A Simple Method for Inducing Class Taxonomies in Knowledge Graphs

Marcin Pietrasik$^{1[0000-0001-7559-8658]}$✉ and
Marek Reformat$^{1,2[0000-0003-4783-0717]}$

^{1} University of Alberta, 116 St & 85 Ave Edmonton AB
^{2} University of Social Sciences, 90-113 Łódź Poland
{pietrasi,reformat}@ualberta.ca

Abstract. The rise of knowledge graphs as a medium for storing and organizing large amounts of data has spurred research interest in automated methods for reasoning with and extracting information from this representation of data. One area which seems to receive less attention is that of inducing a class taxonomy from such graphs. Ontologies, which provide the axiomatic foundation on which knowledge graphs are built, are often governed by a set of class subsumption axioms. These class subsumptions form a class taxonomy which hierarchically organizes the type classes present in the knowledge graph. Manually creating and curating these class taxonomies oftentimes requires expert knowledge and is time costly, especially in large-scale knowledge graphs. Thus, methods capable of inducing the class taxonomy from the knowledge graph data automatically are an appealing solution to the problem. In this paper, we propose a simple method for inducing class taxonomies from knowledge graphs that is scalable to large datasets. Our method borrows ideas from tag hierarchy induction methods, relying on class frequencies and co-occurrences, such that it requires no information outside the knowledge graph's triple representation. We demonstrate the use of our method on three real-world datasets and compare our results with existing tag hierarchy induction methods. We show that our proposed method outperforms existing tag hierarchy induction methods, although both perform well when applied to knowledge graphs.

Keywords: Knowledge Graphs · Taxonomy Induction · Ontologies.

1 Introduction

Knowledge graphs are data storage structures that rely on principles from graph theory to represent information. Specifically, facts are stored as triples which bring together two entities through a relation. In a graphical context, these entities are analogous to nodes, and the relations between them are analogous to edges. In recent years, knowledge graphs have garnered widespread attention as a medium for storing data on the web. Public knowledge bases such as DBpedia [13], YAGO [12], and WikiData [28] are all underpinned by large-scale knowledge graphs containing upwards of one billion triples each. These knowledge bases find uses in personal, academic, and commercial domains and are ubiquitous in the research fields of the Semantic Web, artificial intelligence, and computer science broadly. Furthermore, private companies are known to use proprietary knowledge graphs as a component of their data stores. Google, for instance, uses a knowledge graph derived from Freebase [6] to enhance their search engine results by providing infoboxes which summarize facts about a user's query [24].

Ontologies are often used in conjunction with knowledge graphs to provide an axiomatic foundation on which knowledge graphs are built. In this view, an ontology may be seen as a rule book that provides semantics to a knowledge graph and governs how the information contained within it can be reasoned with. One of the core components of an ontology is the class taxonomy: a set of subsumption axioms between the type classes that may exists in the knowledge graph. When put together, the subsumption axioms form a hierarchy of classes where general concepts appear at the top and their subconcepts appear as their descendants.

One of the challenges that arise when working with large knowledge graphs is that of class taxonomy construction. Manual construction is time consuming and requires curators knowledgeable in the area. DBpedia, for instance, relies on its community to curate its class taxonomy. Similarly, YAGO relies on a combination of information from Wikipedia[1] and WordNet[2], both of which are manually curated. On the other hand, automated methods are not able to induce class taxonomies of the quality necessary to reliably apply to complex knowledge bases. Furthermore, they oftentimes rely on external information which may itself be manually curated or may only be applicable to knowledge bases in a particular domain. With this in mind, the impetus for automatically inducing class taxonomies of high quality from large-scale knowledge graphs becomes apparent.

In this paper, we propose a scalable method for inducing class taxonomies from knowledge graphs without relying on information external to the knowledge graph's triples. Our approach applies methods used to solve the problem of tag hierarchy induction, which involves inducing a hierarchy of tags from a collection of documents and the tags that annotate them. Although extensively studied in the field of natural language processing, these methods have yet to be applied to knowledge graphs to the best of our knowledge. In order to use these methods, we reshape the knowledge graph's triple structure to a tuple structure, exploiting the graph's single dimensionality in assigning entities to type classes. Furthermore, we propose a novel approach to inducing class taxonomies which outperforms existing tag hierarchy induction methods both in terms scalability and quality of induced taxonomies.

The remainder of this paper proceeds with Section 2 which provides an overview of the existing work done on inducing class taxonomies and tag hierarchies. We formalize the problem and introduce notation in Section 3. Our

[1] https://www.wikipedia.org/
[2] https://wordnet.princeton.edu/

proposed method is described in Section 4 and evaluated in Section 5. Section 6 concludes the paper.

2 Related Work

We divide our discussion of related work into two subsections: class taxonomy induction methods and tag hierarchy induction methods. Both of these methods are used to construct a hierarchy of concepts, however they differ in the type of data they are applied to. Class taxonomy induction methods are used on knowledge graphs and thus operate on data represented as triples. Tag hierarchy induction methods operate on documents and the tags that annotate them. In practice, these documents are often blog posts, images, and videos annotated by users on social networking websites. We can view our proposed method as a combination of the aforementioned categories as it takes the input structure of documents and tags but is applied to knowledge graphs to induce a class taxonomy.

2.1 Methods for Class Taxonomy Induction

Völker and Niepert [27] introduce *Statistical Schema Induction* which uses association rule mining on a knowledge graph's transaction table to generate ontology axioms. Each row in the transaction table corresponds to a subject in the graph along with the classes it belongs to. Implication patterns which are consistent with the table are mined from this table to create candidate ontology axioms. The candidate axioms are then sorted in terms of descending certainty values and added greedily to the ontology only if they are logically coherent with axioms added before them. Nickel et al. [18] propose a method using hierarchical clustering on a decomposed representation of the knowledge graph. Specifically, they extend *RESCAL* [17], a method for factorizing a three-way tensor, to better handle sparse large-scale data and apply *OPTICS* [3], a density based hierarchical clustering algorithm. Ristoski et al. [20] rely on entity and text embeddings in their proposed method, *TIEmb*. The intuition behind this approach is that entities of a subclass will be embedded within their parent class's embeddings. Thus if you calculate the centroid for each class's embeddings, you can infer its subclasses as those whose centroid falls within a certain radius. For instance, the class centroids of `Mammals` and `Reptiles` will fall inside the radius of `Animals` although the converse is not true since `Mammals` and `Reptiles` are more specific classes and are expected to have a smaller radius.

2.2 Methods for Tag Hierarchy Induction

Heymann and Garcia-Molina [11] propose a frequency-based approach using cosine similarity to calculate tag generality. In their approach, tags are assigned vectors based on the amount of times they annotate each document. The pairwise cosine similarity between tag vectors is used to build a tag similarity graph.

The closeness centrality of tags in this graph is used as the generality of tags. To build the hierarchy, tags are greedily added – in order of descending generality – as children to the tag in the hierarchy that has the highest degree of similarity. This approach was extended by Benz et al. [4] to better handle synonyms and homonyms in the dataset. Schmitz [23] proposed a method extending on the work done by Sanderson and Croft [22] which uses subsumption rules to identify the relations between parents and children in the hierarchy. The subsumption rules are calculated by tag co-occurrence and filtered to control for "idiosyncratic vocabulary". These rules form a directed graph which is then pruned to create a tree. Solskinnsbakk and Gulla [25] use the Aprioir algorithm [1] to mine a set of association rules from the tags. Each of these rules has the relationship of premise and consequence which the authors treat as that of class and subclass. This is used to construct a tree which is then verified based on the semantics of each tag. Tang et al. [26] use *Latent Dirichlet Allocation* (LDA) [5] to generate topics comprised of tags. Generality can then be calculated following the reasoning that tags with high frequencies across many topics are more general than ones that have a high frequencies in a single topic. Relations between tags are induced based on four divergence measures calculated on the LDA results. *Agglomerative Hierarchical Clustering for Taxonomy Construction* [14] avoids explicitly computing tag generality by employing agglomerative clustering and selecting cluster medoids to be promoted upwards in the hierarchy. Cluster medoids are chosen based on a similarity metric calculated as the divergence between a tag's topic distributions as learned by LDA. Wang et al. [29] propose a taxonomy generation method based on repeated application of k-medoids clustering. As the distance metric necessary for k-medoids clustering, they propose a similarity score based on the weighted sum of document and textual similarities. Levels in the hierarchy are created by repeated application of k-medoids clustering such that for each cluster, the cluster medoid becomes the parent of all other tags in the cluster. Dong et al. [8] propose a supervised learning approach wherein binary classifiers are trained to predict a "broader-narrower" relation between tags. LDA is used to generate topic distributions for tags which act as a basis for three sets of features used to train the classifier. This approach does not guarantee that the relations between tags will form a rooted tree.

3 Problem Description

A knowledge graph, $\mathcal{K}$, is repository of information structured as a collection of triples where each triple relates the subject, s, to the object, o, through a relation, r. More formally, $\mathcal{K} = \{\langle s, r, o \rangle \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}\}$ where $\langle s, r, o \rangle$ is a triple, $\mathcal{E}$ is the set of entities in $\mathcal{K}$, and $\mathcal{R}$ is the set of relations in $\mathcal{K}$. $\mathcal{K}$ can therefore be viewed as a directed graph with nodes representing entities and edges representing relations.

We can think of relation-object pairs, $\langle r, o \rangle$, as tags that describe the subject. In this view, each entity that takes on the role of subject, s_i, is annotated by tags, $t_j \in \mathcal{A}_i$, where $\mathcal{A}_i$ is the set of tags that annotate s_i. We call these entities documents, $d_i \in \mathcal{D}$, such that the set of all documents is a subset of all entities, $\mathcal{D} \subseteq \mathcal{E}$.

Tags are defined as relation-objects pairs, $t := \langle r, o \rangle$, and belong to the set of all tags, the vocabulary, denoted as $\mathcal{V}$, such that $t_j \in \mathcal{V}$. For a concrete example of this notation consider DBpedia, wherein the entity `dbr:Canada` is annotated by the tags ⟨`dbo:capital,dbr:Ottawa`⟩, ⟨`dbo:currency,dbr:Canadian_dollar`⟩, ⟨`rdf:type,dbo:Location`⟩, and ⟨`rdf:type,dbo:Country`⟩ amongst others. In this view, the knowledge base $\mathcal{K}$ may be represented as the set of document-tag tuples $\mathcal{K} = \{\langle d, t \rangle \in \mathcal{D} \times \mathcal{V}\}$, where $\langle d, t \rangle$ is the tuple that relates document d with tag t. We refer to this notation as the tuple structure for the remainder of the paper.

Information in knowledge graphs is often structured using an ontology, which provides semantics to the knowledge graph's triples through an axiomatic foundation which defines how entities and relations associate with one another. A key component of most ontologies is the class taxonomy which organizes classes through a set of class subsumption axioms. These subsumption axioms may be thought of as is-a relations between classes. For instance, in the DBpedia class hierarchy, the subsumption axioms {`dbo:Person` → `dbo:Artist`} and {`dbo:Artist` → `dbo:Painter`} imply that `dbo:Painter` is a `dbo:Artist` and that `dbo:Artist` is a `dbo:Person`. Furthermore, since class subsumption axioms are transitive, `dbo:Painter` is a `dbo:Person`. This taxonomy oftentimes takes the form of a rooted tree with a root class of which all other classes are considered logical descendants of.

The problem of class taxonomy induction from knowledge graphs involves generating subsumption axioms from triples to build the class taxonomy. We notice that in most knowledge graphs, subjects are related to their class type by one relation. This has the effect of reducing the knowledge graph's class identifying triples to a single dimension. The property can be exploited in the tuple structure, since all class identifying relations are the same, they can be ignored without loss of information. For instance, in DBpedia the relation which relates subjects to their class is `rdf:type`. Thus, when compiling a dataset of class identifying tuples, we can treat the tags ⟨`rdf:type,dbo:Country`⟩ and `dbo:Country` as equivalent. Therefore, the tuple ⟨`dbr:Canada`, `dbo:Country`⟩ preserves all information required to induce a class taxonomy. This can be exploited by tag hierarchy induction methods which take documents and their tags as input.

4 Approach

Our proposed method uses class frequencies and co-occurrences to calculate similarity between tags. This approach, inspired by the method proposed by Schmitz, relies on the intuition that subclasses will co-occur in documents with their superclasses more often than with classes they are not logical descendants of. Unlike Schmitz's method which uses this assumption to generate candidate subsumption axioms, our method uses similarity to choose a parent tag which already exists in the taxonomy. In this step, which draws inspiration from Heymann and Garcia-Molina, tags are greedily added to the taxonomy in order of descending generality. Thus, subsumption axioms induced by our method have to abide by the following rules: (1) the parent tag has a higher generality than the child tag; (2) the parent tag is the tag with the highest similarity to the child tag from the tags that exist in the taxonomy when the child tag is being added.

As previously mentioned, our approach leverages the tuple structure of a knowledge graph to induce a class taxonomy in the form of a rooted tree. As such, the first step is data preprocessing wherein all of a knowledge graph's class identifying triples are converted to tuple structure.

4.1 Class Taxonomy Induction Procedure

Before describing the taxonomy induction procedure for our method, we define measures which are calculated on the knowledge graph as required input for our algorithm.

- The number of documents annotated by tag t_a is denoted as D_{t_a}.
- The number of documents annotated by both tags t_a and t_b is denoted as D_{t_a,t_b}. We note that this measure is symmetrical, i.e. $\mathrm{D}_{t_a,t_b} = \mathrm{D}_{t_b,t_a}$.
- The generality of tag t_a, denoted as G_{t_a}, measures how general the concept described by the tag is and how high it belongs in the taxonomy. The generality is defined as:

$$\mathrm{G}_{t_a} = \sum_{t_b \in \mathcal{V}_{-t_a}} \frac{\mathrm{D}_{t_a,t_b}}{\mathrm{D}_{t_b}} \tag{1}$$

  Where $\mathcal{V}_{-t_a}$ is the set of all tags excluding tag t_a.

Having calculated the aforementioned measures, we proceed by sorting tags in the order of descending generality and store them as $\mathcal{V}_{sorted}$. The first element of this list, $\mathcal{V}_{sorted}[0]$, is semantically the most general of all tags and becomes the root tag of the taxonomy. The taxonomy, $\mathcal{T}$, is represented as a set of subsumption axioms between parent and child tags. Formally, each subsumption between parent tag, t_{parent}, and child tag, t_{child}, is represented by $\{t_{parent} \rightarrow t_{child}\}$ such that $\{t_{parent} \rightarrow t_{child}\} \in \mathcal{T}$. The taxonomy is therefore initialized with the root tag as $\mathcal{T} = \{\{\emptyset \rightarrow \mathcal{V}_{sorted}[0]\}\}$ where $\emptyset$ represents a null value, i.e. no parent.

Following initialization, the remaining tags are added to the taxonomy in terms of descending generality by calculating the similarity between the tag being added, t_b, and all the tags already in the taxonomy, $\mathcal{T}*$. The tag $t_a \in \mathcal{T}*$ that has the highest similarity with tag t_b becomes the parent of t_b and $\{t_a \rightarrow t_b\}$ is added to $\mathcal{T}$. The similarity between tags t_a and t_b, denoted as $\mathrm{S}_{t_a \rightarrow t_b}$, measures the degree to which tag t_b is the direct descendant of tag t_a. It is calculated as the degree to which tag t_b is compatible with tag t_a and all the ancestors of t_a:

$$\mathrm{S}_{t_a \rightarrow t_b} = \sum_{t_c \in \mathcal{P}_{t_a}} \alpha^{l_a - l_c} \frac{\mathrm{D}_{t_b,t_c}}{\mathrm{D}_{t_b}} \tag{2}$$

Where $\mathcal{P}_{t_a}$ is the path in the taxonomy from the root tag $\mathcal{V}_{sorted}[0]$ to tag t_a. l_a and l_c denote the levels in the hierarchy of tags t_a and t_c, respectively. The levels are counted from the root tag starting at zero. Thus, the level of $\mathcal{V}_{sorted}[0]$, denoted as $l_{\mathcal{V}_{sorted}[0]}$, is equal to zero, the levels of its children are equal to one, and so on. The decay factor, α, is a hyperparameter that controls the effect ancestors of tag t_a have on its similarity when calculating $\mathrm{S}_{t_a \to t_b}$. By setting the value of α such that $0 < \alpha < 1$, we ensure that the effect is lower the more distant an ancestor tag is. The cases were $\alpha = 0$ and $\alpha = 1$ correspond to ancestors having no effect and equal effect on the similarity, respectively. We explore the effect various α values have on the induced class taxonomy in the following section. The full details of our method's procedure are outlined in Algorithm 1.

Algorithm 1 Procedure for Class Taxonomy Induction

Input: Knowledge graph in tuple structure, $\mathcal{K}$; Document counts annotated by tag, D; Generality of tags, G; Decay factor, α
Output: Induced class taxonomy, $\mathcal{T}$

```
 1: Sort tags in order of descending generality, $\mathcal{V}_{sorted}$
 2: Initialize taxonomy with root tag equal to the tag with highest generality, $\mathcal{T} = \{\{\emptyset \to \mathcal{V}_{sorted}[0]\}\}$
 3: Initialize the set of tags that have already been added to the taxonomy, $\mathcal{T}* = \{\mathcal{V}_{sorted}[0]\}$
 4: for b = 1, 2, ..., $|\mathcal{V}_{sorted}|$ do
 5:     $maxSimTag = \mathcal{V}_{sorted}[0]$
 6:     $maxSimValue = 0$
 7:     for $t_a \in \mathcal{T}*$ do
 8:         Calculate $\mathrm{S}_{t_a \to t_b}$ using Equation 2
 9:         if $\mathrm{S}_{t_a \to t_b} > maxSimValue$ then
10:             $maxSimTag = t_a$
11:             $maxSimValue = \mathrm{S}_{t_a \to t_b}$
12:         end if
13:     end for
14:     $\mathcal{T} = \{maxSimTag \to t_b\} \cup \mathcal{T}$
15:     $\mathcal{T}* = t_b \cup \mathcal{T}*$
16: end for
```

5 Evaluation

Evaluation of class taxonomy induction methods is difficult as there may be several equally valid taxonomies for a dataset. Previous works such as Gu et al. [10] and Wang et al. (2009) [30] have opted for human evaluation, wherein domain experts assess the correctness of relations between classes. Wang et al. (2012) [29] used domain experts to rank entire paths on a three point scale. Others, such as Liu et al. [15] and Almoqhim et al. [2], compare class relations against a gold standard taxonomy. In this approach, a confusion matrix between class subsumption axioms is calculated between the induced and gold standard taxonomies. When a gold standard taxonomy can be established, it is the preferred evaluation method as it provides an objective measurement; as such, it is the one we use in our work. We use the confusion matrix to derive the harmonic mean between precision and recall, the F_1 score [7], as our evaluation metric:

$$precision = \frac{TP}{TP + FP} \tag{3}$$

$$recall = \frac{TP}{TP + FN} \tag{4}$$

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{5}$$

Where TP, FP, and FN are the number of true positives, false positives, and false negatives, respectively.

For the remainder of this section, we first evaluate the effect of our method's hyperparameter, α, on each of the three datasets and provide suggestions for selecting the α value when applying our method to other datasets. This is followed by a comparing our method to the aforementioned Heymann and Garcia-Molina method, Schmitz method, as well as results from the literature. We also provide visualizations of excerpts from the class taxonomies induced by our method on the Life and DBpedia datasets. Finally, our method's computational complexity and the effect of dataset size on induced taxonomies are evaluated. The method was implemented using Python and has been made public alongside our datasets for reproducibility on Github[3].

5.1 Datasets

We evaluate the method on three real-world datasets generated from public online knowledge bases: Life, DBpedia, and WordNet. All three datasets as well as their respective gold standard class taxonomies were generated during the month of November 2019.

The Life dataset was generated by querying the Catalogue of Life: 2019 Annual Checklist (CoL) [21], an online database that indexes living organisms by their taxonomic classification. One hundred thousand living organisms were randomly selected from the GBIF Type Specimen Names [9], an online checklist of 1,226,904 organisms, and queried on CoL at each of their taxonomic ranks to generate the document-tag tuples. The resulting dataset takes the form such that each organism is a document and its membership at each taxonomic rank is a tag related by `is-a`. For instance, the document `Canis_latrans` (coyote) will have the tags ⟨`is-a, Mammalia`⟩ and ⟨`is-a, Canidae`⟩. Furthermore, to anchor the class taxonomy to a root tag, we added the tag ⟨`is-a, LivingOrganism`⟩

[3] https://github.com/mpietrasik/smict

to every document. We note that even though the number of taxonomic ranks is fixed, most organisms in the database are not defined on all of them. As such, the number of tags per document varies from two to ten. In total, there are 100,000 documents and 37,368 unique tags. Since the dataset itself is classified in the correct taxonomic order, the Life gold standard taxonomy could simply be obtained by querying for subsumption axioms from the dataset.

The DBpedia dataset was generated by randomly querying for 50,000 unique subjects in DBpedia for which there exists a triple where the subject is related to a DBpedia class object (an object having the prefix `dbo:`) via the relation `rdf:type`. These 50,000 subjects become the documents in the tuple structure. Following this step, all the triples for each document having the tag form ⟨`rdf:type, dbo:*`⟩ were queried to make the document-tag tuples. (`dbo:*` represents any object with the prefix `dbo`.) In total, 205,793 triples were used to create the dataset with 418 unique tags. The DBpedia gold standard taxonomy was generated from the DBpedia ontology class mappings which can be found on the DBpedia website[4]. At the time of querying, the ontology had 765 classes, 418 of which were present in the dataset. This difference made it necessary to include only those subsumption axioms for which parent and child tags exist in the dataset when computing the confusion matrix. This is similar to the dataset generated in Ristoski et al. where the number of classes present in their dataset was 415.

The WordNet dataset was generated by querying DBpedia for subjects of types that exist in WordNet [16], an English language lexical database. Fifty thousand subjects having a WordNet class object related by `rdf:type` were queried. In DBpedia, WordNet class objects use the `yago:` prefix, giving the tag format ⟨`rdf:type, yago:*`⟩. This process yielded a dataset comprised of 50,000 documents and 1752 unique tags generated from 392,846 triples. To generate the WordNet gold standard taxonomy, DBpedia was queried to learn the relations between WordNet classes through the `rdfs:subClassOf` relation. In this process, `yago:PhysicalEntity100001930` is set as the root class and the taxonomy is built by recursively querying for subclasses using `rdfs:subClassOf` as the relation. This process builds a taxonomy of 30722 tags. To fit the 1752 tags present in the dataset, it was necessary to collapse the gold standard taxonomy. This was done by removing tags in the gold standard taxonomy that are missing in the dataset and adopting orphaned tags with the nearest ancestor existing in the dataset.

5.2 Hyperparameter Sensitivity

We evaluate our method's sensitivity to the decay factor, α, by performing a hyperparameter sweep on each of the three datasets. In this process, our method is

[4] http://mappings.dbpedia.org/server/ontology/classes/

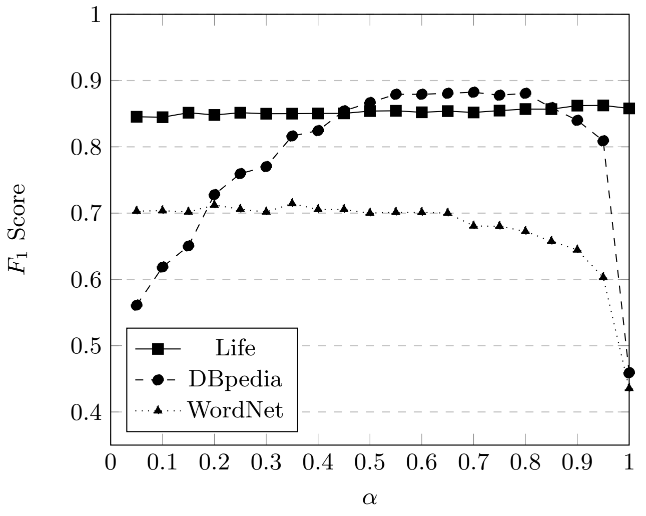

Fig. 1: Comparison of mean test F_1 Scores at varying α values on the Life, DBpedia, and WordNet datasets.

applied five times on each dataset for α values starting at $\alpha = 0$ and increasing by increments of 0.05 up until $\alpha = 1$. This process is analogous to increasing the relative importance of ancestor tags when calculating tag similarity. Furthermore, since similarity is calculated as a summation, increasing α will favour tags lower in the taxonomy. The F_1 scores are calculated and their means at each α value are displayed graphically in Figure 1. For clarity, we omit graphing the mean F_1 scores at $\alpha = 0$ as the values are disproportionately low for all three datasets ($F_1 < 0.1$). This is because when $\alpha = 0$, the similarity gets reduced to $\mathrm{S}_{t_a \to t_b} = \mathrm{D}_{t_a,t_b}/\mathrm{D}_{t_b}$ which has the effect of inducing shallow taxonomies with most tags as children of the root tag.

Upon cursory inspection of the F_1 scores, we notice that there is no clear behaviour that α exhibits which is constant across datasets. This is also apparent when comparing the optimal α values: 0.95, 0.70, and 0.35 for Life, DBpedia, and WordNet datasets, respectively. Furthermore, we notice that as α increases, the trend follows three different patterns: stable, generally increasing, and generally decreasing. A possible reason for the relative stability of α on the Life dataset is its consistency. Due to the strict requirements for source datasets to be included in CoL, all entries are well scrutinised. As such, tags will always appear with their ancestors in the same documents. For example, all 893 instances of the tag `Mammalia` co-occur with the tag's ancestors `Animalia`, `Chordata`, and `LivingOrganism`. In this scenario, there is less information to be gained by incorporating information from higher up in the taxonomy. On the other hand, the DBpedia dataset shows improvement with increasing α values until a peak is reached and F_1 declines. The increase in induced taxonomy quality with increasing α values in consistent with the assumption that taking into account a potential parent's path is advantageous when selecting a parent. The decline in F_1 after $\alpha = 0.8$ can be explained by distant ancestor tags having too strong an influence in assigning parent tags to children. One possible explanation for better F_1 scores of lower α values on WordNet is our method's overall lower F_1 scores on this dataset. Errors in the induced taxonomy propagate downwards and their effect increases with the value of α. Thus, in a taxonomy with many errors, it is advantageous to place a relatively higher value on the similarity between the direct parent tag and its child, as is done with lower α values.

In general, it is difficult to predict the optimal α value a priori, however there are a few rules of thumb to guide this process when applying our method. When there is no prior information about a nature of the dataset or its expected class taxonomy, we suggest using α values around 0.5 as these values perform well (although not optimally) in our experiments. Datasets which are complex, or have low co-occurence rates between ancestor and descendent tags will favour lower α values as these ensure errors will propagate less through the taxonomy. On the other hand, well structured datasets will be less affected by varying α values.

5.3 Results

In our experiments, we applied our proposed method to each of the aforementioned datasets at the α values determined optimal in the previous subsection. Each dataset was applied five times to account for the stochasticity in sorting tags of equal generality. The results of our method as well as those of the comparison methods are summarized in Table 1. We implemented Heymann and Garcia-Molina, and Schmitz methods to the best of our understanding and performed hyperparameter exploration for their respective hyperparameters on each dataset. After obtaining the optimal hyperparameters, we ran the methods five times on each dataset and collected the results. We note that Heymann and Garcia-Molina was not able to terminate sufficiently fast enough for us to obtain results on the Life dataset. In the table we also included the results reported in previous work applied on the DBpedia dataset. Although the DBpedia dataset was derived similarly to our own, conclusions in comparing this method to our proposed method should be drawn cautiously. We indicate these entries in the table with a footnote.

In general, our method outperforms the other two tag hierarchy induction methods as shown by the mean F_1 scores. We notice similarly high precision and recall values which suggests that it's both capable of inducing subsumption axioms (recall) while ensuring these axioms are correct (precision). Furthermore, closer inspection of the results reveals that many of the errors can be categorized by two types, which we illustrate by using results from the DBpedia dataset. In the first, the order between parent and child tags are reversed as in the induced {`dbo:Guitarist` → `dbo:Instrumentalist`} when the correct order is {`dbo:Instrumentalist` → `dbo:Guitarist`}. In the second, a tag is misplaced as the child of its sibling, for instance, the gold standard classification of educational institutions is {{`dbo:EducationalInstitution` → `dbo:University`}, {dbo:EducationalInstitution $\rightarrow$ dbo:College}} while our induced taxonomy gives the following: {{dbo:EducationalInstitution $\rightarrow$ dbo:University}, {dbo:University $\rightarrow$ dbo:College}}. Finally, our induced taxonomy includes subsumption axioms which are considered incorrect as per the gold standard but may not be to a human evaluator. An example of this is that our method induced the subsumption axiom {dbo:SportFacility $\rightarrow$ dbo:Stadium} while the gold standard considers {dbo:Venue $\rightarrow$ dbo:Stadium} to be the correct parent for dbo:Stadium. We provide an excerpt of our induced class taxonomies on the Life and DBpedia datasets in Figure 2.

Table 1: Method results (mean ± standard deviation) on the Life, DBpedia, and WordNet datasets.

	Life			**DBpedia**			**WordNet**		
Method	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
Heymann and	–	–	–	.7944	.8021	.7982	.6027	.5814	.5918
Garcia-Molina	–	–	–	±.01483	±.01500	.01490	±.0116	±.0112	±.0114
Schmitz	.8936	.7966	.8423	.8063	.7962	.8013	.8140	.7756	.7943
	±0	±0	±0	±0	±0	±0	±0	±0	±0
Paulheim and	–	–	–	.1040	.2190	.1410	–	–	–
Fümkranz[5] [19, 20]	–	–	–	–	–	–	–	–	–
Ristoski	–	–	–	.5940	.4650	.5210	–	–	–
et al. [20] [5]	–	–	–	–	–	–	–	–	–
Völker and	–	–	–	.9920	.9970	.9950	–	–	–
Niepert [27] [5]	–	–	–	–	–	–	–	–	–
Our method	.8740	.8513	.8625	.8781	.8867	.8824	.7275	.7018	.7144
	±.0041	±.0040	±.0040	±.0051	±.0052	±.0052	±.0070	±.0068	±.0069

5.4 Computational Complexity Evaluation

One of the most salient issues that arises when applying class taxonomy induction methods to real-world knowledge graphs is that of scalability. As mentioned previously, DBpedia, Yago, and WikiData have upwards of one billion triples each, thus for a method to operate on these datasets, it has to be computationally efficient. It is important to note, however, that in inducing a class taxonomy, it is not necessary to use all the triples available in the knowledge graph but rather to only use as many as is required to achieve an acceptable result. We discuss this idea in the following subsection.

The most computationally taxing procedure in our method is that of calculating the number of documents annotated by two tags, D_{t_a,t_b}, which has a worst case time complexity of $\mathcal{O}(|\mathcal{D}||\mathcal{V}|^2)$, where $|\mathcal{D}|$ and $|\mathcal{V}|$ are the number of documents and tags, respectively. It is important to note, however, that the

[5] The result for this method was obtained from the literature.

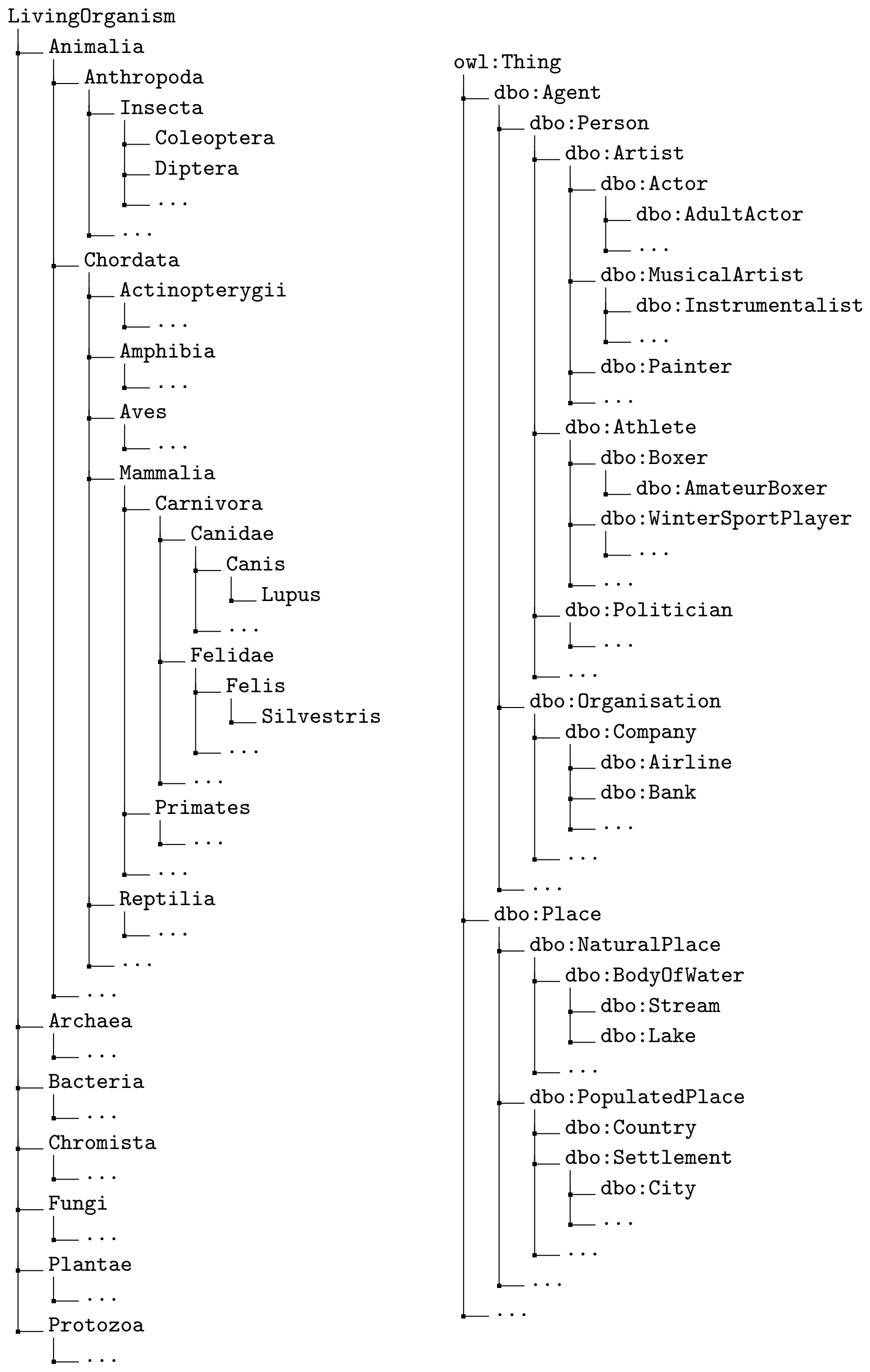

Fig. 2: Excerpts of the induced class taxonomies for the Life (left) and DBpedia (right) datasets. Ellipses denote addition child classes omitted for brevity.

worst case only occurs when all documents are annotated by all tags. In this scenario, every subject in a knowledge graph is of every class type in the ontology. The average computation complexity of our algorithm is $\mathcal{O}(|\mathcal{D}|\overline{|\mathcal{A}|}^2)$ where $\overline{|\mathcal{A}|}$ is the average number of tags that annotate a document. In our experiments our method was faster to terminate than both the Heymann and Garcia-Molina and Schmitz methods on all three datasets.

5.5 Effect of Dataset Size on Induced Taxonomy

As mentioned previously, although a method's scalability to large knowledge graphs is important in the context of the Semantic Web, it's not the case that larger datasets will produce better taxonomies. To demonstrate this, we applied our method as described in the Results subsection to DBpedia datasets at differing document counts. Each dataset was derived the same way as described in the Datasets subsection, such that all of the smaller DBpedia datasets are strict subsets of the larger ones. A summary of the results is displayed in Table 2. We note that runtime measures the execution of our method without including time for input and output. We notice that although larger datasets obtain higher F_1 scores, the incremental increase in F_1 diminishes, and the scores plateau after 20,000 documents. However, relying on F_1 score as the sole comparison metric may be misguiding since it is calculated on the tags which exist in the dataset. Thus since there are 211 unique tags in the DBpedia 1,000 dataset and 428 unique tags in the DBpedia 100,000 dataset, the induced taxonomy of the latter will be over twice as large as the former.

Table 2: Summary of our method's results on DBpedia datasets at various document counts, $|\mathcal{D}|$.

$\|\mathcal{D}\|$	$\|\mathcal{V}\|$	# of Triples	Optimal α	Runtime (sec)	F_1
100000	428	422860	0.65	1.6311	0.8810
90000	427	379444	0.65	1.5131	0.8808
80000	425	336084	0.45	1.3340	0.8826
70000	424	292791	0.55	1.1248	0.8847
60000	423	249383	0.70	0.9767	0.8783
50000	418	205793	0.70	0.8556	0.8824
40000	414	164470	0.70	0.6545	0.8783
30000	408	123408	0.55	0.5564	0.8716
20000	392	82381	0.65	0.3652	0.8791
10000	365	41081	0.65	0.2001	0.8425
5000	326	20481	0.70	0.1161	0.8354
2500	284	10330	0.60	0.0670	0.8372
1000	211	4097	0.35	0.0280	0.7632

6 Conclusions

In this paper, we described the problem of inducing class hierarchies from knowledge graphs and its significance to the Semantic Web community. In our contribution to this research area, we proposed an approach to the problem by marrying the fields of class taxonomy induction from knowledge graphs with tag hierarchy induction from documents and tags. To this end, we reshaped the knowledge graph to a tuple structure and applied two existing tag hierarchy induction methods to show the viability of such an approach. Furthermore, we proposed a novel method for inducing class taxonomies that relies solely on class frequencies and co-occurrences and can thus be applied on knowledge graphs irrespective of their content. We showed our method's ability to induce class hierarchies by applying it on three real-world datasets and evaluating it against their respective gold standard taxonomies. Results demonstrate that our method induces better taxonomies than other tag hierarchy induction methods and can be reliably applied to large-scale knowledge graphs.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proc. 20th int. conf. very large data bases, VLDB. vol. 1215, pp. 487–499 (1994)
2. Almoqhim, F., Millard, D.E., Shadbolt, N.: Improving on popularity as a proxy for generality when building tag hierarchies from folksonomies. In: International Conference on Social Informatics. pp. 95–111. Springer (2014)
3. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: Optics: ordering points to identify the clustering structure. In: ACM Sigmod record. vol. 28, pp. 49–60. ACM (1999)
4. Benz, D., Hotho, A., Stützer, S., Stumme, G.: Semantics made by you and me: Self-emerging ontologies can capture the diversity of shared knowledge (2010)
5. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of machine Learning research **3**(Jan), 993–1022 (2003)
6. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. pp. 1247–1250. AcM (2008)
7. Chinchor, N.: Muc-4 evaluation metrics. In: Proceedings of the 4th conference on Message understanding. pp. 22–29. Association for Computational Linguistics (1992)
8. Dong, H., Wang, W., Coenen, F.: Learning relations from social tagging data. In: Pacific Rim International Conference on Artificial Intelligence. pp. 29–41. Springer (2018)
9. Döring, M.: Gbif type specimen names (2017), https://doi.org/10.15468/sl9pyf
10. Gu, C., Yin, G., Wang, T., Yang, C., Wang, H.: A supervised approach for tag hierarchy construction in open source communities. In: Proceedings of the 7th Asia-Pacific Symposium on Internetware. pp. 148–152. ACM (2015)
11. Heymann, P., Garcia-Molina, H.: Collaborative creation of communal hierarchical taxonomies in social tagging systems. Tech. rep. (2006)

12. Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: Yago2: A spatially and temporally enhanced knowledge base from wikipedia. Artificial Intelligence **194**, 28–61 (2013)
13. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al.: Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web **6**(2), 167–195 (2015)
14. Li, X., Wang, H., Yin, G., Wang, T., Yang, C., Yu, Y., Tang, D.: Inducing taxonomy from tags: An agglomerative hierarchical clustering framework. In: International Conference on Advanced Data Mining and Applications. pp. 64–77. Springer (2012)
15. Liu, K., Fang, B., Zhang, W.: Ontology emergence from folksonomies. In: Proceedings of the 19th ACM international conference on Information and knowledge management. pp. 1109–1118. ACM (2010)
16. Miller, G.A.: Wordnet: a lexical database for english. Communications of the ACM **38**(11), 39–41 (1995)
17. Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. (2011)
18. Nickel, M., Tresp, V., Kriegel, H.P.: Factorizing yago: scalable machine learning for linked data. In: Proceedings of the 21st international conference on World Wide Web. pp. 271–280. ACM (2012)
19. Paulheim, H., Fümkranz, J.: Unsupervised generation of data mining features from linked open data. In: Proceedings of the 2nd international conference on web intelligence, mining and semantics. p. 31. ACM (2012)
20. Ristoski, P., Faralli, S., Ponzetto, S.P., Paulheim, H.: Large-scale taxonomy induction using entity and word embeddings. In: Proceedings of the International Conference on Web Intelligence. pp. 81–87. ACM (2017)
21. Roskov Y., Ower G., O.T.N.D.B.N.K.P.B.T.D.R.D.W.N.E.v.Z.J.P.L.e.: Species 2000 & itis catalogue of life, 2019 annual checklist. (2019)
22. Sanderson, M., Croft, B.: Deriving concept hierarchies from text. In: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. pp. 206–213. ACM (1999)
23. Schmitz, P.: Inducing ontology from flickr tags. In: Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland. vol. 50, p. 39 (2006)
24. Singhal, A.: Introducing the knowledge graph: things, not strings (2012), https://www.blog.google/products/search/introducing-knowledge-graph-things-not/
25. Solskinnsbakk, G., Gulla, J.A.: A hybrid approach to constructing tag hierarchies. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". pp. 975–982. Springer (2010)
26. Tang, J., Leung, H.f., Luo, Q., Chen, D., Gong, J.: Towards ontology learning from folksonomies. In: Twenty-First International Joint Conference on Artificial Intelligence (2009)
27. Völker, J., Niepert, M.: Statistical schema induction. In: Extended Semantic Web Conference. pp. 124–138. Springer (2011)
28. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledge base (2014)
29. Wang, S., Lo, D., Jiang, L.: Inferring semantically related software terms and their taxonomy by leveraging collaborative tagging. In: 2012 28th IEEE International Conference on Software Maintenance (ICSM). pp. 604–607. IEEE (2012)
30. Wang, W., Barnaghi, P.M., Bargiela, A.: Probabilistic topic models for learning terminological ontologies. IEEE Transactions on Knowledge and Data Engineering **22**(7), 1028–1040 (2009)