Generating Explainable Abstractions for Wikidata Entities

Nicholas Klein, Filip Ilievski, and Pedro Szekely

Information Sciences Institute, University of Southern California
{nmklein,ilievski,pszekely}@isi.edu

Abstract. The large coverage and quality of the Wikidata knowledge graph make it suitable for usage in downstream applications, such as entity summarization, entity linking, and question answering. Yet, most retrieval and similarity-based methods for Wikidata make limited use of its semantics, and lose the link between the rich structure in Wikidata and the decision-making algorithm. In this paper, we investigate how to define abstractive representations (profiles) of Wikidata entities. We propose a scalable method that can produce profiles for Wikidata entities based on salient labels associated with their types. We represent the resulting profiles as a graph, and compute profile embeddings. Our empirical analysis shows that the profiles can capture similarity competitively to baselines, but excel in terms of explainability. On the task of neural entity linking in tables, the profiles outperform all baselines in terms of accuracy, whereas their human-readable representation clearly explains the source of improvement. We make our code and data available to facilitate novel use cases based on the Wikidata profiles.

Keywords: Wikidata · Entity profiling · Representation learning · Explainability

1 Introduction

Due to its coverage, quality, and integration with Wikipedia, Wikidata has been gaining popularity in recent research on entity summarization [4], entity linking [5,20], and question answering [1]. Such Wikidata applications depend on high-quality retrieval of relevant knowledge for a given task. However, we observe a lack of precise methods that retrieve and reason over relevant knowledge in Wikidata. Reasoning systems that rely on network metrics like PageRank [22], or learn graph embeddings [2,27], encode the structure of the graph, but lose much of the precise information. These strategies are insufficient in downstream tasks that rely on more precise notions of similarity, such as entity linking. For example, in order to disambiguate a mention of the film 'Inside Man' in a web table that describes films filmed in the USA around 2010, it is crucial to align the spatio-temporal context of the table to the entity. The film *Inside Man (Q81224)* produced between 2000 and 2010 in the USA is much more appropriate in this context than the film *The Inside Man (Q10671423)*, which has been produced between 1980 and 1990 in Sweden.

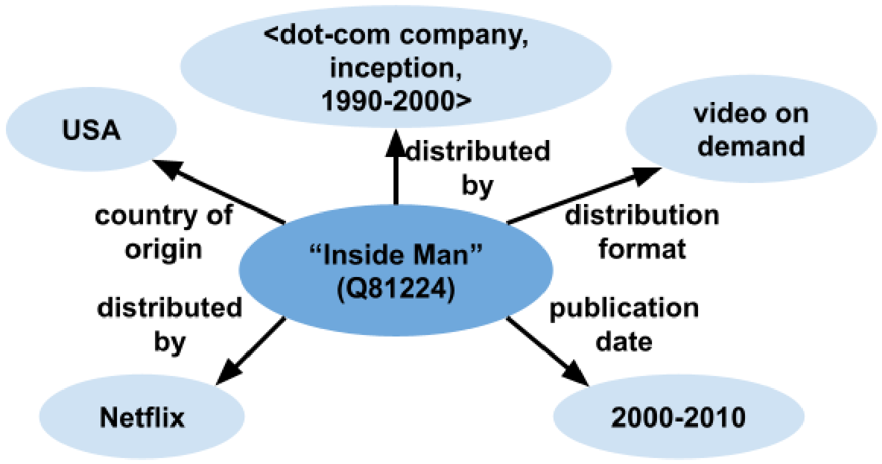


Fig. 1. Profile labels for the entity Inside Man (Q81224).

Rich reasoning knowledge is present in Wikidata. Yet, Wikidata has no intrinsic mechanisms to prioritize certain pieces of information in context. Prior work on summarization aims to provide a representative subset of the knowledge about a knowledge graph (KG) entity. Extractive summarization methods select a subset of the knowledge about an entity based on aspects like frequency, distinctiveness, and informativeness, but these have not been applied to summarize Wikidata entities at scale and do not attempt to abstract over the existing knowledge [17]. Abstractive summarization methods for Wikidata have generally been built to produce a human-readable description rather than preserving the graph structure [7]. Graph profiling [32] is able to provide machine-readable summaries for entities, but it has not been applied at scale or on hyper-relational graphs.

In this paper, we investigate how to define formal representations of Wikidata entities. Following prior work [32], we frame this task as profiling, i.e., an abstractive summarization task that aims to create representations that capture the distinguishable properties of an entity. An example of a profile for the Wikidata entity Inside Man (Q81224) is shown in figure 1, combining readily available information in Wikidata (distributed by Netflix) with aggregated information (publication date 2000-2010). By distilling and transforming salient knowledge about an entity, we expect that Wikidata profiles will be able to improve the accuracy and explainability on downstream applications, like entity linking. The contributions of this work are:

1. We motivate and define seven requirements for optimal entity representations in Wikidata: salience, support for hyper-relational knowledge models, sparsity, explainability, scalability, utility, and customizability.

2. We use a method that satisfies all seven requirements, and produces concise entity abstractions: *entity profiles*, which serve as human- and machine-readable representations of the Wikidata entities. We create graph embeddings for the entity profiles to enable them to be more easily and meaningfully used for comparing entities of differing types.
3. We evaluate the utility of the entity profiles and their various embeddings on estimating entity similarity, showing that the profiles capture meaningful and explainable similarity.
4. We evaluate the entity representations on a downstream task of tabular entity linking, by integrating them into a neural method. Our results show that the resulting method outperforms competitive baselines, and that the profiles provide human-readable explanation of the corrected errors.
5. We release the code[1] and the generated entity representations[2] publicly, in order to facilitate customization of our profiles to novel use cases and new knowledge graphs.

2 Requirements

To the best of our knowledge, no prior work has explored the creation of Wikidata profiles, i.e., abstract summaries of entities that capture their most salient information. Considering the growing adoption of Wikidata, it is urgent to devise strategies that generate profiles for a given entity based on its type and graph context. Below, we motivate seven requirements for Wikidata profiles:

R1. Salience Entities in knowledge graphs have various extent of information. Wikidata has no intrinsic mechanism to prioritize salient statements for an entity. It is important to devise strategies that can measure salience for an entity given its context, and this is a primary requirement for entity profiles.

R2. Hyper-relational knowledge model Popular knowledge graphs, like Wikidata and YAGO, represent their information in a hyper-relational format, where each edge can be associated with additional information, represented as a qualifier. For example, Barack Obama being a president of the USA is associated with the period between 2008 and 2016. A method that profiles Wikidata entities should exploit such information stored in the qualifiers.

R3. Addressing sparsity Modern knowledge graphs are notoriously sparse, e.g., only about 1% of the people in Wikidata contain information for attributes like native language and religion [9]. The sparsity of Wikidata is notably larger than other graphs like DBpedia.[3] An entity profiling method for Wikidata must appropriately address the sparsity of most of the entities in the knowledge graph.

[1] `https://github.com/nicklein/kgtk/tree/entity_profiling/entity_profiling`,
`https://github.com/nicklein/table-linker-pipelines`

[2] `https://drive.google.com/drive/folders/1cwLwzoamRVDKB5oG_Mxj_bdK1FGdzmdd?usp=sharing`

[3] In March 2020, DBpedia averaged 26 facts per entity while Wikidata - only 12.5 [26].

Table 1. An overview of the labels computed by our method.

Value type	**Format of Label's Value**	**Example**
Attribute Value Label (AVL)	a numeric value	*<human, mass, 185 lbs>*
Attribute Interval Label (AIL)	a numeric range	*<human, mass, 180 - 190 lbs>*
Relational Label (REL)	an entity	*<human, country of citizenship, United States of America>*
Relational Attribute Value Label (RAVL)	type constrained with an AVL	*<human, country of citizenship, <country, life expectancy, 78 years>>*
Relational Attribute Interval Label (RAIL)	type constrained with an AIL	*<human, country of citizenship, <country, life expectancy, 75-85 years>>*

R4. Explainability The entity profiles should be understandable by humans. They should correspond to human intuition and provide useful representations that introduce users to Wikidata entities. Profiles should use a *structured* representation to preserve the most salient semantic features of entities, rather than a free-text summary, which is the goal of the related task of summarization. In fact, summaries of profiles could then be easily be inferred based on a lexicalization system like [1], but not vice versa.

R5. Scalability Given the large growth of Wikidata [10], and the increasing set of use cases that benefit from it, its profiling method should be scalable to many types and a large number of entities.

R6. Utility The generated Wikidata profiles should clearly benefit common downstream applications, such as concept similarity and entity linking, both in terms of accuracy as well as transparency.

R7. Customizability To suit different use cases, the profiles may need to be computed multiple times with different parameters, and possibly over different KGs. To facilitate customizability, the method for generating profiles and its parameter values should be transparent, and its code and data should be publicly available.

Our work is loosely based on the profiling method by Zhang et al. [32] which computes profiles for DBpedia and LinkedMDB. As such, this prior method fulfills the requirements of salience (R1) and largely the requirements R3 on sparsity and R4 on explainability. However, this method has not been adapted to hyper-relational knowledge graphs (R2), its scalability is limited (R5), and its utility on downstream tasks has not been reported (R6). Finally, the provided experimental details and code for this method are incomplete, thus hindering its customizability to novel use cases and KGs (R7).

3 Profiling

A profile consists of salient property-value pairs for an entity type. The set of profile statements (*labels*) in Wikidata are of the form $< t, p, v >$, where t represents an entity type in Wikidata and p is a Wikidata property describing an attribute or a relation. The value v can be numeric (e.g., 75), a numeric range (75-85), an entity (`United States`), or a label with either an interval or a numeric value (e.g., countries with population of 2-10 million people). We provide an overview of the label types in Table 1. An entity profile is a union of the profile labels for all its semantic types.

Our profiling procedure, inspired by [32], consists of six steps: graph preprocessing, label generation, label filtering, embedding creation, profile curation, and profile embedding creation.

Preprocessing We use a subset of Wikidata from which entities from a small number of very populous classes of entities have been omitted[4]. We transform datetime values in Wikidata to year values. Values with finer precision are truncated, and values with coarser precision are interpreted as year-precision. Quantities with units are kept intact, and units are considered part of a label's property (e.g., 'mass in kg' and 'mass in lbs' are treated like two different, non-comparable properties). Entities may have multiple entries for a quantity property in the case of time-series data (e.g., multiple values for the population of Los Angeles at different points in time). To mitigate confusing results (e.g., labeling all countries as having a small population since all countries at some point had a smaller population), we use temporal validity qualifiers (e.g., start time) when available, in order to keep only the most up-to-date entries.

Label generation aims to create a large pool of labels from Wikidata. This step requires two substeps. First, we associate each entity to its semantic types. Wikidata defines two is-a relations: instance-of (P31) and subclass-of (P279). Based on these two relations, over 2 million nodes in Wikidata serve as classes, i.e., objects in statements where the predicate is P31 or P279. The number of classes in Wikidata is much larger than that in other KGs. For illustration, DBpedia contains less than 700 classes. Using only P31 results in a more restricted set of entity-type connections, and a smaller set of types per entity on average (1.05), while combining P31 with a transitive closure over P279 would result in a larger, and potentially noisier, set of types (38.66). In order to maximize efficiency (R5) of the profile computation and avoid introducing noise, we generate labels based solely on direct P31 links.

In the second step, we query Wikidata to generate type-based profiles, for each value type in Table 1. Attribute-valued (AVL) and relational (REL) labels are obtained directly through graph patterns. Within the AVLs, we discard

[4] The subset was taken from the 2021-02-15 Wikidata dump, omitting entities of the classes Scientific Publication (Q591041), Star (Q523), Galaxy (Q318), Review Article (Q7318358), Gene (Q7187), Chemical Compound (Q11173), and Protein (Q8054). The subset is available here: `https://drive.google.com/drive/folders/1cwLwzoamRVDKB5oG_Mxj_bdK1FGdzmdd?usp=sharing`.

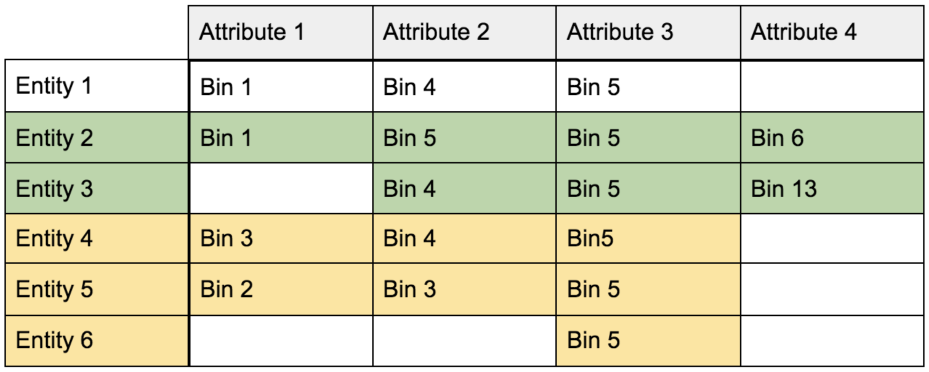
string-valued AVLs, as it is unclear how these would contribute to the utility of the profiles (R6). AILs are created by discretizing the AVLs: year-valued AVLs are translated into fixed-width 10-year bins, whereas quantity-valued AVLs are translated into bins based on what percentile their value falls into for their particular property. Finally, RAVLs and RAILs are created by querying Wikidata and using the already-created AVLs and AILs.

Label filtering To support the salience requirement (R1), we filter out labels with extreme frequencies. We filter labels which are extremely frequent (e.g., humans being instances of humans), as these are trivial. We filter labels which are extremely rare (e.g., identifier values), as their semantics is generally not representative. Formally, we define a *support* parameter, which captures the percentage of entities of the label's type that the label is applicable to. We only keep the labels whose support is between s_{min} and s_{max}.

Our analysis revealed that using the same $[s_{min}, s_{max}]$ interval for all five value types, as done by Zhang et. al. [32], is undesirable, as over 90% of the labels belong to the RAIL and RAVL value types. In addition, the particularly large sparsity of Wikidata (R3) could lead to a large imbalance across entity types, as well-populated entity types (e.g., *human*) will have many more labels compared to more sparse types (e.g., *fictional character in a musical work*). These factors can cause labels to apply to a smaller percentage of entities, making it more difficult to filter out unrepresentative labels without throwing away many potentially useful labels. Thus, we define support threshold values for each value type individually, favoring value types which can be expected to be more useful (RELs and AILs) over non-discretized variants (AVLs) and two-hop structures which are less intuitive to humans (RAVLs and RAILs) (R4). To further ensure that REL and AIL get a sufficient number of labels for each entity type, we set their threshold per entity type to the lowest value from a set T for which the number of labels is lower than, or equal to, pre-defined thresholds, t_{rel} and t_{ail}.

Embedding creation Similar to Zhang et al. [32], we train random walk embeddings of Wikidata nodes in order to assess their distinctiveness. We consider three types of walks, based on homophily, numeric attributes, and graph structure. The walks generated by each method are passed into a SkipGram model [19] in order to create the corresponding embeddings.

The *homophily* walks are generated directly on the Wikidata sub-graph of entity-to-entity relations. Thus, the corresponding 'H-embeddings' are computed identically to DeepWalk [23]. The *attributive* (A) and the *structural* (S) embeddings require novel strategies to generate walks. The method proposed by Zhang et al. would not be applicable to generate attributive walks, as it has been designed to work for a single type. We instead propose to discretize the numeric attributes of each entity into percentile-based bins. Here, each entity is represented with the binned values of its known attributes, which we will refer to as a *bin combination.* A 'hop' from a bin combination B_f to another bin combination B_t is allowed in the following cases (illustrated in figure 2): 1. All attribute bins in B_f are in B_t, with the exception of up to one of the attribute bins that is off by one percentile-bin. B_t may have extra bins. In figure 2, this allows hopping



	Attribute 1	Attribute 2	Attribute 3	Attribute 4
Entity 1	Bin 1	Bin 4	Bin 5	
Entity 2	Bin 1	Bin 5	Bin 5	Bin 6
Entity 3		Bin 4	Bin 5	Bin 13
Entity 4	Bin 3	Bin 4	Bin5	
Entity 5	Bin 2	Bin 3	Bin 5	
Entity 6			Bin 5	

Fig. 2. Examples of entities with different combinations of attribute-bins that could (highlighted green) and could not (highlighted yellow) be hopped to from Entity 1.

from entity 1 to entity 2, but not to entities 4 or 5. 2. All attribute bins in B_f are in B_t, with the exception of up to one of the attribute bins for which B_t has no bin for that attribute. B_t may have extra bins. In figure 2, this allows hopping from entity 1 to entity 3, but not to entity 6. With these rules, we can scalably compute a dictionary that defines the entities that can be hopped to from each entity by first computing the bin combinations that can be hopped to from each bin combination, and then expanding the bin combinations to their corresponding entities. Once this mapping of entities that can be hopped to from each entity has been created, it can be used to perform random walks as follows: given a starting node, the next node in the walk is chosen via uniformly random selection from the current node's list of entities that can be hopped to. The current node is then updated based on the selected next-node, and the walk can be iteratively continued in this fashion until the desired walk length is reached.

Analogously, to compute S-walks, we first define the allowed hops for each entity. Here, we seek structurally similar entities, which are defined as entities which connect to nodes of similar types. For scalability purposes, we create vectors that represent each entity with an array that contains its number of neighbors for n_t most common neighbor types. Each entity's n_n nearest neighbors are taken to form the set that can be hopped to during a walk.

Profile curation Because node embeddings natively capture entity similarity, we use them to compute distinctiveness of each remaining label. We apply the following formula:

$$D(l) = Avg_{e \in \varepsilon_t^l}(dist(e, centroid(\varepsilon_t^l))) - Avg_{e \in \varepsilon_t^{\bar{l}}}(dist(e, centroid(\varepsilon_t^l)))$$

where ε_t^l is the set of entities of type t that match label l, and $\varepsilon_t^{\bar{l}}$ is the set of entities of type t that do not match label l.

We select the labels to be included in the final profile for each type following an iterative label selection process [32]. This process selects distinctive labels while simultaneously increasing the number of entities that we can create profiles for and decreasing redundancy in the final set of labels. To start, the final label set for type t, L_t^f, is initialized to an empty set, and the candidate label set, L_t^c, is set to the filtered set of labels of type t. At each iteration, we choose the label l from L_t^c that has maximum value according to a re-ranking function, and move l from L_t^c to L_t^f. The re-ranking function is defined as

$$score(l) = [D(l) + \delta * reward(l, L_t^f) - (1 - \delta) * penalty(l, L_t^f)]$$

Here, $D(l)$ is the distinctiveness of label l. $reward(l, L_t^f)$ is a function that captures the potential increase in the total coverage of entities of type t in the KG by the labels in L_t^f if l were to be added to it. $penalty(l, L_t)$ is a function that captures the potential increase in redundancy of labels in L_t^f if l were to be added to it. δ is a hyperparameter that adjusts if we care more about increasing total coverage versus minimizing redundancy.

Specifically, *reward* and *penalty* are defined as follows:

$$reward(l_i, L_t) = \frac{|\bigcup_{l_j \in (L_t \cup \{l_i\})} \varepsilon_t^{l_j}|}{|\varepsilon_t|}$$

$$penalty(l_i, L_t) = \frac{\sum_{l_j \in L_t} |\varepsilon_t^{l_i} \cap \varepsilon_t^{l_j}|}{|L_t| * |\varepsilon_t|}$$

$\varepsilon_t^{l_j}$ is the set of entities of type t that match label l_j, whereas ε_t is the set of entities of type t.

This process ends either when we run out of labels in L_t^c, or when all labels remaining in L_t^c receive a score below a cutoff, $score_{min}$. This process is repeated to create a final label set for every type we want to profile. Since profiling is intended to abstract entities of a given type for the purpose of distinguishing them, it makes less sense to be used for types with few entities. Therefore we limit the types for which we create profiles to those that have more than 300 entities, resulting in 2,469 types to create final label sets for.

Profile embedding creation We learn profile representations in order to integrate the computed profiles into techniques for downstream tasks. To do so, we construct a profile-based knowledge graph (PKG), where Wikidata entities and profile-label-IDs are nodes, and edges are used to connect entities to their profile-label-IDs and profile-label-IDs to the profile-labels' information. Specifically, we include edges from each profile-label l to the following: entities that have l in their profile, l's class, l's property, l's value, a node that represents the class "Profile label", and other labels that have both the same class and property as l. Additionally, we create separate edges for the different kinds of values a profile label can have. These edges are for values that are Wikidata entities, other labels (for 2-hop labels), the lower and upper bounds of a year-range, and the lower and upper bounds of a quantity range. We use two standard graph embedding models: TransE [2] and ComplEx [27], in order to compute embeddings of the PKG nodes.

Implementation We opted to use five quantile bins for quantity-valued AILs based on the intuition that humans categorize quantities into a small number of ranges (e.g. low, low-mid, mid, high-mid, high). During label filtering, s_{max} is set at .9 for all label templates. The lower bound s_{min} is set at .1 for AVLs, RAVLs, and RAILs. For RELs and AILs, we use $T = [.1, .03, .01, .003, .001, .0003]$, $t_{ail} = 100$ and $t_{rel} = 500$. We set $n_t = 100$ and $n_n = 20$. For the H-embeddings we generate 5 walks of length 8 from every entity. The SkipGram model uses a representation size of 200, a window size of 5, and a min_count of 21. For the A-embeddings, 10 walks of length 10 are performed from all 12.1M entities that have a numeric attribute, we use a representation size of 200, a window size of 5, and a min_count of 0. For the S-embeddings, 10 walks of length 10 are generated from all 39M entities that have a neighbor of some type within the top 100 neighbor types. We use a representation size of 200, a window size of 5, and a min_count of 0. To create HAS-embeddings, H-, A-, and S-walks are combined by performing 10 walks of length 10 from every applicable entity for each walk strategy and concatenating them. The walks are then passed to a SkipGram model using a representation size of 200, a window size of 5, and a min_count of 0. During the profile curation, we set δ to .5 and $score_{min}$ to 0. The lower bound for the number of class instances required to create profiles for a class is set to 300. Future work should include further tuning of these parameters.

For efficient graph manipulation and computation of graph embeddings, we use the Knowledge Graph ToolKit (KGTK) [8]. To query neighboring embeddings efficiently, we build and search FAISS indices.[5] We use GraphTool to generate random walks.[6] We use the gensim package to compute random walk embeddings.[7]

4 Statistics

The resulting profiles cover 25,875,545 out of the total 42,546,555 entities in the preprocessed graph. Candidate generation on our input graph creates 80,375,852 labels for these entities. After label filtering, we are left with 10,881,631, and after profile curation, there are a total of 921,300 profile labels. We profile 2,469 classes out of the 63,159 P31-defined classes in the input graph. Statistics of the final profile labels by label-template are presented in table 2.

5 Intrinsic Evaluation

We measure the performance of the resulting profiles and profile-based embeddings on the task of entity similarity, expecting that they natively capture entity similarity, analogous to the word embeddings' ability to capture word similarity.

[5] https://github.com/facebookresearch/faiss
[6] https://graph-tool.skewed.de/
[7] https://radimrehurek.com/gensim/index.html

Table 2. Profile statistics per label type.

Label Template	**# Unique Labels**	**Avg # Labels per Entity (per Entity that has a Profile)**	**# Distinct Properties or Property Paths**
AVL	523	0.0258 (0.0425)	57
AIL	113,974	0.4935 (0.8114)	447
REL	507,143	1.4333 (2.3567)	931
RAVL	89,050	5.7490 (9.4530)	616
RAIL	167,771	9.2655 (15.2350)	1102

5.1 Task and Benchmark

Following common practice in the concept and word similarity tasks, we assume that the similarity of two entities can be expressed with a decimal score. Similarity methods are evaluated based on the correspondence of their scores to human judgments. As, to the best of our knowledge, there is currently no benchmark with similarity scores for Wikidata entities, we create one based on the popular word similarity dataset WordSim-353 [6].[8] WordSim-353 contains 353 pairs of English words (344 without duplicate pairs), along with human similarity scores. We map these word pairs to Wikidata nodes manually. As we observed that the original scores often conflate the notions of semantic similarity (*car-bike*) with relatedness (*car-wheel*), we re-annotated the dataset with numeric scores between 1 and 4: using 1 for (near-) identity; 2 for cases where two entities are partially substitutable, one is a slight specification of the other, or they are close siblings of the same category; 3 for pairs that are related by one of the following relations: distant inheritance, location, utility/capability, part-whole, antonymy, or domain; and 4 for unrelated pairs. Five researchers participated in this annotation, after which we produced new scores by averaging the five annotations. We make the resulting benchmark available for future evaluations.[9] As this data is dominantly about concept pairs, we select an entity subset with the 41 pairs for which we have profiles. We evaluate against human judgments on this subset using ordinal association metrics: Kendall's τ and Spearman correlation.

5.2 Methods

We use cosine similarity between the profile embeddings of two entities as their similarity score. We evaluate the TransE and ComplEx profile embeddings (*P-TransE* and *P-ComplEx*, respectively). Analogously, we use cosine similarity of the following baseline embeddings: 1) *TransE* and *ComplEx* embeddings computed over the original Wikidata graph; 2) *H-embeddings* and *S-embeddings*;[10]

[8] `http://www.gabrilovich.com/resources/data/wordsim353/wordsim353.html`

[9] `https://drive.google.com/drive/folders/1cwLwzoamRVDKB5oG_Mxj_bdK1FGdzmdd?usp=sharing`

[10] We exclude the result for the A-embeddings as they only cover 16/41 entity pairs.

Table 3. Correlation with human similarity rankings of 41 entity pairs, measured by Kendall's τ and Spearman's correlation coefficient.

Method	τ	**Spearman**
BERT	0.220	0.310
Class	**0.507**	**0.689**
TransE	0.407	0.523
ComplEx	<u>0.465</u>	<u>0.608</u>
H-embeddings	0.349	0.453
S-embeddings	0.171	0.223
Profile	0.418	0.536
P-TransE	0.418	0.561
P-ComplEx	0.393	0.518

3) *BERT* embeddings, where we use sentence-transformer's BERT-large model to compute an embedding over a lexicalized sentence of the node label and manually chosen relations.

Besides embeddings, we also evaluate our human-readable *profiles* directly on the similarity task. Here, we move the entity type from the labels and represent it as an additional aspect to compare on. Finally, our *class* baseline computes the set of common is-a parents for two nodes, where each shared parents is weighted by its inverse document frequency (IDF), computed based on the number of instances that transitively belong to that parent class.[11]

5.3 Results

Table 3 presents the results of our similarity evaluation.[12] The profiles and the profile embeddings show competitive performance to the other baselines. The best similarity is obtained by the class-based metric, whereas the Profile method and ComplEx come second. We observe that using the profile embeddings results in similar performance to using the profiles directly, indicating that the profile embeddings are capturing the information in the profiles that is relevant to similarity measurement. Interestingly, the correlation between the original TransE and ComplEx embeddings and the profile ones is medium: 0.460 and 0.594 τ, which shows that the profile embeddings and the original embeddings partially focus on the same knowledge.

The main advantage of the profiles over all other baselines is their explainability. For example, the profiles rank the pair king-queen as 7th highest, with similarity of 0.615, because of their five shared labels: subclass of monarch, instance of profession, instance of noble title, instance of position, and honorific

[11] Here, the is-a relations are computed as a transitive closure over both the subclass-of (P279) and the instance-of (P31) relations.

[12] We omit the results for the A-embeddings as these only cover 16 of the 41 pairs. Class, ComplEx, and TransE all miss 1 pair, BERT misses 8, and S-embeddings miss 5. Missing scores for a method are filled with its median score.

prefix ‘majesty’. Meanwhile, the profiles give the pair king-cabbage a minimal score of 0.0, as no salient labels between these two entities overlap.

6 Extrinsic Evaluation

We test the utility of our profiling method extrinsically on the table linking task. Assuming that the entities in the same table column are semantically similar (e.g., European countries, or films made between 2000 and 2010), we expect that the similarity captured by our profiles will be meaningful in the process of selecting the correct candidate for each cell. We measure the impact of the profiles on the table linking performance and compare it to other baselines.

6.1 Task and Benchmark

Given a table with cell values expressed in natural language, the table linking task asks systems to associate each cell with a single knowledge graph entity. The systems are then evaluated by using accuracy, i.e., counting the percentage of cells for which a system provided the correct top-1 prediction. We evaluate on two benchmarks: T2Dv2[13] and SemTab 2020 round 4 (SemTabR4)[11]. For both benchmarks, we randomly select a subset of tables and split them into train, development, and test sets. From the T2DV2 benchmark, we selected 46 tables, which we split into 30 train, 8 dev, and 8 test tables. The SemTab 2020 round 4 data consists of 22k tables - we randomly sample 86 tables and we split them into 50/18/18 sets. For a more meaningful comparison of the different table linking variants, we omit cells that do not have a Wikidata ground truth or do not have the ground truth entity amongst their candidates. The resulting tables included in these benchmarks all have at least 4 cells to link.

6.2 Neural table linking

We perform table linking by using a Siamese neural network model with a contrastive loss objective function. The neural model takes the features for a single candidate as input and provides a plausibility score between 0 and 1 for that candidate. We generate the set of candidates for a cell by combining the results of four ElasticSearch queries over the Wikidata labels and aliases, namely: exact match, fuzzy match, n-gram query, and a prefix n-gram query. The candidate with maximum score per cell is selected as the prediction by the neural method.

We define several base feature sets that the profile and profile embedding features will be added to, each with an increasing number of commonly used features. *String_Sim4 (StrSim4)* comprises four string-similarity metric scores that look at a candidate’s label: Jaro-Winkler, Monge-Elkan, Levenshtein, and Elastic-Search. *String_Sim6 (StrSim6)*: contains *String_Sim4*’s features plus two additional string-similarity metric scores that can also look at a candidate’s alias–the first returns 1 if the candidate is the only exact-match in the cell, and

Table 4. Micro-averaged top-1 accuracy score on test tables for models trained with different sets of features.

	Dataset						
	T2Dv2			SemTabR4			
	Base Feature Set			**Base Feature Set**			
Base+	StrSim4	StrSim6	Standard7	StrSim4	StrSim6	Standard7	**Avg**
None	0.5167	0.6003	0.6077	0.6680	0.7952	0.8594	0.6746
H	0.8124	0.7418	0.7641	**0.9390**	**0.9341**	**0.9498**	<u>0.8569</u>
A	0.6379	0.6257	0.6379	0.7281	0.7694	0.8603	0.7099
S	0.5362	0.6081	0.5841	0.6806	0.7684	0.8498	0.6712
TransE	0.5889	0.5984	0.6294	0.6700	0.7663	0.8470	0.6833
ComplEx	0.5718	0.5969	0.6004	0.6903	0.7918	0.8584	0.6850
Profile-TransE	0.6983	0.6564	0.6669	0.9133	0.9159	<u>0.9206</u>	0.7952
Profile-ComplEx	0.6023	0.7116	0.6922	<u>0.9138</u>	<u>0.9187</u>	0.9202	0.7931
Profiles	0.7685	<u>0.8484</u>	<u>0.8251</u>	0.8720	0.9013	0.9010	0.8527
TF-IDF-Profile	**0.8614**	0.7881	0.8136	0.8485	0.9062	0.9053	0.8539
D >.2 -Profile	<u>0.8547</u>	**0.8604**	**0.8779**	0.8388	0.8812	0.8695	**0.8638**

the second is Monge-Elkan computed on the candidate's alias. Finally, *Standard7* contains *String_Sim6*'s features plus PageRank.

We complement the base sets of features with one similarity feature at a time. Specifically, we evaluate similarity features computed using our profiles and profile embeddings against baseline similarity features computed using ComplEx, TransE, H-, A-, and S-embeddings. We use two variants of the profiles: Profiles and Profiles-TFIDF, where *TF-IDF* weights profile-labels by their TF-IDF, computed by viewing each table as a document, the profile-labels within a table as terms, and the set of tables as the set of all documents. We use two variants of the profile embeddings: P-TransE and P-ComplEx.

All similarity features require a contextual representation for a table to compute similarity against. For this purpose, we first disambiguate 25% of the table cells with their ground truth, which are used as pseudo-ground-truth (PGT) entities. The neural network is then used to predict the best candidate for the remaining 75% cells, by combining the base features with one similarity feature at a time. The embedding-based similarity features are computed as the average cosine similarity between the entity embedding and each of the PGT embeddings. The profile score for a given candidate is the average intersection size between the set of labels in its profile and the labels of each PGT entity, normalized within its cell.

To account for variations in performance due to the choice of PGT entities, we create 4 folds, each with a different 25% of cells chosen as PGT.[13]

[13] We report the median of the system performance on the 75% non-centroid cells. For fair comparison, we use the same 25-75 splits for all our neural network variants. The four splits for each benchmark along with the features we evaluate are available here: `https://drive.google.com/drive/folders/1cwLwzoamRVDKB5oG_Mxj_bdK1FGdzmdd?usp=sharing`.

6.3 Results

The results of the neural model with different sets of input features are shown in table 4. We observe that every added embedding feature improves the neural linking performance over the base set of features for the set of 4 string similarity features. For the sets of 6 and 7 base features, this holds for nearly all methods, except for the S-embeddings, TransE, and ComplEx.

Comparing the methods, we observe that the profiles and the profile embeddings clearly outperform the baselines. The only baseline that performs competitively to the profile methods is the H-embedding baseline, which suggests that relatedness is more relevant to candidate selection than similarity of numeric attributes and structure within the graph. The relative performance of the profiles and the profile-based embeddings differs per dataset. On the T2DV2 dataset, the profiles perform better than the profile embeddings, while this trend is reversed on the SemTabR4 data. Because ground truth entities within a table column are likely to be of the same class, the embeddings' ability to more fluidly compare entities of differing classes may cause them to consider more of the candidates as good choices. While this demonstrates that the profiles offer useful information for performing candidate selection, it also suggests a room for improving the profile-embeddings to better preserve the salient information encoded by the profiles.

On both datasets, and both for ComplEx and TransE, the profile embeddings clearly perform much better than the corresponding graph embeddings computed on the original graph, suggesting that the profiles capture salient information from the knowledge graph, which is critical for table linking methods. When we use the smallest baseline of 4 string similarity features (first column), both the vanilla- and the profile-graph embeddings improve performance of the model. When more features are added to the baseline, the vanilla graph embeddings become less useful, however the profile-graph embeddings are still able to increase the model's accuracy, which demonstrates their robustness.

We provide intuition for the improvement brought by the profile embeddings over the original embeddings on this task. We consider a table from the T2DV2 dataset which describes films that were made between 2005 and 2015, are generally produced in the USA, and distributed by Netflix. This can be deduced from the profiles of the PGT entities. The default neural network consisting of 7 base features chose the wrong candidate for the film "Inside Man" to be *The Inside Man (Q10671423)*, whose profile includes the labels: country of origin Sweden, publication date 1980-1990, and cast member being a human whose work period start is 1950-1960. Enhancing the set of features with the vanilla ComplEx embeddings does not help this issue. However, our *Profile* and *P-ComplEx* methods both are able to fix this error, to the correct film *Inside Man (Q81224)*, whose profile includes information that it is distributed by Netflix, its country of origin is USA, and its publication date is 2000-2010.

Such examples demonstrate the ability of our profiles to capture meaningful similarity between Wikidata entities, which is useful for downstream tasks such as entity linking. In addition, the coupling of the profiles with profile-embeddings allows for explanation of the decision of the model, thus enabling trust, error analysis, and future model improvements.

7 Related Work

Entity summarization Abstractive summarization methods for Wikidata exist [7], aiming to provide a short, abstract-like natural language description for an entity, and to enable natural language processing applications like question answering [1]. Entity profiling aims to extract a distinctive set of facets for an entity and represent them as entity-centered subgraphs. In this regard, entity profiling resembles extractive entity summarization, as both attempt to create a structured representation of an entity, by leveraging a subset of its triples. In order to select the most representative facets for an entity, three key notions have been used in entity summarization [17]: 1) frequency and centrality, measured through PageRank, property frequency, and topic modeling; 2) informativeness of features, based on either statistical or ontological information; and 3) diversity and coverage, based on textual, structural, or semantic information. The profiling method proposed in this paper is largely driven by the three notions proposed in earlier work: frequency, informativeness, and diversity. However, we focus on abstractive summarization of entities in Wikidata, by leveraging global properties and statistics of the knowledge graph in order to select and refine property values. Thus, the most similar prior work is by Zhang et al. [32], which defines profiles as salient representations of entities in the DBpedia and LinkedMDB knowledge graphs. Our method extends this prior work for hyper-relational knowledge graph, enhances its scalability, and evaluates the utility of the generated representations on downstream tasks.

Profiling methods Other profiling methods that aggregate or select entity representations have been proposed as well. Spina et al. [25] construct profiles for entities mentioned in microblog posts, by collecting commonly associated topics with an entity. However, this work is orthogonal to ours, as we focus on devising entity representations in knowledge graphs rather than text. In the realm of social media analysis, the term ‘profiling’ refers to a task of inferring a specific property (e.g., user’s location) based on other known information, such as their social network [12] or expressed content [18]. Li et al. [14] approach profiling as a task of consolidating an entity representation based on multiple structured sources, which depends on high-quality entity matching and error detection. In [9], we propose to use profiles as models of groups in the Wikidata knowledge graph, intended to capture a set of beliefs about the attributes of members of that group. Here, a group is defined through a set of attribute values (e.g., members of the class Human with nationality=French and occupation=politician). This prior work extends the notion of ‘type’ to a group that is based on any combination of attribute values, but its focus is on predicting expectations for unknown values, rather than selecting the most salient attribute values for an entity.

Embedding representations A standard method to encode an entity in a knowledge graph is by learning a vector embedding. Representation learning models are typically optimized to capture the triples in the KG and can be divided into: translation-based models [2,28,16], tensor factorization-based models [21,31,27], and neural networks-based models [24,3]. An alternative approach is to learn KG embeddings based on descriptions, by either text-based [30,29] or relation path-based models [15]. The profiles are orthogonal to embedding representations: profiles focus on obtaining a subgraph that captures salient and explicit labels for a node, whereas embeddings intend to encode graph representations into numeric vectors that can be seamlessly integrated into larger neural systems. The complementarity is shown in this work, where we use various embedding methods to encode the generated profiles.

8 Conclusions

This paper investigated how to define profile representations of Wikidata entities, by abstracting triples from the original graph and selecting a distinctive and salient subset of the abstracted graph. We devised seven requirements and a corresponding method that considers the data model, size, and sparsity of Wikidata. The resulting profiles describe millions of entities. We represent them as a knowledge graph and compute standard graph embeddings over the abstracted graph. The generated profiles and profile embeddings perform competitively to relevant baselines on the task of estimating entity similarity, while also providing a direct explanation of their similarity score through the profile overlap. On the downstream task of entity linking in tables, our method is able to outperform all baselines. Further analysis shows that the profiles are directly responsible for fixing of prior errors, demonstrating both their utility and explainability. Future work should perform further tuning and formal analysis of the greedy re-ranking procedure, investigate how to generate more effective profile embeddings, and devise more suitable benchmarks for directly evaluating entity similarity. We make all our code and data available and invite the community to explore the utility of profiles in novel applications like entity linking in text and entity recommendation.

References

1. Agarwal, O., Ge, H., Shakeri, S., Al-Rfou, R.: Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training (2021)
2. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. Advances in neural information processing systems **26** (2013)
3. Cai, L., Wang, W.Y.: Kbgan: Adversarial learning for knowledge graph embeddings. arXiv preprint arXiv:1711.04071 (2017)
4. Chisholm, A., Radford, W., Hachey, B.: Learning to generate one-sentence biographies from wikidata. arXiv preprint arXiv:1702.06235 (2017)
5. Delpeuch, A.: Opentapioca: Lightweight entity linking for wikidata. arXiv preprint arXiv:1904.09131 (2019)

6. Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin, E.: Placing search in context: The concept revisited. ACM Transactions on Information Systems pp. 20(1):116–131 (2002)
7. Gunel, B., Zhu, C., Zeng, M., Huang, X.: Mind the facts: Knowledge-boosted coherent abstractive text summarization. arXiv preprint arXiv:2006.15435 (2020)
8. Ilievski, F., Garijo, D., Chalupsky, H., Divvala, N.T., Yao, Y., Rogers, C., Li, R., Liu, J., Singh, A., Schwabe, D., et al.: Kgtk: a toolkit for large knowledge graph manipulation and analysis. In: International Semantic Web Conference. pp. 278–293. Springer (2020)
9. Ilievski, F., Hovy, E., Vossen, P., Schlobach, S., Xie, Q.: The role of knowledge in determining identity of long-tail entities. Journal of Web Semantics **61**, 100565 (2020)
10. Ilievski, F., Szekely, P., Schwabe, D.: Commonsense knowledge in wikidata. In: ISWC Wikidata workshop (2020)
11. Jiménez-Ruiz, E., Hassanzadeh, O., Efthymiou, V., Chen, J., Srinivas, K., Cutrona, V.: Results of semtab 2020. In: CEUR Workshop Proceedings. vol. 2775, pp. 1–8 (2020)
12. Jurgens, D.: That's what friends are for: Inferring location in online social media platforms based on social relationships. In: Seventh International AAAI Conference on Weblogs and Social Media (2013)
13. Lehmberg, O., Ritze, D., Meusel, R., Bizer, C.: A large public corpus of web tables containing time and context metadata. In: Proceedings of the 25th International Conference Companion on World Wide Web. pp. 75–76 (2016)
14. Li, F., Lee, M.L., Hsu, W.: Entity profiling with varying source reliabilities. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 1146–1155 (2014)
15. Lin, Y., Liu, Z., Luan, H., Sun, M., Rao, S., Liu, S.: Modeling relation paths for representation learning of knowledge bases. arXiv preprint arXiv:1506.00379 (2015)
16. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Twenty-ninth AAAI conference on artificial intelligence (2015)
17. Liu, Q., Cheng, G., Gunaratna, K., Qu, Y.: Entity summarization: State of the art and future challenges. Journal of Web Semantics p. 100647 (2021)
18. Mahmud, J., Nichols, J., Drews, C.: Where is this tweet from? inferring home locations of twitter users. In: Sixth International AAAI Conference on Weblogs and Social Media (2012)
19. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems. vol. 26. Curran Associates, Inc. (2013), `https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf`
20. Mulang, I.O., Singh, K., Vyas, A., Shekarpour, S., Sakor, A., Vidal, M.E., Auer, S., Lehmann, J.: Context-aware entity linking with attentive neural networks on wikidata knowledge graph. arXiv preprint arXiv:1912.06214 (2019)
21. Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: Icml (2011)
22. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Tech. rep., Stanford InfoLab (1999)
23. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. arXiv:1403.6652 (2014)

24. Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: Advances in neural information processing systems. pp. 926–934 (2013)
25. Spina, D., Meij, E., Oghina, A., Bui, M.T., Breuss, M., de Rijke, M., et al.: A corpus for entity profiling in microblog posts. In: LREC workshop on language engineering for online reputation management. vol. 168 (2012)
26. Tanon, T.P., Weikum, G., Suchanek, F.: Yago 4: A reason-able knowledge base. Harth A. et al. (eds) The Semantic Web. ESWC 2020. Lecture Notes in Computer Science **12123** (2020)
27. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: International conference on machine learning. pp. 2071–2080. PMLR (2016)
28. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 28 (2014)
29. Wang, Z., Li, J., Liu, Z., Tang, J.: Text-enhanced representation learning for knowledge graph. In: Proceedings of International Joint Conference on Artificial Intelligent (IJCAI). pp. 4–17 (2016)
30. Xie, R., Liu, Z., Sun, M., et al.: Representation learning of knowledge graphs with hierarchical types. In: IJCAI. pp. 2965–2971 (2016)
31. Yang, B., Yih, W.t., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575 (2014)
32. Zhang, X., Yang, Q., Ding, J., Wang, Z.: Entity profiling in knowledge graphs. IEEE Access **8**, 27257–27266 (2020)