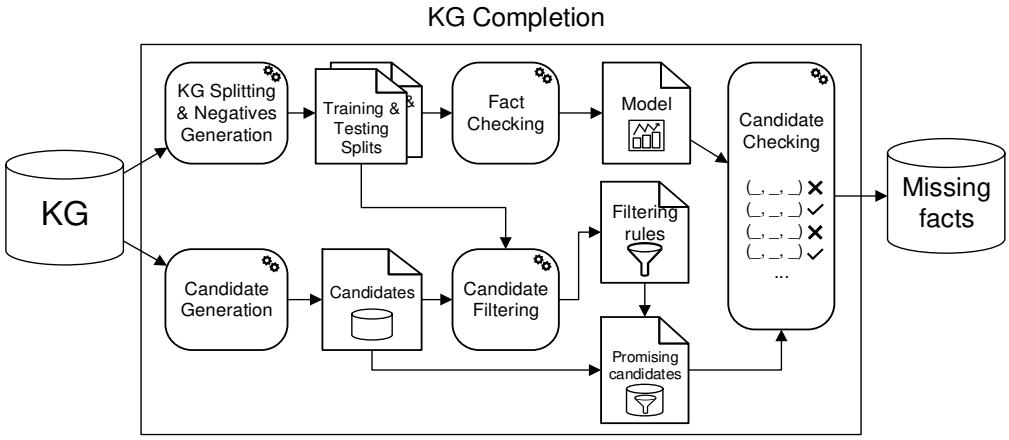


Figure 1: Knowledge Graph Completion workflow

only contain positive examples [24]. Note that this is merely a simplified description of the pre-processing of a KG for illustration purposes, and that certain proposals may require further splitting, for example, to fine-tune a set of hyperparameters for a neural network [26].

Besides creating the fact checking model, it is necessary to generate a set of promising candidates, which are triples that may be correct but that are not present in the KG. This requires the generation of an initial, exhaustive set of candidates, which can be produced by applying brute force, i.e., producing all possible triples not contained in the KG, regardless of whether they are promising or not. Due to the size of this initial set of candidates, it has to be subsequently filtered to preemptively remove many unlikely candidates while retaining as many promising ones as possible. Usually, this is accomplished by generating a number of filtering rules, and applying these rules to the candidate set to produce the set of promising candidates, a task called candidate filtering [25].

Finally, the fact checking model is applied to the set of promising candidates, which results in the set of missing triples that are considered correct and that may be added to the KG.

In the literature, we can find a number of fact checking proposals that produce models to classify a triple as correct or incorrect, following approaches such as rule-based reasoning [10, 14, 19, 23], neural tensor networks [26], embedded spaces [7, 25, 27], tensor factorization [22], concepts of nearest neighbors [9] and random walks [11, 18]. However, when it comes to candidate filtering, all current approaches are encompassed within a specific fact checking technique, and they are very basic. Most of the existing completion proposals either rely on a handmade or provided set of promising candidates, or include a baseline filtering technique that only works well for certain relations [25].

In this paper we present CHAI, a method for generating rules that are able to filter candidate triples in the context of the KG completion process by combining a number of criteria in such a way that it optimizes a given fitness function. CHAI produces rules that can be applied on the initial set of candidates and produce a reduced set that contains only the promising candidate triples. Then, this set can be passed on to any fact checking technique to check the correctness of each promising candidate and identify correct triples that complete the KG. CHAI rules are based on different criteria that take the internal features of the KG into account, such as the domains and ranges of every relation in the KG, in addition to the distances between its entities.

The advantages of our proposal are that it does not require any additional intervention from the user or external sources of information to create the rules, and that it significantly reduces the size of the candidate set while maximizing the number of promising candidates that are selected by the produced rules. This, as a result, allows for a more effective application of the fact checking techniques in KG completion contexts. The experimental results show that it outperforms other proposals in the state of the art of candidate filtering in terms of coverage and reduction rate. CHAI is able to achieve good performance when dealing with all relationships in every KG under study, which proves that it is a generic and effective method, suitable for real-world contexts. The sets of promising candidate triples that are produced by the application of our rules are significantly smaller than the initial set composed of every possible missing triple, while still including up to 99% of the correct facts thanks to the usage of a fitness function that aims to optimize both coverage and reduction rate.

The rest of this paper is as follows: Section 2 presents the related work in filtering candidate triples for KG completion. Section 3 describes our proposal for candidate filtering. Section 4 reports on our experimental results; finally, Section 5 contains our conclusions.

2 RELATED WORK

Nowadays, there exist a number of proposals to select candidate triples for KG completion.

Wei et al. [28] use TransE [7] as an embedding-based model to produce sets of candidate entities. TransE obtains embeddings for entities and relations in the KG that maintain semantic similarities, i.e., ensuring that the embedding of entities that are related by means of a relation are close in the vector space. They calculate similarity scores for each entity using their embedded representations and sort them in descending order of similarity, selecting the top N entities as candidates and producing the corresponding candidate

triples, leaving the rest out. Also, due to the way they compute similarities, the entire set of entities in the KG has to be considered every time the user desires to produce a candidate triple.

Gardner et al. [12] propose a random walk-based approach for selecting candidates. Their proposal, which is an evolution of the Path Ranking Algorithm (PRA) by Lao and Cohen [15], finds possible paths that connect a source entity with a target entity and ranks these paths by frequency, keeping the top N paths as relevant. In PRA, features are calculated only for paths in the top-N selection; meanwhile, in the evolved version by Gardner et al. [12], features are also calculated for paths that are similar to the ones in the top-N selection, according to a similarity function. Note that performing random walks means that these proposals may not cover all relevant paths, which may have an impact on their performance. Also, the value of N must be carefully selected since it determines the amount of paths that are taken into account, which impacts their performance.

Finally, Shi and Weninger [25] use a heuristic to select candidate entities for a given relation. The heuristic selects all entities that appear as the target of a given relation in the training split as possible candidates for said relation. The performance of this technique depends largely on the distribution of the dataset between training and testing splits, since an entity that is not found as the target of a given relation in the training set would be discarded.

In general, it is computationally expensive to train embedding-based techniques, and those based on random walk may miss relevant information. The only proposal that is similar enough to ours, by Shi and Weninger, uses a simple heuristic to filter out unlikely candidate triples, but this heuristic may not be fit for all kinds of relations, especially for those that have a domain or range that is composed by several types of entities. Our proposal, CHAI, produces rules that only have to be applied once to produce an entire set of candidates. They are also entirely deterministic, and thus not prone to missing candidates in comparison to a random walk-based proposal. Additionally, we propose a set of criteria for building rules that are broader than only considering those whose entities appear as the target of a relation under study, adding the possibility to select candidate triples based on the domain and ranges of all relations in a KG, in addition to distance-based criteria, thus providing a more generic and effective method for candidate filtering.

3 OUR PROPOSAL

In this section, we introduce our proposal for generating rules that filter candidates in the context of KG completion. We first introduce some preliminary concepts that are needed to understand our proposal, and then we introduce an algorithm for generating suitable rules to select possible candidates. For the sake of the example, we provide a visual representation of a possible KG in Figure 2, which is separated into a training and a testing split in a random manner.

3.1 Preliminaries

We now define the notation that is used in the following sections.

Definition 3.1. **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 between a source entity s and a target entity t, where $s, t \in \mathcal{E}$ and $r \in \mathcal{R}$. We denote triples as (s, r, t). Other authors also refer to triples as $(subject, predicate, object)$.

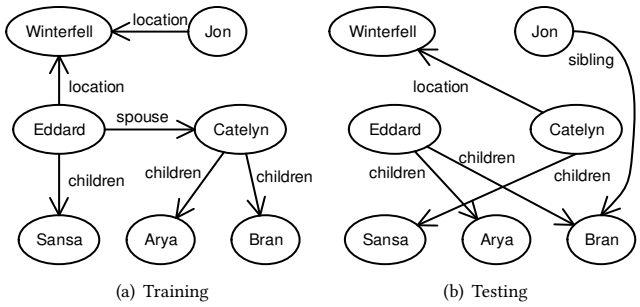


Figure 2: An example of a KG with its triples randomly split into training and testing.

Definition 3.2. **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 directed graph defined by the 3-tuple $(\mathcal{E}, \mathcal{R}, \mathcal{T})$. We denote a Knowledge Graph as KG.

Definition 3.3. **Candidate:** Let KG $= (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph, let $s \in \mathcal{E}$ be an entity and let $r \in \mathcal{R}$ be a relation in KG. We define a candidate as a triple (s, r, t) has a chance of representing real-world knowledge, even if it does not exist in $\mathcal{T}$.

In the KG depicted in Figure 2, a candidate triple is (*Eddard, children, Jon*).

Definition 3.4. **Path:** Let KG $= (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph, and let $s, t \in \mathcal{E}$ be two entities in KG. We define a path p of length n between s and t as a sequence of triples of the form $\langle (e_i, r_i, e_{i+1}) \rangle$ 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 $path(\text{KG}, s, t, r_1, r_2, \ldots, r_n)$, or $path_n(\text{KG}, s, t)$ for short.

In Figure 2(a), a possible path of length 2 between the entities *Eddard* and *Arya* is $\langle$(*Eddard, spouse, Catelyn*), (*Catelyn, children, Arya*)$\rangle$.

Definition 3.5. **Distance between entities:** Let KG $= (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph, and let $s, t \in \mathcal{E}$ be two entities in KG. We define the distance between s and t as the length of the shortest path that exists between s and t in KG, i.e., $|path_n(\text{KG}, s, t)|$ such that $\nexists\, path_i(\text{KG}, s, t) \mid i < n$. We denote the distance between s and t as $dist(\text{KG}, s, t)$.

In Figure 2(a), the distance between the entities *Eddard* and *Arya* is 2, while in Figure 2(b) their distance is 1.

Definition 3.6. **Fitness function:** Let KG $= (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph, let C be a set of candidates and let C' be a set of filtered candidates, with $C' \subseteq C$. We define fitness as a function $fitness(\text{KG}, C, C') \rightarrow \mathbb{R}$ that assigns a score to the filtered set of candidates, with respect to the original set of candidates and KG.

3.2 Proposed criteria and rules

We propose a set of criteria for filtering candidates in the context of a Knowledge Graph $(\mathcal{E}, \mathcal{R}, \mathcal{T})$. Note that a criterion is a predicate that assigns a True/False binary label to a candidate triple. Each criterion is devised following a different approach, and therefore the sets of candidates that are selected by each of them are relatively disjoint, although they might overlap to some extent (we discuss these implications in Section 4.3). These criteria are as follows:

Existing source entity and relation: Let (s, r, t) be a triple in $\mathcal{T}$. This criterion selects all candidates whose source entity and relation appear as such for some triple in $\mathcal{T}$:

$$exists_{\text{KG}}((s, r, t)) \Leftrightarrow \exists\, e \in \mathcal{E} \mid (s, r, e) \in \mathcal{T}$$

Target is in the domain of a relation $rel \in \mathcal{R}$: Let (s, r, t) be a triple in $\mathcal{T}$. This criterion selects all candidates whose target entity appears at least once as the source in a triple that has rel as its relation:

$$dom_{\text{KG},rel}((s, r, t)) \Leftrightarrow \exists\, e \in \mathcal{E} \mid (t, rel, e) \in \mathcal{T}$$

Target is in the range of a relation $rel \in \mathcal{R}$: Let (s, r, t) be a triple in $\mathcal{T}$. This criterion selects all candidates whose target entity appears at least once as the target in a triple that has rel as its relation:

$$ran_{\text{KG},rel}((s, r, t)) \Leftrightarrow \exists\, e \in \mathcal{E} \mid (e, rel, t) \in \mathcal{T}$$

Entities are within distance i: Let (s, r, t) be a triple in $\mathcal{T}$. This criterion selects all candidates whose source and target entities have a distance between them that is at most i:

$$distance_{\text{KG},i}((s, r, t)) \Leftrightarrow dist(\text{KG}, s, t) \leq i$$

In Figure 2(a), $exists_{\text{KG}}((Eddard,\ children,\ Catelyn)) = True$, $dom_{\text{KG},children}((Eddard,\ spouse,\ Catelyn)) = True$, $ran_{\text{KG},children}((Eddard,\ spouse,\ Arya)) = True$, and $distance_{\text{KG},1}((Eddard,\ spouse,\ Catelyn)) = True$.

The rationale behind these criteria is manifold. First, $exists_{\text{KG}}$ ensures that only reasonable combinations of source entities and relations are present in the candidate triples, as it avoids scenarios such as a geographical place being the father of a person. Second, $dom_{\text{KG},rel}$ and $ran_{\text{KG},rel}$ are for further filtering triples based on their target entities: for instance, it may be reasonable to expect that good candidate triples for the relation *children* should have a target entity that represents a person, and thus these criteria allows CHAI to select such target entities. Even if the types of the entities are not included as information in the KG, it is reasonable to assume that entities that appear in the domain or a range of a relation usually belong to one or more specific types. By considering triples that include these entities as their targets, CHAI can provide candidates with entities that most likely belong to a certain type. Finally, $distance_{\text{KG},i}$ covers the assumption that a good candidate triple should be such that its source and target entities are close to each other in the Knowledge Graph.

Additionally, we define a rule as a conjunction of the criterion $exists_{\text{KG}}$ and the disjunction of other criteria.

Definition 3.7. **Rule:** Let KG be a Knowledge Graph, let C be a set of candidates, and let $c_1, c_2, \ldots, c_n$ be a number of criteria other than the *exists* criterion. We define a rule as a conjunction of the criterion $exists_{\text{KG}}$ and the disjunction of one or more criteria, i.e., $exists_{\text{KG}} \wedge (c_1 \vee c_2 \vee \ldots \vee c_n)$.

By enforcing the $exists_{\text{KG}}$ criterion on all candidate triples, we make sure that the resulting set of candidates has a lower number of incorrect or noisy candidates, as all of them have a combination of source entity and relation that already exists in the original KG while still allowing all possible target entities: for example, the candidate triple *(Eddard, children, Jon)* would pass the $exists_{\text{KG}}$ criterion, whereas *(Winterfell, spouse, Jon)* would not. In addition, the disjunction of criteria present in the rule allows for more flexibility: longer rules with more criteria are less strict, and thus produce more candidates by combining different criteria.

3.3 Proposed algorithm

The algorithm that we proposed for generating rules for candidate filtering is shown in Algorithm 1. It receives the set of candidates to be filtered, the original KG in the form of a training and a testing split and a relation as input, and it outputs the generated rule for the relation and the filtered set of candidates that is produced by applying the generated rule.

First, the input set of candidates is narrowed down to those that include the relation for which CHAI is being applied. Subsequently, a rule that contains only the $exists_{\text{KG}}$ criterion is generated, and the set of candidates that results from applying it is obtained, which will be further refined by adding more criteria to the rule.

Then, a set of criteria is instantiated, which contains the *dom* and *ran* criteria for every possible relation in the KG, as well as the *distance* criterion for up to a certain maximum distance. These criteria are the ones that will be used for building the rule, however, not every criterion in the set has necessarily to be added to the rule.

Following the previous step, a fitness value is computed for each criterion, by applying a certain fitness function on the set of candidates that are selected by that criterion. The previous set of criteria is then sorted in descending order of the fitness value that is obtained in this manner.

Finally, these ordered criteria are added in an iterative fashion to the rule under generation. Every time a criterion is added, the resulting set of filtered candidates is computed, and the fitness value associated to the rule is updated. This process is repeated until the fitness value exceeds a given threshold or the set of available criteria is depleted, at which point no more criteria will be added to the rule. Once this process ends, the generated rule is returned.

In Figure 3, we present an example on the process of generating a rule for the relation *children*, with a threshold $\theta = 0.95$. Through the process, different criteria are iteratively added to the rule, and the fitness value is included for every step. Once the fitness value meets or exceeds the threshold, the process ends and the rule is returned. In this case, selecting candidate triples whose target entities represent people would provide a good result.

Since an integral KG completion process involves every relation in the KG, CHAI should be applied once for each relation in the KG, in order to produce the complete set of rules and suitable candidates for KG Completion. This results in a total set whose size is significantly smaller than that of the input set of candidates, while still containing as many suitable candidates as possible.

Algorithm 1: CHAI

Input: $\text{KG}_{trn} = (\mathcal{E}, \mathcal{R}, \mathcal{T}_{trn})$: Training split of the KG
$\text{KG}_{tst} = (\mathcal{E}, \mathcal{R}, \mathcal{T}_{tst})$: Testing split of the KG
candidates : Set of potential candidates to be filtered
rel : Selected relation in $\mathcal{R}$
fitness : Fitness function
N : Maximum distance value for distance-based criteria
θ : Fitness threshold value

Output: *rule* : Generated rule

```
function CHAI(KG_trn, KG_tst, candidates, rel, fitness, N, θ)
    // Select the candidates in which the relation rel appears
    rc ← {(s, r, t) ∈ candidates | r = rel}
    // Initialize the rule to initially contain only exists_KG_trn
    rule ← exists_KG_trn
    // Obtain a set of initially filtered candidates by applying
    // exists_KG_trn
    fc ← apply exists_KG_trn to rc
    // Add all possible criteria to the set of available criteria
    // using the training split of the KG
    criteria ← ∅
    forall r ∈ R, i ∈ [1..N] do
        criteria ← criteria ∪ {dom_KG_trn,r, ran_KG_trn,r, distance_KG_trn,i}
    // Sort all criteria by the fitness value obtained on the set of
    // filtered candidates they generate, using the testing split
    criteria ← sort criteria by fitness(KG_tst, fc, apply criteria to fc)
    forall criterion ∈ criteria do
        // Apply the rule to obtain a set of filtered candidates
        selected_candidates ← apply rule to fc
        // Compute the fitness value of the previous set
        // using the testing split
        if fitness(KG_tst, fc, selected_candidates) < θ then
            // Add current criterion if the threshold is not met
            rule ← add criterion to rule
    return rule
```

Rule	Fitness value	Meets threshold?
$exists_{\text{KG}} \wedge$	-	-
$(\ ran_{\text{KG},spouse} \vee$	0.80	No
$ran_{\text{KG},children} \vee$	0.92	No
$ran_{\text{KG},parent}\ )$	0.96	Yes

Figure 3: An example rule being built for the relation *children*

4 EXPERIMENTAL STUDY

In this section we present the experimental results that confirm that CHAI is effective in practice. First, we introduce the experimental setting. Then, we present the results of applying CHAI on several well-known Knowledge Graphs, comparing them against those of a state-of-the-art baseline technique by Shi and Weninger [25]. Finally, we discuss these results.

4.1 Setup and datasets

We evaluated CHAI using four different Knowledge Graphs that are openly available and commonly used for the task of KG completion: FB13 [26], WN18 [6] (which are subsets of Freebase [3] and Wordnet [20], respectively), a subset of NELL introduced by Gardner and Mitchell [11], and EPSRC[1], which contains information about the grants provided by the Engineering and Physical Sciences Research Council of the United Kingdom. All of these datasets were obtained from the publicly available AYNEC-DataGen tool [1], and an overview of their metadata can be found in Table 1. We used CHAI to generate rules for every relation in every dataset, except for the NELL dataset, in which we focused on the same subset of 10 relations as Gardner and Mitchell [11] due to the high number of total relations. All experiments were conducted on a computer with 32GB of RAM and an Intel Core i9-9900K CPU.

Table 1: Metadata for the KGs we used for evaluation

KG	Training triples	Test triples	Relations
FB13	285,208	78,490	13
WN18	117,160	58,564	18
NELL	201,870	13,491	519 (10)
EPSRC	341,372	85,337	20

The results of the approach followed by Shi and Weninger [25] were used as a baseline. Their proposal consists in generating candidate triples by altering the target entities of the triples already present in the KG, and replacing them by all entities that can be found in the range of the relation present in the triple. This is equivalent to applying only the $ran_{\text{KG},r}$ criterion, where r is the relation for which CHAI is being applied. Therefore, we obtained the baseline results by modifying CHAI to include only that criterion.

It is important to note that we do not compare CHAI to the fact checking phase of [25] because CHAI is not a fact checking technique, we instead compare it to the approach they use to generate candidate triples for their evaluation. Thus, we use measures that evaluate the aptness of the sets of candidates that CHAI generates for their further correctness checking, such as coverage and reduction rate.

4.2 Evaluation parameters

To conduct our experiments, we set the distance threshold N for the *distance* criterion to 4. This value was chosen empirically, aiming to allow for a threshold as high as possible while still being reasonable in terms of computation time. Additionally, these distances were computed on a partially undirected version of the KGs. This was

[1] http://epsrc.rkbexplorer.com

done to fully exploit the highly relational nature of KGs, while still not allowing paths that would be connected by means of entities with a very high in-degree such as genders or nationalities.

The training and testing splits of the KGs were already provided by the datasets that we used, and thus we provided CHAI with these splits as is. We evaluated CHAI using a fitness function that combines reduction rate and coverage using their harmonic mean, as shown in Eq. 1. The θ threshold value required by the algorithm was set to 0.99, so as to allow CHAI to find highly satisfactory rules, and to study the evolution of the coverage and reduction rate of said rules if they keep growing in size without meeting the threshold.

Let C be a candidates set, and $C' \subseteq C$ a set of filtered candidates:

$$fitness(\text{KG}, C, C') = \frac{2 \cdot rr(C, C') \cdot coverage(\text{KG}, C')}{rr(C, C') + coverage(\text{KG}, C')}, \text{ where} \quad (1)$$

$$rr(C, C') = 1 - \frac{|C'|}{|C|}$$

$$coverage(\text{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T}), C') = \frac{|C' \cap \mathcal{T}|}{|\mathcal{T}|}$$

This fitness function was devised under the following rationale: focusing only on coverage would result in very long rules that allow as many candidates as possible, however, this would not be desirable as we aim to reduce the size of the set of candidates, to avoid having to evaluate low-quality candidates. Conversely, focusing only on reduction rate would yield very short (and thus more restrictive) rules. As a consequence, this fitness function achieves a compromise between reduction rate and coverage, and allows for more flexibility in the lengths of the rules in contrast to focusing only on one objective. To illustrate this difference, we have tested CHAI using three different fitness functions: only coverage, only reduction rate and the proposed harmonic mean. The results achieved by every fitness function are shown in Figure 5.

Our implementation of CHAI with the specified parameters and KGs, along with more detailed visual representations of the results of our experiments with respect to individual relations, is freely available for public use[2].

4.3 Results and discussion

In the following, we present the results achieved by CHAI on the datasets under evaluation and the conclusions we draw from them. For the sake of brevity, we present general results for all datasets, however a detailed report which includes results for every single relation in all datasets can be found alongside our implementation.

Figure 4 reports on the evolution of the coverage and reduction rate for all datasets under study as rules grow in size, while Figure 5 display the values for the coverage and reduction rate for every iteration in all datasets as points in a 2-dimensional space. Finally, Table 2 provides an overview on the average maximum coverage and reduction rate that CHAI achieves for the relations in all KGs under study. This Table also includes the average coverage and reduction rate values achieved by the proposal of Shi and Weninger [25], which was denoted as "baseline" for brevity.

[2]https://github.com/tdg-seville/CHAI

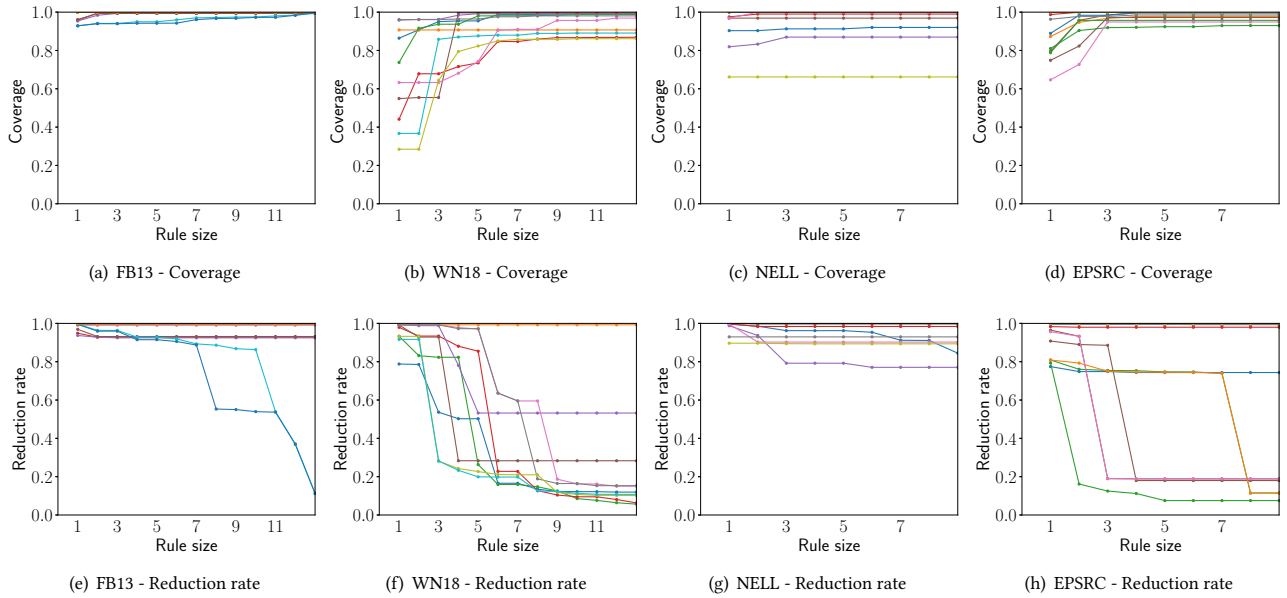
Table 2: Summary of the average maximum achieved coverage and reduction rate for the KGs under evaluation. The average values are in bold, while the 95% confidence interval is shown between parentheses.

KG	Avg. max. coverage (CHAI)	Avg. coverage (baseline)	Avg. max. RR (CHAI)	Avg. RR (baseline)
FB13	**0.92** (0.76-1.00)	**0.78** (0.58-0.99)	**0.91** (0.76-1.00)	**0.91** (0.76-1.00)
WN18	**0.94** (0.89-0.99)	**0.49** (0.26-0.72)	**0.97** (0.93-1.00)	**0.93** (0.85-1.00)
NELL	**0.89** (0.78-1.00)	**0.53** (0.26-0.80)	**0.97** (0.95-1.00)	**0.99** (0.99-1.00)
EPSRC	**0.99** (0.98-1.00)	**0.82** (0.68-0.97)	**0.95** (0.91-0.99)	**0.95** (0.92-0.99)

These results allow us to distinguish between two types of relations: those for which a high coverage value is obtained with a very short rule, and those for which the coverage starts at a lower value and increments as rules grow in size, as shown in Figure 4. We consider the former type of relations to be categorical, as they have a range of possible target entities that is relatively small: for example, the entities that are targets for the relation *location* are unlikely to appear as the target for any other relation, and thus using the entities that are targets for *location* as possible candidates for locations yields a very good result. On the other hand, relations that are non-categorical have a much wider range of possible candidates: in the case of the relation *children*, an entity may produce a good candidate even if it does not appear as the target of *children* (for example, they may appear in the relation *sibling*).

This conclusion is reinforced by the results shown in Figure 5, where there are groups of iterations in the top-right corner (the area of both high reduction rate and high coverage), which are obtained for categorical relations with a small subset of possible targets, while a different group of iterations show more scattered results in the top area, denoting that in order to achieve a high coverage, a bigger set of candidates must be used (non-categorical relations). Additionally, the results shown in this Figure lead us to the conclusion that using only reduction rate as the fitness function results in a very poor coverage, as the algorithm stops after having selected only one criterion that allows a very small number of candidates. On the contrary, using only coverage as the fitness function provides better results, but with a clear tendency towards prioritizing coverage at the expense of a lower reduction rate, while using the harmonic mean yields more balanced results.

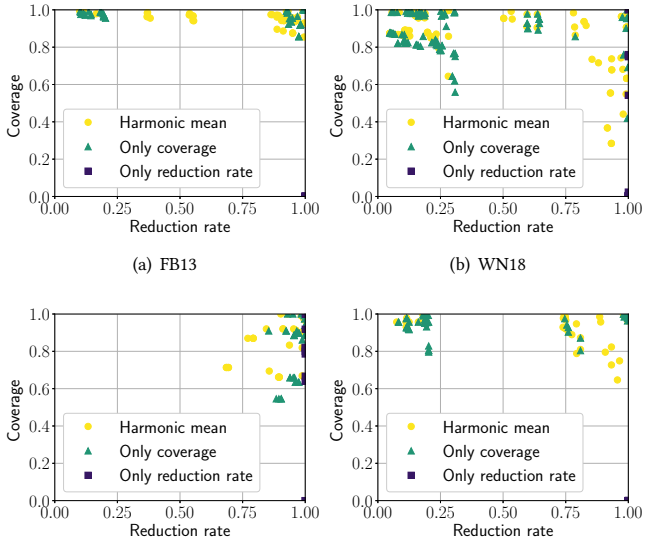
In the case of non-categorical relations, there exists a trade-off between coverage and reduction rate. This is to be expected, since rules are disjunctions of criteria and thus rules that comprise more criteria are more likely to filter out less candidates, increasing coverage but decreasing the reduction rate. In these cases, it is up to the user to decide whether they are interested in achieving a very high coverage with a lower reduction rate, or a higher reduction rate with a usually lower coverage. For this kind of relations, distance-based criteria are generally useful: for example, one's parents or spouse are usually found within a short distance in the KG. In



(a) FB13 - Coverage (b) WN18 - Coverage (c) NELL - Coverage (d) EPSRC - Coverage

(e) FB13 - Reduction rate (f) WN18 - Reduction rate (g) NELL - Reduction rate (h) EPSRC - Reduction rate

Figure 4: Evolution for the coverage (top) and reduction rate (bottom) values for all relations in every KG and different rule sizes. Each line represents a relation.



(a) FB13 (b) WN18

(c) NELL (d) EPSRC

Figure 5: Reduction rate (x-axis) and coverage (y-axis) obtained throughout the different iterations for all KGs using different fitness functions.

categorical relations, however, one is able to obtain both a high reduction rate and a high coverage, because the entities that are suitable targets for said relations can easily be found by analyzing the domains and ranges of the relation in question or other relations.

Regarding the reduction rate, one has to note that it is computed with regard to the set of candidates that are selected by the *exists* criterion, and not with respect to the cartesian product of all possible candidates. This results in reduction rates that are in all cases lower than those that would be obtained in the latter case, though more evenly spread between 0 and 1. Given that high reduction rates are still achieved, we do not consider this to be a problem, but rather an adequate way of measuring how many candidates are selected with respect to an initial set of filtered candidates, with many unlikely candidates already filtered out.

Finally, it is worth noting that CHAI consistently manages to achieve high coverage values for all datasets under study, as it can be seen in Figure 4. These values start to converge with rules composed of approximately five criteria, and thus we find that the criteria that are ranked higher by means of the proposed fitness function (Eq. 1) are indeed successful in allowing promising candidates to pass the filter. When comparing CHAI to the baseline approach proposed by Shi and Weninger [25], CHAI achieves much higher values of coverage while still being able to obtain similar reduction rates, as shown in Table 2. The values for CHAI shown in said Table refer to the average maximum values that can be achieved, and thus our proposal is more versatile as it allows the user to prioritize a higher coverage by using longer rules, or a higher reduction rate by using shorter rules. CHAI works well for all kinds of relations due to the versatility of the criteria it uses, while the proposal by Shi and Weninger [25] is not able to deal as effectively with non-categorical relations in terms of coverage, hindering its overall performance.

4.4 Limitations

While CHAI obtains satisfactory results, it is not without limitations. Perhaps the most important one would arise in the case of a KG

with a very high number of total relations, because the amount of domain and range-based criteria would be equally high. In this case, the fitness function would have to be computed for every criterion in order to sort them by decreasing fitness value, resulting in a potentially high computational cost. Besides, CHAI may not work as well in very sparse KGs where all or most relations share the same entities in their domains and ranges, because the distance-based criterion would need a much higher threshold due to the sparsity of possible paths, and the domain and range-based ones would not prove useful.

5 CONCLUSIONS

In this paper we have presented CHAI, a method for generating rules that can filter candidate triples in the context of the Knowledge Graph completion task. Our proposal is applicable to any Knowledge Graph and it allows for the generation of new knowledge by producing sets of promising candidate triples that contain potentially correct information to be analyzed by a fact checking model, which leads to a more efficient completion process.

When evaluated on four well-known Knowledge Graphs, our results show that CHAI is able to generate rules that not only produce a very large proportion of the missing knowledge that is considered correct in a typical testing split, but that also results in significantly smaller sets of candidates than those that would be obtained by naïvely using all entities in a KG as potential candidates (up to less than 1% of their original sizes). Our results also show that CHAI is able to outperform the most recent similar proposal in the state of the art by achieving high levels of coverage (up to 99%) in all kinds of relations.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Daniel Ayala, Agustín Borrego, Inma Hernández, Carlos R. Rivero, and David Ruiz. 2019. AYNEC: All You Need for Evaluating Completion Techniques in Knowledge Graphs. In *ESWC*. 397–411. https://doi.org/10.1007/978-3-030-21348-0_26

[2] Daniel Ayala, Inma Hernández, David Ruiz, and Miguel Toro. 2019. TAPON: A two-phase machine learning approach for semantic labelling. *Knowledge-Based Systems* 163 (2019), 931–943. https://doi.org/10.1016/j.knosys.2018.10.017

[3] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *SIGMOD*. ACM, 1247–1250. https://doi.org/10.1145/1376616.1376746

[4] Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question Answering with Subgraph Embeddings. In *EMNLP*. ACL, 615–620. http://aclweb.org/anthology/D/D14/D14-1067.pdf

[5] Antoine Bordes and Evgeniy Gabrilovich. 2014. Constructing and Mining Web-scale Knowledge Graphs. In *SIGKDD*. ACM, 1967–1967. https://doi.org/10.1145/2623330.2630803

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

[7] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*. 2787–2795. https://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data

[8] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion. In *SIGKDD*. ACM, 601–610. https://doi.org/10.1145/2623330.2623623

[9] Sébastien Ferré. 2019. Link Prediction in Knowledge Graphs with Concepts of Nearest Neighbours. In *ESWC*. Springer International Publishing, 84–100. https://doi.org/10.1007/978-3-030-21348-0_6

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

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

[12] Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom M. Mitchell. 2014. Incorporating Vector Space Similarity in Random Walk Inference over Knowledge Bases. In *EMNLP*. 397–406. http://aclweb.org/anthology/D/D14/D14-1044.pdf

[13] Michael Glass and Alfio Gliozzo. 2018. A Dataset for Web-Scale Knowledge Base Population. In *ESWC*. Springer, 256–271. https://doi.org/10.1007/978-3-319-93417-4_17

[14] Vinh Thinh Ho, Daria Stepanova, Mohamed H. Gad-Elrab, Evgeny Kharlamov, and Gerhard Weikum. 2018. Learning Rules from Incomplete KGs using Embeddings. In *ISWC*, Vol. 2180. http://ceur-ws.org/Vol-2180/paper-25.pdf

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

[16] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 6, 2 (2015), 167–195. https://doi.org/10.3233/SW-140134

[17] Peng Lin, Qi Song, and Yinghui Wu. 2018. Fact Checking in Knowledge Graphs with Ontological Subgraph Patterns. *Data Science and Engineering* 3 (2018), 341–358. https://doi.org/10.1007/s41019-018-0082-4

[18] Sahisnu Mazumder and Bing Liu. 2017. Context-aware Path Ranking for Knowledge Base Completion. In *IJCAI*. AAAI Press, 1195–1201. https://doi.org/10.24963/ijcai.2017/166

[19] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. 2019. Anytime Bottom-Up Rule Learning for Knowledge Graph Completion. In *IJCAI*. ijcai.org, 3137–3143. https://doi.org/10.24963/ijcai.2019/435

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

[21] 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. 2018. Never-ending Learning. *Commun. ACM* 61, 5 (2018), 103–115. https://doi.org/10.1145/3191513

[22] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing YAGO: scalable machine learning for linked data. In *WWW*. ACM, 271–280. https://doi.org/10.1145/2187836.2187874

[23] Pouya Ghiasnezhad Omran, Kewen Wang, and Zhe Wang. 2018. Scalable Rule Learning via Learning Representation. In *IJCAI*. ijcai.org, 2149–2155. https://doi.org/10.24963/ijcai.2018/297

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

[25] Baoxu Shi and Tim Weninger. 2018. Open-World Knowledge Graph Completion. In *AAAI*. 1957–1964. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16055

[26] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*. 926–934. https://papers.nips.cc/paper/5028-reasoning-with-neural-tensor-networks-for-knowledge-base-completion

[27] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI*, Vol. 14. AAAI Press, 1112–1119. https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531

[28] Zhuoyu Wei, Jun Zhao, Kang Liu, Zhenyu Qi, Zhengya Sun, and Guanhua Tian. 2015. Large-scale Knowledge Base Completion: Inferring via Grounding Network Sampling over Selected Instances. In *CIKM*. 1331–1340. https://doi.org/10.1145/2806416.2806513

[29] Ziqi Zhang. 2017. Effective and efficient Semantic Table Interpretation using TableMiner$^+$. *Semantic Web* 8, 6 (2017), 921–957. https://doi.org/10.3233/SW-160242