Relevant Entity Selection: Knowledge Graph Bootstrapping via Zero-Shot Analogical Pruning
Lucas Jarnac, Miguel Couceiro, Pierre Monnin

To cite this version:

HAL Id: hal-04254979
https://inria.hal.science/hal-04254979
Submitted on 23 Oct 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
ABSTRACT

Knowledge Graph Construction (KGC) can be seen as an iterative process starting from a high quality nucleus that is refined by knowledge extraction approaches in a virtuous loop. Such a nucleus can be obtained from knowledge existing in an open KG like Wikidata. However, due to the size of such generic KGs, integrating them as a whole may entail irrelevant content and scalability issues. We propose an analogy-based approach that starts from seed entities of interest in a generic KG, and keeps or prunes their neighboring entities. We evaluate our approach on Wikidata through two manually labeled datasets that contain either domain-homogeneous or heterogeneous seed entities. We empirically show that our analogy-based approach outperforms LSTM, Random Forest, SVM, and MLP, with a drastically lower number of parameters. We also evaluate its generalization potential in a transfer learning setting. These results advocate for the further integration of analogy-based inference in tasks related to the KG lifecycle.

CCS CONCEPTS

• Computing methodologies → Knowledge representation and reasoning; Artificial intelligence; • Information systems → Clustering and classification; World Wide Web.

KEYWORDS

knowledge graph, construction, analogical inference, zero-shot learning, graph embedding

1 INTRODUCTION

Knowledge graphs (KGs) are “graphs of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities” [16]. More formally, KGs are directed and labeled multigraphs \((E, R, T)\), where \(E\) is the set of entities, \(R\) is the set of relations, and \(T\) is the set of triples \((h, r, t)\in E\times R\times E\), where \(r\) qualifies the relation holding between \(h\) and \(t\). An example of such a triple could be \((BarackObama, \text{instanceOf}, \text{Person})\). KGs have proven useful in many academic and industrial applications, including search enhancement, question-answering, recommender systems, and eXplainable Artificial Intelligence [16, 35, 45].

Building and completing a KG can be achieved with knowledge extraction approaches from structured or unstructured data (e.g., tables, texts) [40, 48]. This forms a virtuous loop in which the KG is both a supporting structure that provides entities and relations of interest to detect in data and the target structure to refine and complete. However, the cold start problem appears when the initial KG is empty, which motivates the need to build first a high quality nucleus [48]. Such a nucleus could be manually bootstrapped by experts, but this process is time-consuming. Some authors thus propose to focus on premium sources of entities and categories to automatically constitute the nucleus [48]. In this view, several works consider Wikidata [47], a large and generic KG collaboratively built to support Wikipedia, as a premium source [19, 41]. However, the sheer size of Wikidata entails a need to restrict Wikidata knowledge to be integrated into the KG nucleus to avoid irrelevant knowledge and scalability issues. As Wikidata contains more than 100 million entities\(^1\), authors adopt a distillation [41] or a pruning [19] process, in which seed entities\(^2\) of interest are identified in Wikidata and only parts of their neighborhood are included in the KG nucleus (see Figure 1). The selection of the neighboring entities leverages the ontology hierarchy, either only upward [41], or both upward and downward [19]. The latter brings much more entities, which makes it more prone to gathering irrelevant knowledge in the KG nucleus.

Building and completing a KG can be achieved with knowledge extraction approaches from structured or unstructured data (e.g., tables, texts) [40, 48]. This forms a virtuous loop in which the KG is both a supporting structure that provides entities and relations of interest to detect in data and the target structure to refine and complete. However, the cold start problem appears when the initial KG is empty, which motivates the need to build first a high quality nucleus [48]. Such a nucleus could be manually bootstrapped by experts, but this process is time-consuming. Some authors thus propose to focus on premium sources of entities and categories to automatically constitute the nucleus [48]. In this view, several works consider Wikidata [47], a large and generic KG collaboratively built to support Wikipedia, as a premium source [19, 41]. However, the sheer size of Wikidata entails a need to restrict Wikidata knowledge to be integrated into the KG nucleus to avoid irrelevant knowledge and scalability issues. As Wikidata contains more than 100 million entities\(^1\), authors adopt a distillation [41] or a pruning [19] process, in which seed entities\(^2\) of interest are identified in Wikidata and only parts of their neighborhood are included in the KG nucleus (see Figure 1). The selection of the neighboring entities leverages the ontology hierarchy, either only upward [41], or both upward and downward [19]. The latter brings much more entities, which makes it more prone to gathering irrelevant knowledge in the KG nucleus. For instance, the downward neighbors of Microsoft SharePoint include Content Management System, a relevant entity to keep, and Dating App, an irrelevant one to prune. Jarnac and Monnin [19] thus use several pruning thresholds based on node degrees and distances in the embedding space but highlight the difficulty to set global thresholds when applied to heterogeneous entities with different distributions of degrees and distances. Additionally, to the best of our knowledge, there is no publicly available benchmark dataset to evaluate such approaches.

In our work, we propose to tackle the limitations of fixed thresholds by training classifiers to select (or keep) relevant neighboring entities and prune irrelevant ones in a KG bootstrapping process. Specifically, we propose an analogy-based zero-shot approach. Analogies are quadruples of the form Paris : France :: Berlin :

---

\(^1\)https://www.wikidata.org/wiki/Wikidata:Statistics

\(^2\)Note that entities in Wikidata are identified by QIDs.
We experiment with Wikidata and two datasets in Section 4 and we also assess the performance of the different approaches based on gathering terms of interest may be relevant. In this view, Jarnac and Monnin [19] propose to perform a downward expansion (see Subsection 3.1). We empirically compare the behavior of our approach with several classifiers (e.g., Random Forest, LSTM) and symbolic approaches (e.g., depth pruning, threshold pruning [19]).

The main contributions of the paper are:

- We propose an analogy-based zero-shot approach to select relevant entities in the neighborhood of seed entities. This approach only needs training examples of entities to keep or prune for some seed entities and can extrapolate to new seed entities without selection / pruning examples for them.
- We present a comparative study of our analogy-based approach to other methodologies with respect to different performance metrics, the number of parameters to be trained and the generality of the models considered.
- We provide two annotated datasets of seed entities and relevant or irrelevant neighbors in Wikidata to start constituting publicly available benchmarks for the community.
- We present a comparative study of our analogy-based approach to other methodologies with respect to different performance metrics, the number of parameters to be trained and the generality of the models considered.
- We provide two annotated datasets of seed entities and relevant or irrelevant neighbors in Wikidata to start constituting publicly available benchmarks for the community.

The remainder of this article is structured as follows. We briefly survey related work about KG bootstrapping and analogy-based inference in Section 2, and we detail our analogy-based zero-shot approach to select relevant entities to bootstrap a KG in Section 3. We experiment with Wikidata and two datasets in Section 4 and we discuss our results in Section 5. Finally, Section 6 summarizes our work and outlines future research work.

2 RELATED WORK

Our work positions within approaches focusing on bootstrapping KGs, especially by pruning to limit the scope of the built KG. We review some prominent works of this line of research in Subsection 2.1. Additionally, we rely on analogical reasoning which has recently achieved significant performance on NLP-related tasks, and has been identified as a promising research direction for KG-related tasks as outlined in Subsection 2.2.

2.1 KG Bootstrapping and Pruning

The construction of ontologies and KGs usually entails the possibility of their reuse for other purposes. That is why, it is common to leverage existing ontologies and KGs to bootstrap others [15, 44], since they can be seen as premium sources [48]. To illustrate, YAGO3 combines the taxonomy of WordNet and the categories of Wikipedia pages [26], Knowledge Vault integrates the FreeBase KG [11], and P2xLOD first integrates several biomedical KGs to then interconnect and enrich them [33].

Due to the size and generic aspect of some KGs and ontologies, some authors resort to pruning to construct domain-specific KGs from them. One of the early examples is the work of Swartout et al. in 1996 [44] where they propose to build a domain-specific ontology starting from a large and generic ontology, SENSUS, of 50,000+ concepts. To do so, they start with seed concepts from the domain of interest that are manually linked to SENSUS. They then include all super-concepts up to the root. They also discuss that some subtrees bring additional concepts of interest. They manually identify them with the rationale that if some nodes of a subtree have been identified relevant, then the other nodes of the subtree may be relevant too. However, such a manual process is time-consuming.

Furthermore, incorporating subtrees of large ontologies may come at the expense of incorporating irrelevant knowledge, which is difficult to manually assess. To face such issues, automatic distillation or pruning approaches can be considered. The distillation process can be guided by documents from the domain of interest. For instance, Babayeva et al. develop a domain ontology for Cyber Defence exercises by collecting concepts from an existing ontology and documents on this topic [4]. Sibita et al. [41] build a KG about customer requirements starting from client verbatim and Wikidata. Specifically, they detect entities in text, link them to Wikidata, and integrate their direct classes and all their super-classes.

Regarding pruning approaches, they can be classified into two categories: aggressive pruning based on topology of the graph and soft pruning that requires human input to define the relevant taxonomic concepts. In [14], Faralli et al. introduce the CrumbTrail algorithm that prunes a directed noisy knowledge graph with the aim of obtaining an acyclic subgraph that contains all previously selected seed nodes. Using this CrumbTrail algorithm, Bordea et al. propose to build domain-specific taxonomies from the KG of Wikipedia categories. After a user has selected leaf nodes and a root node, the algorithm is applied on the KG to build the directed and acyclic graph that will form the taxonomy. They provide three datasets but they are not directly applicable to our task. Indeed, they have specific concerns w.r.t. to taxonomy building (e.g., upward extension from leaves to root, acyclic aspect) while we mainly focus on gathering terms of interest w.r.t. a domain without such concerns. Additionally, we consider that seed nodes may not be leaves and propose to perform a downward expansion (see Subsection 3.1).

This is a more difficult pruning task since not all subclasses of a class of interest may be relevant. In this view, Jarnac and Monnin [19] bootstrap an enterprise KG by expanding a set of business terms semi-manually aligned to Wikidata entities along their ontology.
hierarchy. According to the authors, the distance in the embedding space appears to be a good indicator of topic similarity or drift. Thus, to limit the expansion, they propose an automatic approach relying on node degree and distance thresholds. However, such thresholds are globally fixed for all seed entities, which may lead to varying performance when these entities belong to heterogeneous domains.

2.2 Analogy-Based Inference in KGs

Analogy-based inference is a basic process in human cognition [6, 31] that is tightly related to abstraction, adaptation and creativity. Analogy-based inference can be viewed as transferring knowledge from a source domain to a different, but somewhat similar, target domain by leveraging simultaneously on similarities and dissimilarities. Most of the literature in analogy-based inference is built on the notion of analogical proportions, i.e., statements of the form “A is to B as C is to D” represented as \( A : B :: C : D \) [28], and relies on two main tasks, namely, analogy detection that involves deciding whether a quadruple \( (A, B, C, D) \) constitutes a valid analogy \( A : B :: C : D \), and analogy solving that consists in finding the possible elements X that make \( A : B :: C : X \) a valid analogy.

When the underlying objects \( A, B, C \) and \( D \) are represented as vectors \( e_A, e_B, e_C \) and \( e_D \), respectively, in some vector space \( \mathbb{R}^n \), analogical proportions can be thought of in geometric terms as the parallelogram rule \( e_D - e_C = e_B - e_A \). For instance, the underlying elements \( A, B, C \) and \( D \) of the analogical proportion could be words [46], and \( e_A, e_B, e_C \) and \( e_D \) their vectorial representations [29, 30] or larger chunks of text such as sentences [1, 2, 51]. Analogy-based inference has been used to solve hard reasoning tasks and has shown its potential with competitive results in several machine learning tasks such as classification, decision making and recommendation [9, 12, 13, 17], in data augmentation through analogical extrapolation for model learning, especially in environments with few labeled examples [7, 8]. Moreover, it has been successfully applied in classical natural language processing (NLP) tasks such as machine translation [21], several semantic [23, 24] and morphological tasks [3, 27, 34], as well as in (visual) question answering [39], solving puzzles and scholastic aptitude tests [37], and target sense verification [50].

Analogy-based inference can also be used to address and tackle tasks related to the KG lifecycle such as semantic table interpretation or knowledge matching [32]. A few works already exist in this line of research and rely on graph embedding, similarly to NLP approaches relying on character or word embeddings. For example, Liu et al. [25] tackles the task of link prediction, i.e., completing triples \( (h, r, ?) \), and study whether KG embedding models respect the parallelogram rule of analogical inference. They show that modeling analogical structures in KG embedding models brings additional performance. Similarly, Yao et al. [49] propose a model based on analogy functions to enhance a KG embedding model for link prediction. Alternatively, Portisch et al. [38] evaluate whether KG embedding models for link prediction or data mining can be used for analogy detection. In a similar fashion, we propose to leverage KG embeddings in an analogy detection task, where analogies serve to detect relevant or irrelevant entities w.r.t. seed entities of interest.

3 ANALOGY-BASED ZERO-SHOT SELECTION OF RELEVANT ENTITIES

We consider that we have at our disposal a set of seed entities of interest. Such entities can be identified (semi-)manually by experts and/or via an automatic extraction of entities from texts in the domains of interest [19, 41, 44]. These seed entities are aligned with a generic KG, e.g., Wikidata, and we retrieve their neighboring entities of interest along the ontology hierarchy. We first describe how we traverse the ontology hierarchy (Subsection 3.1), and then how we keep relevant entities and prune irrelevant ones during this traversal using an analogy-based model (Subsection 3.2).

3.1 Expansion Along the Ontology Hierarchy

We retrieve the neighboring entities of seed entities of interest along the ontology hierarchy as illustrated in Figure 2, which distinguishes two directions for the expansion.

In the upward expansion (Figure 2b), we retrieve from a seed entity \( \circ \), its first-level classes \( \bullet \) following P31 (“instance of”) and P279 (“subclass of”) edges, i.e., we retrieve the classes that the entity directly instantiates, or those that directly subsume it. Then, we retrieve all their superclasses \( \diamond \) by following P279 edges up to the root of the hierarchy in a breadth-first search expansion.

In the downward expansion (Figure 2c), we retrieve from a seed entity \( \circ \), its first-level classes \( \bullet \) by following P31, P279, and reversed P279 edges, i.e., we retrieve the classes that the entity directly instantiates, those that directly subsume it, or those that it directly subsumes. We then retrieve all their subclasses \( \bigcirc \) following reversed P279 edges up to the leaves, in a breadth-first search expansion.

3.2 Selection of Relevant Entities

In this subsection, we focus on the problem of keeping relevant and pruning irrelevant entities when traversed during the expansion described in Subsection 3.1.

3.2.1 Formalization. We formalize the problem as follows: given a seed entity \( e_s \) and an entity \( e_r \) reached during the graph expansion from \( e_s \), our goal is to decide whether to keep or to prune \( e_r \). If \( e_r \) is
kept, then its neighbors are explored as described in Subsection 3.1. Otherwise, its neighbors are not explored.

We propose an analogy-based zero-shot classifier model $A$ such that:

$$
A(e_s, e_r) = \begin{cases} 
1 & \text{if } e_s \text{ should be kept} \\
0 & \text{if } e_s \text{ should be pruned} 
\end{cases}
$$

(1)

Recall that analogies are statements of the form “A is to B as C is to D” represented as $A : B :: C : D$ [27]. In our case, we use analogies of the form

$$
e_1^k : e_1^k :: e_2^k : e_2^k
$$

(2)

where $e_1^k$ and $e_2^k$ are two seed entities, $e_1^k$ is reached during the expansion from $e_1^1$, and $e_2^k$ is reached during the expansion from $e_2^1$. Using analogical inference, if Equation (2) is a valid analogy and we know the decision for the pair $(e_1^k, e_1^r)$, then we can extrapolate the decision for the pair $(e_2^k, e_2^r)$. To illustrate, if the analogy Hadoop : Big Data :: SharePoint : Content Management System is valid and we know that Big Data should be kept, then Content Management System should also be kept.

We propose three different configurations for our analogy-based classifier, that consider different valid and invalid analogies. To ease notation, we note $k$ a keeping decision for a pair, and $p$ a pruning decision for a pair. The three configurations are as follows:

- **Configuration C1**: Valid analogies are of the form $k :: k$. Invalid analogies are of the form $k :: p$.
- **Configuration C2**: Valid analogies are of the form $k :: k$. Invalid analogies are of the form $k :: p$ and $p :: p$.
- **Configuration C3**: Valid analogies are of the form $k :: k$ and $p :: p$. Invalid analogies are of the form $p :: k$ and $k :: p$.

It is worth noting that depending on the chosen configuration, the aforementioned inference is adaptable. For example, with $C_1$ and $C_2$, valid analogies only allow to extrapolate keeping decisions. On the contrary, with $C_3$, valid analogies can conclude on keeping or pruning the pair $(e_1^k, e_2^k)$, depending on the known decision for the pair $(e_1^k, e_1^r)$. The same rationale can be applied on invalid analogies. For example, with $C_1$, invalid analogies lead to a pruning decision for the pair $(e_1^k, e_1^r)$. On the contrary, with $C_3$, invalid analogies lead to decide for the pair $(e_1^k, e_2^k)$ the opposite decision of the pair $(e_1^k, e_1^r)$.

In addition to configurations, we propose to consider or not paths in the graph within the analogy-based model. To illustrate, consider the analogy in Equation (2) and the two following expansion paths that generated it:

$$
e_1^1 \Rightarrow e_1^3 \Rightarrow e_1^1
$$

$$
e_1^2 \Rightarrow e_1^4 \Rightarrow e_1^5 \Rightarrow e_1^2
$$

Our first formalization in Equation (2) only considers seed entities and reached entities, on which a decision is known or a decision is to be made by the model. We also propose to consider the paths leading to the reached entities. In this view, the analogy in Equation (2) becomes as follows:

$$
e_1^k : (e_1^k, e_1^r) :: e_2^k : (e_1^k, e_2^r, e_2^2).
$$

(3)

3.2.2 **Model.** We adopt the supervised machine learning model proposed by Lim et al. [23]. This model is presented in Figure 3 and relies on convolutional neural networks (CNNs). It takes as input the vector embeddings of each constituent of a quadruple.

Figure 3: The analogy-based classifier model with two convolution layers and one fully connected layer from [23].

In our case, for a quadruple without paths (i.e., Equation (2)), we simply concatenate the embeddings of the entities (i.e., $e_1^1, e_1^2, e_2^1, e_2^2$). For a quadruple with paths (i.e., Equation (3)), we concatenate the embeddings of the entities (i.e., $e_1^1, e_1^2, e_2^1, e_2^2, e_2^1, e_2^2$) and use zero-padding in order to respect the model’s fixed input dimension. We experimented with three zero-padding methods:

- **before** zeros are added before the sequence (e.g., before $e_1^1$ and before $e_2^1$)
- **between** zeros are added between the embedding of the seed entity and the embeddings of the entities in the path (e.g., between $e_1^1$ and $e_1^2$, and between $e_2^1$ and $e_2^2$)
- **after** zeros are added after the embeddings of the entities in the path (e.g., after $e_1^1$ and after $e_2^1$)

The model has two convolution layers, followed by a flattening operation and one fully connected layer. The first convolution layer is composed of $n_1$ filters. Each filter has a kernel size of $1 \times$ sequence length and a ReLU activation function. Filters are initialized with a He normal initializer and applied with a stride of (1, sequence length). The second convolution layer is composed of $n_2$ filters. Each filter has a kernel size of $2 \times$ and a ReLU activation function. Filters are initialized with a He normal initializer and applied with a stride of (2, 2). The last layer is a fully connected layer with one output and a sigmoid activation function to obtain a binary classification score in $[0, 1]$. We also add dropout after each convolution layer.

This model is well-suited to the task at hand. Indeed, the first convolution layer allows to compute dissimilarities for each pair of entities, while the second convolution layer compares these dissimilarities between the first and the second pairs forming the quadruple to classify as a valid or invalid analogy.

3.2.3 **Training.** We consider that we have at our disposal pairs $(e_1^k, e_1^r)$ whose keeping or pruning decision is known (e.g., annotations by experts).

To train our model, for each annotated pair $(e_1^k, e_1^r)$, and for each form of valid and invalid analogies of the considered configuration, we build $M$ analogies by sampling $M$ other adequate labeled pairs. To illustrate, in configuration $C_1$, invalid analogies are of the form $k :: p$. Thus, for a pair $(e_1^k, e_1^r)$ whose decision is keep, we build $M$ invalid analogies by selecting $M$ other pairs whose decision is prune. These $M$ pairs are selected by ascending order of proximity of seed entities in the embedding space. We then train our model by
minimizing the binary cross-entropy loss and taking into account possible unbalancing between valid and invalid analogies.

3.2.4 Inference. At inference, on an unknown pair \((e_1^2, e_2^2)\):

1. We select \(N\) pairs \((e_1^1, e_2^1)\) whose decision is known to be keeping and \(N\) pairs \((e_1^3, e_2^3)\) whose decision is known to be pruning. Specifically, for each type of decision, we order known pairs by ascending proximity of \(e_1^1\) and \(e_2^1\) in the embedding space and select the \(N\) first.
2. We generate \(2N\) quadruples with selected known pairs on the left and the unknown pair \((e_1^2, e_2^2)\) on the right.\(^3\)
3. For each of these quadruples, our model predicts whether it is a valid or invalid analogy, which constitutes a keeping or pruning prediction, depending on the chosen configuration (see Subsubsection 3.2.1).
4. We compute the average of the scores output by the model (in \([0, 1]\)) on each of the \(2N\) quadruples as follows:
   - For \(C_1\): we interpret the score as a vote for keeping
   - For \(C_2\): we interpret the score as a vote for keeping
   - For \(C_3\): (i) if the known pair \((e_1^1, e_2^1)\) has a keeping decision, the score is considered as a vote for keeping; (ii) if the known pair \((e_1^3, e_2^3)\) has a pruning decision, the score is considered as a vote for pruning. Indeed, in this case, a score close to 1 corresponds to a valid analogy of the form \(p \bowtie p\). A score close to 0 corresponds to an invalid analogy of the form \(p \bowtie k\). Thus, we use \(1 - \text{score}\) as a vote for keeping.
5. If the averaged keeping score is above a fixed threshold, we keep \(e_2^2\). Otherwise we prune it.

It should be noted that, at inference, our model extrapolates on pairs in which \(e_2^2\) and potentially \(e_2^3\) were not seen in training. This makes our approach fundamentally zero-shot.

4 EXPERIMENTS

We evaluate our analogy-based model on the Wikidata knowledge graph \([47]\) and two datasets containing seed entities and labeled keeping and pruning decisions for their neighboring entities. In particular, we compare the latter with baseline models such as Multi-Layer Perceptron (MLP), Long Short Term Memory (LSTM), Support Vector Machine (SVM), Random Forest, depth pruning, and threshold pruning \([19]\). We also evaluate our model in a transfer learning setting.

Since our approach requires KG embeddings, we experiment with the pre-trained embeddings of Wikidata available in PyTorch-BigGraph \([22]\).\(^4\) These embeddings were learned for more than 78,000,000 entities of the 2019-03-06 version of Wikidata. For building, training, and evaluating our models we used TensorFlow’s Keras API and scikit-learn \([36]\). Datasets\(^5\) and code\(^6\) of our experiments are publicly available.

4.1 Datasets

To the best of our knowledge, there is no publicly available benchmark dataset for the present task. This motivated us to build and publicly release the two datasets whose characteristics are detailed in Table 1. Specifically, we gathered two sets of seed entities: 455 seed entities from the Computer Science / Information Technology domain for Dataset 1 (e.g., entities related to telecommunications, network, or programming languages), and 105 seed entities from more heterogeneous domains for Dataset 2 (e.g., entities related to food, music, sport, or science). Table 1 shows the number of nodes reached with an unconstrained (i.e., without pruning) upward and downward expansion as described in Section 3.1 for both datasets. It can be noticed that the number of reached nodes upward is drastically lower than the number of reached nodes downward. This motivated us to solely focus on pruning during the downward expansion.

To obtain labeled keeping and pruning decisions for downward nodes for both datasets without having to label the whole neighborhood, we adopted the following process. We performed a downward expansion with the pruning approach proposed by Jarnac and Monnin \([19]\) with thresholds based on node degrees and distance in the embedding space. To configure these thresholds, we set \(\alpha = 1.5, \gamma = 20, \beta = 1.3\) following \([19]\). Accordingly to their proposal, we also consider two different embeddings for entities: (i) Embedding \(E_1\) in which the embedding of an entity is its vector in the considered pre-trained embeddings, and (ii) Embedding \(E_2\) in which the embedding of an entity is the centroid of the embeddings of its instances. If an entity does not have instances, its pre-trained embedding vector is used instead. Then, we manually labeled keeping and pruning decisions output by this approach on the two sets of seed entities. Note that, since we use pre-trained embeddings from 2019 and a Wikidata dump from 2022, some entities do not have embeddings. To deal with this problem, we filtered both datasets to ensure that all seed and reached entities have an embedding vector.

4.2 Experimental Setup

We now describe our experimental protocol.

4.2.1 Cross validation. We applied a 5-fold cross validation. We split the seed entities of each dataset into 5 sets \(S_i, i \in \{1, 2, 3, 4, 5\}\), where each set contains the same number of seed entities. Each set \(S_i\) is successively used for testing, while \(S_{(i-1)}\) is used for validation, and the remaining sets are used for training. To prevent over-fitting, we implement an early-stopping method based on the validation loss. We set the patience to 5 for models trained with 50 epochs, and to 20 for models trained with 200 epochs.

Such a splitting on seed entities at testing, guarantees that we evaluate the ability of the model to generalize on unseen seed entities. Additionally, some entities reached from these unseen seed entities are not seen during training either, as highlighted in Table 2. Such an experimental setup thus assesses the model’s capability to learn a relative similarity or dissimilarity between seed entities and reached entities, and to extrapolate it on unseen seed and reached entities. This extrapolation roots our zero-shot approach.

It can be noticed in Table 2 that test seed entities in Dataset 1 lead to more entities that were seen in training than in Dataset 2 (51-61% instead of 10-21%). This is a direct consequence of the

---

\(^3\)It is noteworthy that for configuration \(C_1\), the \(N\) pairs \((e_1^1, e_2^1)\) whose decision is pruning are not used, leading to only \(N\) quadruples being generated.

\(^4\)https://torchbiggraph.readthedocs.io/en/latest/pretrained_embeddings.html

\(^5\)https://doi.org/10.5281/zenodo.8091584

\(^6\)https://github.com/Orange-OpenSource/analogical-pruning
are not considered. In this view, we compare our proposed analogy-based model to the following baseline models: MLP, LSTM, Random Forest, SVM, depth pruning, and threshold pruning [19]. We call the model that does not consider paths (Equation (2)), and the model that considers paths (Equation (3)).

We consider paths of length \(3\), and \(4\). We use the Adam optimizer. We set the batch size do 32.

### 4.2.2 Transfer learning
We also tested our model in a transfer learning setting. We trained our model on labeled decisions of seed entities from different domains, leading to different entities being traversed when expanding from each seed entity.

### 4.3 Models
We compare our proposed analogy-based model to the following baseline models: MLP, LSTM, Random Forest, SVM, depth pruning, and threshold pruning [19]. We call analogy the model that does not consider paths (Equation (2)), and path analogy the model that consider paths (Equation (3)).

Note that the dimension of the pre-trained embeddings of Wiki-data is 200. Some parameters are used by several models and are detailed below:

- **Batch size** We set the batch size do 32.
- **Optimizer** We use the Adam optimizer.
- **Embedding** We consider the embeddings \(\mathcal{E}_1\) and \(\mathcal{E}_2\), as proposed in [19] and explained in Subsection 4.1.
- **Concatenation** Consider a pair \((e_s, e_r)\) formed by a seed entity and an entity reached. To decide whether to keep or prune \(e_r\), some models can take as input the horizontal concatenation of the embeddings of \(e_s\) and \(e_r\) (called horizontal or their difference (called translation).
- **Zero padding** We consider three zero-padding methods: before, between, and after, as detailed in Subsection 3.2.2.
- **Learning rate** We test with learning rates \(\epsilon \in \{0.0001, 0.001, 0.01\}\).
- **Dropout rate** We test with dropout rates \(\epsilon \in \{0, 0.3, 0.5\}\).
- **Path length** We consider paths of length \(\ell \in \{3, 4, 5\}\).

### Analogy configuration
We consider the three configurations for valid and invalid analogies \(C_1, C_2, C_3\) presented in Subsubsection 3.2.1.

### Number of filters
We test with \((n_1, n_2) \in \{(2, 1), (4, 2), (8, 4), (16, 8), (32, 16), (64, 32), (128, 64), (256, 128)\}\).

The parameters used by the different considered models are given below, where we also describe specific parameters that are only applicable to one model.

- **Analogy (A)** Batch size, optimizer, embedding, learning rate, dropout rate, analogy configuration, number of filters,
- \(\mathcal{M} \in \{5, 10, 15, 20, 50, 100\}\), \(N = 20\).
- **Path analogy (PA)** Batch size, optimizer, embedding, learning rate,
- dropout rate, zero padding, path length, analogy configuration, number of filters,
- \(\mathcal{M} \in \{5, 10, 15, 20, 50, 100\}\), and \(N = 20\).
- **SVM** Embedding, concatenation, and unlimited number of iterations.
- **Random Forest (RF)** Embedding, concatenation, and number of estimators \(\mathcal{M} \in \{10, 50, 100, 150, 200, 250, 300, 400, 500\}\).
- **MLP** Batch size, optimizer, embedding, concatenation, learning rate,
- dropout rate, and hidden layers \(\mathcal{M} \in \{(100), (100, 50), (100, 50, 25), (200), (200, 100), (200, 100, 50), (200, 100, 50, 25)\}\).
- **LSTM** Batch size, optimizer, embedding, learning rate, zero padding, path length, and number of units \(\mathcal{M} \in \{50, 100, 150\}\).
- **Depth pruning (D)** Depth threshold \(\mathcal{M} \in \{1, 20\}\).
- **Threshold pruning (T)** Embedding, \(\alpha \in \{1.0, 1.1, \ldots, 2.0\}\),
- \(\gamma = 20, \beta \in \{1.0, 1.1, \ldots, 2.0\}\), and absolute degree \(\mathcal{M} = 200\).

Note that for models with a non-zero dropout rate, we use Monte Carlo Dropout.

For all models except analogy-based models, we explored all combinations of different parameter values. Given the important parameter space, for analogy-based models we first fixed the embedding to \(\mathcal{E}_1\), and the configuration to \(C_1\) on Dataset 2 in order to find the three best numbers of filters, the two best path lengths, the best zero padding method, and the best dropout rate. We then experimented with this reduced parameter space on Dataset 1 and Dataset 2.

### 4.4 Evaluation Metrics
Figure 4 illustrates the expansion along the ontology hierarchy from a seed entity \(\mathcal{E}_0\) with a pruning model. In such a setting, we only evaluate the model on nodes that are associated with a gold decision, i.e., nodes depicted by \(\bigcirc\) are not considered. In this view, it should be noted that it is possible for the model to leave nodes with gold decisions unexplored due to erroneous pruning decisions.

### Table 1: Statistics of Dataset 1 and Dataset 2 before and after filtering to only retain entities with embeddings. K stands for Keeping and P stands for Pruning.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Seed entities</th>
<th># Nodes up</th>
<th># Nodes down</th>
<th># P decisions</th>
<th>Depths P</th>
<th># K decisions</th>
<th>Depths K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 1 w/o filtering</td>
<td>455</td>
<td>1,507</td>
<td>2,593,609</td>
<td>3,464</td>
<td>[1, 4]</td>
<td>1,769</td>
<td>[1, 4]</td>
</tr>
<tr>
<td>Dataset 1 w/ filtering</td>
<td>439</td>
<td>1,469</td>
<td>2,593,575</td>
<td>2,910</td>
<td>[1, 4]</td>
<td>1,619</td>
<td>[1, 4]</td>
</tr>
<tr>
<td>Dataset 2 w/o filtering</td>
<td>104</td>
<td>1,152</td>
<td>1,247,383</td>
<td>314</td>
<td>[1, 2]</td>
<td>577</td>
<td>[1, 3]</td>
</tr>
<tr>
<td>Dataset 2 w/ filtering</td>
<td>104</td>
<td>1,152</td>
<td>1,247,383</td>
<td>314</td>
<td>[1, 2]</td>
<td>577</td>
<td>[1, 3]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fold</th>
<th>Fold 2</th>
<th>Fold 3</th>
<th>Fold 4</th>
<th>Fold 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 1</td>
<td>56.51</td>
<td>59.61</td>
<td>51.72</td>
<td>55.76</td>
</tr>
<tr>
<td>Dataset 2</td>
<td>20.50</td>
<td>21.49</td>
<td>12.36</td>
<td>17.04</td>
</tr>
</tbody>
</table>

### Table 2: Percentage of entities reached when testing and already seen when training, for each fold and dataset.

### Table 3: Parameters used by the different considered models.

<table>
<thead>
<tr>
<th>Analogy configuration</th>
<th>Number of filters</th>
<th>Depth pruning</th>
<th>Threshold pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analogy (A)</td>
<td>(\mathcal{M} \in {5, 10, 15, 20, 50, 100}), (N = 20).</td>
<td>(\mathcal{M} \in {1, 20}).</td>
<td>(\alpha \in {1.0, 1.1, \ldots, 2.0}), (\gamma = 20, \beta \in {1.0, 1.1, \ldots, 2.0}), and absolute degree (\mathcal{M} = 200).</td>
</tr>
</tbody>
</table>
higher in the hierarchy (i.e., nodes depicted by ☐). To take into account these various cases in our evaluation, we use the following metrics:

\[
\text{Precision} = \frac{K}{K + P} \quad \text{Recall} = \frac{K}{K + P + K}
\]

\[
\text{Accuracy} = \frac{K}{K + P} \quad \text{F}1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]

This corresponds to a binary classification in which the keeping decision is the positive class and the pruning decision is the negative class.

4.5 Results

We introduce our results in this section following the two setups described in Section 4. We further discuss them in Section 5.

4.5.1 Cross validation. We present in Table 3 the performance of the different models on the task of keeping relevant entities and pruning irrelevant ones on Dataset 1 and Dataset 2. Figure 5 depicts these results with error plots to better assess the variability or stability of each model. Note that we present the results of the reference Threshold whose decisions were labeled to build the datasets. However, we do not use them to draw comparisons and conclusions because of the bias that would constitute using such results in both the dataset building and evaluation processes.

For each model, Table 3 and Figure 5 only present the best results in terms of F1-score (primary criterion) and accuracy (secondary criterion) that were obtained when exploring the parameter space. The best parameters were the following:

**Analogy** Embedding $E_1$, $(n_1, n_2) = (16, 8)$, dropout = 0.5, configuration $C_2$ and learning rate = 0.001 (for Dataset 1), configuration $C_1$ and learning rate = 0.01 (for Dataset 2).

**Path analogy** Embedding $E_1$, configuration $C_1$, learning rate = 0.001, zero-padding = between, path length = 4, $(n_1, n_2) = (16, 8)$, and dropout = 0 (for Dataset 1), path length = 3, $(n_1, n_2) = (4, 2)$, and dropout = 0.3 (for Dataset 2).

SVM Embedding $E_1$, concatenation = horizontal (for Dataset 1), and concatenation = translation (for Dataset 2).

Random Forest concatenation = horizontal, embedding $E_2$ and 200 estimators (for Dataset 1), embedding $E_1$ and 300 estimators (for Dataset 2).

MLP hidden layers = (200, 100, 50), embedding $E_1$, concatenation = horizontal, learning rate = 0.001, dropout = 0 (for Dataset 1), and Embedding $E_2$, concatenation = translation, learning rate = 0.01, dropout = 0.3 (for Dataset 2).

LSTM Embedding $E_1$, zero-padding = before, number of units = 150, path length = 5 and learning rate = 0.01 (for Dataset 1), path length = 3 and learning rate = 0.001 (for Dataset 2).

Threshold pruning $\alpha = 1.0$, embedding $E_1$, $\beta = 2.0$ (for Dataset 1), $\beta = 1.8$ (for Dataset 2).

4.5.2 Transfer learning. For our transfer learning setting, we used the best parameters found during the cross-validation on Dataset 1. We trained each model on 80% of Dataset 1, using 20% as validation for early-stopping. We then tested the trained models on Dataset 2. Results are presented in Table 4.

5 DISCUSSION

Table 3 highlights that depth in the ontology hierarchy cannot be used to keep relevant entities and prune irrelevant ones. Here, with a depth of 3-4, we obtain a perfect recall but a low precision. Using greater thresholds does not change results while lower ones would reduce the recall in favor of the precision. This was expected, especially in a collaborative and generic knowledge graph such as Wikidata, since different communities may have different granular representations of knowledge. To tackle this issue, one would need to specify different depth thresholds depending on the seed entity domains. Additionally, not all subclasses of an interesting
Table 3: Pruning evaluation results on Dataset 1 and Dataset 2, with the parameters leading to the best results for each model. P stands for average precision, R stands for average recall, F1 stands for average F1-score, and ACC stands for average accuracy. The best results are in bold and we underline the second best result. Also, we decided to present the reference threshold results used in the construction of both datasets.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dataset 1</th>
<th></th>
<th></th>
<th>Dataset 2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>F1</td>
<td>ACC</td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>Random Forest</td>
<td>71.68 ± 8.66</td>
<td>48.18 ± 7.10</td>
<td>56.66 ± 2.25</td>
<td>66.57 ± 3.09</td>
<td>68.85 ± 15.59</td>
<td><strong>100.00 ± 0.00</strong></td>
</tr>
<tr>
<td>SVM</td>
<td>45.43 ± 5.84</td>
<td>61.79 ± 3.90</td>
<td>52.02 ± 3.86</td>
<td>45.08 ± 7.09</td>
<td>78.73 ± 7.29</td>
<td>67.01 ± 12.18</td>
</tr>
<tr>
<td>LSTM</td>
<td>60.12 ± 8.20</td>
<td>66.68 ± 2.09</td>
<td>62.94 ± 4.98</td>
<td>66.68 ± 3.62</td>
<td>73.99 ± 9.80</td>
<td>93.80 ± 8.96</td>
</tr>
<tr>
<td>Analogy</td>
<td>79.72 ± 5.17</td>
<td><strong>76.00 ± 6.59</strong></td>
<td><strong>77.43 ± 2.38</strong></td>
<td>83.48 ± 3.05</td>
<td>78.49 ± 8.80</td>
<td>94.58 ± 2.96</td>
</tr>
<tr>
<td>Path analogy</td>
<td>80.10 ± 0.84</td>
<td>74.44 ± 5.28</td>
<td>77.06 ± 2.89</td>
<td>83.51 ± 2.87</td>
<td>81.63 ± 8.27</td>
<td>94.90 ± 2.16</td>
</tr>
</tbody>
</table>

Depth

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Dataset 1</th>
<th></th>
<th></th>
<th>Dataset 2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>74.50 ± 5.22</td>
<td>81.86 ± 2.25</td>
<td>77.94 ± 3.58</td>
<td>83.02 ± 5.38</td>
<td>84.02 ± 5.72</td>
<td>89.51 ± 6.12</td>
</tr>
</tbody>
</table>

Table 4: Transfer results obtained by training on Dataset 1 and testing on Dataset 2. P stands for average precision, R stands for average recall, F1 stands for average F1-score, and ACC stands for average accuracy. The best results are in bold and the second best are underlined.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dataset 1 → 2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>F1</td>
</tr>
<tr>
<td>Random Forest</td>
<td>83.09</td>
<td>20.40</td>
<td>32.75</td>
</tr>
<tr>
<td>SVM</td>
<td>64.62</td>
<td>57.04</td>
<td>60.59</td>
</tr>
<tr>
<td>MLP</td>
<td>69.90</td>
<td>51.99</td>
<td>59.63</td>
</tr>
<tr>
<td>LSTM</td>
<td><strong>92.83</strong></td>
<td>74.73</td>
<td>82.80</td>
</tr>
<tr>
<td>Analogy</td>
<td>74.95</td>
<td>64.26</td>
<td>69.19</td>
</tr>
<tr>
<td>Path analogy</td>
<td><strong>91.49</strong></td>
<td><strong>83.39</strong></td>
<td><strong>87.25</strong></td>
</tr>
</tbody>
</table>

Table 5: Number of trainable parameters for each model.

for models to correctly learn similarity and dissimilarity between reached entities and seed entities. Additionally, such an homogeneity may also entail a fuzzy keeping/pruning boundary. To illustrate, starting from a network protocol of a specific layer of the OSI model, a protocol from another OSI layer was manually labeled with a pruning decision. Such a very fine-grained decision may be difficult to capture by models. To better reflect the performance of each model, we also provide their number of trainable parameters in Table 5. It is then striking that the path analogy model obtains close or better performance than the LSTM with 150 to 800 times fewer parameters. This global evaluation, taking into account the performance measured with traditional metrics as well as the number of trainable parameters, shows the superiority of our proposed analogy-based model.

Table 4 shows the performance of the compared models on the transfer learning setting. Again, it appears that the LSTM and path-analogy models are the two best performing models. While the LSTM obtains a better precision, the path-analogy model outperforms on recall, F1-score, and accuracy by 4 to 9 points. To better assess the generalization capability of these two models, we provide in Table 6 the breakdown of results from Table 3 depending on whether the reached entities when testing were seen or unseen during training. As can be expected, both models perform less on unseen entities. We notice that their scores are similar on Dataset 1 whereas the path analogy model outperforms the LSTM on both unseen and seen entities in Dataset 2. Recall that Dataset 2 contains much more unseen entities in testing that Dataset 1 (Table 2). These results thus demonstrate the higher generalization capability of our proposed analogy-based model. We posit that the formalization of
analogue quadruples and the use of a CNN lead the model to learn to compute and compare relative similarities and dissimilarities between the two pairs in a quadruple. In turn, this leads to better extrapolation capabilities to decide on an unseen pair when compared to a seen pair within an analogue quadruple. Consequently, we think analogy-based models are well-suited for such zero-shot settings.

To extend our approach, we could envision to test our model in a few-shot setting by having some labeled decisions to train on for seed entities considered in testing. In a real-world use-case scenario, this would correspond to asking experts to label a few neighbors of each seed entity before performing the expansion along the ontology hierarchy. This could be of interest to further test the extrapolation capability of our analogy-based model. However, we believe that in a real-world scenario, experts would rather label as many neighbors as possible of some seed entities and expect the model to extrapolate on new seed entities, hence our focus on the zero-shot setting. As aforementioned, some pruning decisions are motivated by errors in the ontology hierarchy of Wikidata, which is known to contain to be potentially noisy [42]. Another extension of our approach thus consists in applying it to ontology maintenance.

Regarding our model, we leverage KG embeddings pre-trained with a translational model. However, there exist several types of KG embedding models, such as translational, complex, Gaussian or Graph Neural Network-based ones [20]. It would thus be interesting to evaluate which types of embedding models are better suited to serve an analogy-based model. Additionally, instead of using frozen KG embeddings previously learned on a specific task, we could envision learning simultaneously the graph embeddings and the CNN layers, similarly to what was done in [27] with character and word embeddings. Finally, it is noteworthy that our work does not rely on Large Language Models (LLMs) such as BERT [10]. This purposely allows us to assess if the structure of the graph provides enough information to learn useful embeddings for selecting relevant entities. Future research directions could involve enriching our approach with LLMs, while raising additional issues to face such as noisy labels or homonyms. To illustrate, the entity “role” can be a part played by a performer or an identity of an item in relation to another specified item.

### 6 CONCLUSION

In this paper, we considered the task of bootstrapping a knowledge graph (KG) by selecting relevant entities in the neighborhood of seed entities of interest in a generic KG. We proposed an analogy-based model to keep or prune neighbors of seed entities in a zero-shot setting and two labeled datasets to evaluate models on this task. Compared with standard classifiers, our model outperformed while presenting a drastically lower number of parameters. Additionally, it showed better extrapolation capabilities in zero-shot and transfer learning settings. Such results advocate for the further study of analogy-based models in tasks related to the KG lifecycle or requiring extrapolation capabilities, which we will address in future work.

### ACKNOWLEDGMENTS

This work is supported by the AT2TA project (https://at2ta.loria.fr/) funded by the French National Research Agency (“Agence Nationale de la Recherche” - ANR) under grant ANR-22-CE23-0023.

### REFERENCES


---

Table 6: Performance of the LSTM and path analogy models depending on whether entities reached when testing were seen or unseen in training. P stands for average precision, R stands for average recall, F1 stands for average F1-score, and ACC stands for average accuracy. Best results for each category (i.e., seen or unseen) are in bold.

<table>
<thead>
<tr>
<th></th>
<th>Dataset 1</th>
<th>Dataset 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P R F1 ACC</td>
<td>P R F1 ACC</td>
</tr>
<tr>
<td>LSTM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unseen entities</td>
<td>73.90 ± 5.97</td>
<td>69.69 ± 7.90</td>
</tr>
<tr>
<td>Seen entities</td>
<td>84.63 ± 6.80</td>
<td>80.95 ± 5.51</td>
</tr>
<tr>
<td>Path analogy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unseen entities</td>
<td>73.82 ± 3.62</td>
<td>69.44 ± 6.09</td>
</tr>
<tr>
<td>Seen entities</td>
<td>85.20 ± 2.10</td>
<td>78.25 ± 5.20</td>
</tr>
</tbody>
</table>

---


