Semantic Web 0 (0) 1
IOS Press

A simple method for inducing class taxonomies in knowledge graphs

Marcin Pietrasik [a,*], and Marek Z. Reformat [a,b]

[a] *Department of Electrical and Computer Engineering, University of Alberta, Edmonton AB, Canada*
E-mail: pietrasi@ualberta.ca

[b] *Information Technology Institute, University of Social Sciences, 90-113 Łódź, Poland*

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 form 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. Furthermore, we show that the induced hierarchy may be used as a foundation for hierarchical clustering of knowledge graph subjects. We demonstrate the use of our method on four 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, clustering

1. Introduction

Knowledge graphs are data structures that use principles of 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 [1], YAGO [2], and WikiData [3] 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. 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 [4] to enhance their search engine results by providing infoboxes which summarize facts retrieved as due to a user's query [5].

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 vocabulary and a rule book that provides semantics to a knowledge graph and governs how the information contained within it is represented and how 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

*Corresponding author. E-mail: pietrasi@ualberta.ca.

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 based on their content. 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 selected and organized. 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 identifying 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. Using the new structure, we construct a novel approach to inducing class taxonomies which outperforms existing tag hierarchy induction methods both in terms scalability and quality of induced taxonomies. Finally, we show that an induced class taxonomy may be used as the foundation for performing hierarchical clustering on the knowledge graph's subjects. The idea behind this is that each class in the taxonomy may serve as a hierarchical cluster, reducing the clustering procedure to merely assigning each entity to one class in the taxonomy. Empirical evaluation demonstrates that this process constructs coherent hierarchical clusters.

The remainder of this paper proceeds with Section 2 which provides an overview of the existing work done on inducing class taxonomies, tag hierarchies, and cluster hierarchies. We formalize the problem and introduce notation in Section 3. Our proposed method is described in Section 4 and evaluated in Section 5. Section 6 concludes the paper.

[1] https://www.wikipedia.org/

[2] https://wordnet.princeton.edu/

2. Related work

We divide our discussion of related work into three subsections: class taxonomy induction methods, tag hierarchy induction methods, and hierarchical clustering methods for knowledge graphs. The first two 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. Hierarchical clustering methods seek to learn clusters of knowledge graph entities based on shared semantics and organize them hierarchically such that descendant clusters contain more specific instances of their corresponding ancestors.

2.1. Methods for class taxonomy induction

Völker and Niepert [6] 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. [7] propose a method using hierarchical clustering on a decomposed representation of the knowledge graph. Specifically, they extend *RESCAL* [8], a method for factorizing a three-way tensor, to better handle sparse large-scale data and apply *OPTICS* [9], a density based hierarchical clustering algorithm.

Ristoski et al. [10] 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 decreasing generality – as children to the tag in the hierarchy that has the highest degree of similarity. This approach was extended by Benz et al. [12] to better handle synonyms and homonyms in the dataset.

Schmitz [13] unveils a method extending on the work done by Sanderson and Croft [14] 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 [15] use the Aprioir algorithm [16] 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.

The application of *Latent Dirichlet Allocation* (LDA) [17] to generate topics comprised of tags is proposed in Tang et al. [18]. 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* [19] 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. [20] introduce 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.

A supervised learning approach is used in Dong et al. [21] where 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.

2.3. Methods for hierarchical clustering

In an early method, Roy et al. [22] sample a graph from a generative model in a fashion reminiscent of blockmodeling. The model is learned by performing inference on its parameters via the Metropolis-Hastings algorithm. A consequence of this process is the generation of a tree describing entity similarity. Nickel et al. [7] perform hierarchical clustering on latent representations learned by the aforementioned *RESCAL* method. Using these latent representations has the advantage of being agnostic to the underlying hierarchical clustering method used, allowing for flexibility to adapt to different data.

In an approach which bears similarity to our own, Chen and Reformat [23] describe each subject in a knowledge graph by its relation-object pairs. These pairs are then used to calculate a similarity matrix between subjects on which agglomerative hierarchical clustering is performed using the extended Ward's minimum variance [24] as its measure. Mohamed [25] takes a similar approach wherein subjects which are described by the same relation-object pairs are assigned to the same groups. The similarity between these groups is then calculated to construct a hierarchy.

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, s. 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 $\mathcal{V}$, i.e., $t_j \in \mathcal{A}_i$ and $\mathcal{A}_i \subseteq \mathcal{V}$. 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* $\rightarrow$ *dbo:Artist*$\}$ and $\{$*dbo:Artist* $\rightarrow$ *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 decreasing generality. Thus, subsumption axioms induced by our method have to abide by the following rules:

- The parent tag has a higher generality than the child tag.
- 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.

We can populate the induced class taxonomy with documents, which has the effect of hierarchically clustering the knowledge graph's subject entities. The process for this is to merely find the class, in the hierarchy, to which the document belongs to and assign it to that class. We can then treat each class as a cluster and its constituent documents as cluster elements. The result of this is a hierarchical structure of clusters annotated by tags and with strong inheritance properties.

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

$$G_a = \sum_{t_b \in \mathcal{V}_{-t_a}} \frac{D_{a,b}}{D_b} \tag{1}$$

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 decreasing 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 decreasing 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 $\{a \rightarrow b\}$ is added to $\mathcal{T}$. The similarity between tags t_a and t_b, denoted as $S_{a\rightarrow b}$, 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:

$$S_{a\rightarrow b} = \sum_{t_c \in \mathcal{P}_a} \alpha^{l_a - l_c} \frac{D_{b,c}}{D_b} \tag{2}$$

Where $\mathcal{P}_a$ 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 $S_{a\rightarrow b}$. 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 in a form of sets $\mathcal{D}$ and $\mathcal{V}$; document counts annotated by tag(s) $D_{i(,j)}$; generality of tags G_i; decay factor α
Output: induced class taxonomy subsumption axioms $\mathcal{T}$ and $\mathcal{T}*$

Sort tags in order of decreasing generality G_i, create $\mathcal{V}_{sorted}$
Initialize taxonomy with root tag equal to the tag with highest generality, $\mathcal{T} = \{\{\emptyset \rightarrow \mathcal{V}_{sorted}[0]\}\}$
Initialize the set of tags that have already been added to the taxonomy, $\mathcal{T}* = \{\mathcal{V}_{sorted}[0]\}$
for b = 1, 2, ..., $|\mathcal{V}_{sorted}|$ **do**
    $t_b = \mathcal{V}_{sorted}[b]$
    $maxSimTag = \mathcal{V}_{sorted}[0]$
    $maxSimValue = 0$
    for $t_a \in \mathcal{T}*$ **do**
        Calculate $S_{a\rightarrow b}$ using Equation 2
        if $S_{a\rightarrow b} > maxSimValue$ **then**
            $maxSimTag = t_a$
            $maxSimValue = S_{a\rightarrow b}$
        end if
    end for
    $\mathcal{T} = \{maxSimTag \rightarrow t_b\} \cup \mathcal{T}$
    $\mathcal{T}* = t_b \cup \mathcal{T}*$
end for

4.2. Hierarchical clustering procedure

We can use the induced taxonomy as the foundation of a hierarchical clustering of documents, i.e. the knowledge graph's subject entities. The taxonomy is used to initialize the clusters such that each tag in the taxonomy becomes a cluster and the hierarchical relations between tags are extended to the clusters. The tags may then be seen as annotations for each cluster. We exploit this in our notation such that c_a is the cluster initialized from tag t_a. Documents are assigned to clusters by the degree to which they belong to a cluster. Belonging of document d_i to cluster c_a, denoted $\mathrm{B}_{i\to a}$, is calculated as the Jaccard coefficient between the document's tags, $\mathcal{A}_i$, and the tags encountered in the path from the root cluster to cluster c_a, denoted $\mathcal{P}_a$. Formally, this is:

$$\mathrm{B}_{i\to a} = \frac{|\mathcal{A}_i \cap \mathcal{P}_a|}{|\mathcal{A}_i \cup \mathcal{P}_a|} \tag{3}$$

Each document is added to the cluster to which it has the highest degree of belonging. We denote the set of documents that belong to cluster c_a as $\mathcal{C}_a$. The process of assigning documents to clusters may be parallelized to increase performance.

This process may induce a hierarchy containing empty clusters which need to get pruned. Pruning is performed by traversing the hierarchy depth first and removing all empty clusters. In addition, non-empty clusters which have empty parent clusters are reattached as the children of their first non-empty ancestor. If a non-empty cluster has no non-empty ancestors, it becomes the child of the root. The root cluster is never removed, regardless of whether it is empty or not. The hierarchical clustering process is summarized in Algorithm 2.

Algorithm 2 Procedure for Hierarchical Clustering

Input: knowledge graph in tuple structure in a form of sets $\mathcal{D}$ and $\mathcal{V}$; class taxonomy as subsumption axioms $\mathcal{T}$ and $\mathcal{T}*$; paths in hierarchy to clusters $\mathcal{P}_a$; decay factor α

Output: cluster hierarchy as subsumption axioms $\mathcal{T}$; cluster members $\mathcal{C}$

```
1:  Initialize all clusters C as empty
2:  for d_i ∈ D do
3:     maxBelClus = None
4:     maxBelValue = 0
5:     for c_a ∈ T* do
6:        Calculate B_{i→a} using Equation 3
7:        if B_{i→a} > maxBelValue then
8:           maxBelClus = c_a
9:           maxBelValue = B_{i→a}
10:       end if
11:    end for
12:    C_maxBelClus = C_maxBelClus ∪ d_i
13: end for
14: Prune cluster hierarchy defined by T and C recursively
```

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. [26] and Wang et al. (2009) [27] have opted for human evaluation, wherein domain experts assess the correctness of relations between classes. Wang et al. (2012) [20] used domain experts to rank entire paths on a three point scale. Others, such as Liu et al. [28] and Almoqhim et al. [29], 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 [30], as our evaluation metric:

$$precision = \frac{TP}{TP+FP} \tag{4}$$

$$recall = \frac{TP}{TP+FN} \tag{5}$$

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{6}$$

Where TP, FP, and FN are the number of true positives, false positives, and false negatives, respectively. Since the F_1 score is also used in evaluating the quality of the cluster hierarchy, we use the notation Tax-F_1 to refer to the F_1 score calculated on the induced and gold standard taxonomies.

We evaluate the hierarchical clustering by calculating the F_1 score of: the belonging of documents to clusters (Doc-F_1); and how well clusters represent the tags from in the vocabulary (Tag-F_1). Doc-F_1 and Tag-F_1 highlight the trade-off between large, heterogeneous clusters on a strongly heritable hierarchy (favoured by Doc-F_1) and smaller homogeneous clusters on a less heritable hierarchy (favoured by Tag-F_1). For obtaining the former, each cluster inherits all the documents of its descendant clusters and the F_1 score is calculated such that a document is correctly assigned to a cluster if both document and cluster are annotated by the same tag. The latter is obtained in a way similar to the technique used in [7]. As before, each cluster inherits all the documents of its descendant clusters and the F_1 score between each tag and each cluster is calculated. The F_1 that is highest among the clusters becomes the score of the tag.

For the remainder of this section, we first evaluate the effect of our method's hyperparameter, α, on each of the four datasets and provide suggestions for selecting the α value when applying our method to other datasets. This is followed by a comparison 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, DBpedia, and IIMB 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][4].

5.1. Datasets

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

5.1.1. Life

The Life dataset was generated by querying the Catalogue of Life: 2019 Annual Checklist (CoL) [31], 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 [32], 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 $\langle$*is-a, Mammalia*$\rangle$ and $\langle$*is-a, Canidae*$\rangle$. Furthermore, to anchor the class taxonomy to a root tag, we added the tag $\langle$*is-a, LivingOrganism*$\rangle$ 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.

5.1.2. DBpedia

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 $\langle$*rdf:type, dbo:**$\rangle$ 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 taken from the DBpedia ontology class mappings which can be found on the DBpedia website[5]. 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. [10] where the number of classes present in their dataset was 415.

5.1.3. WordNet

The WordNet dataset was generated by querying DBpedia for subjects of types that exist in WordNet [33], 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 $\langle$*rdf:type, yago:**$\rangle$. 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

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

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

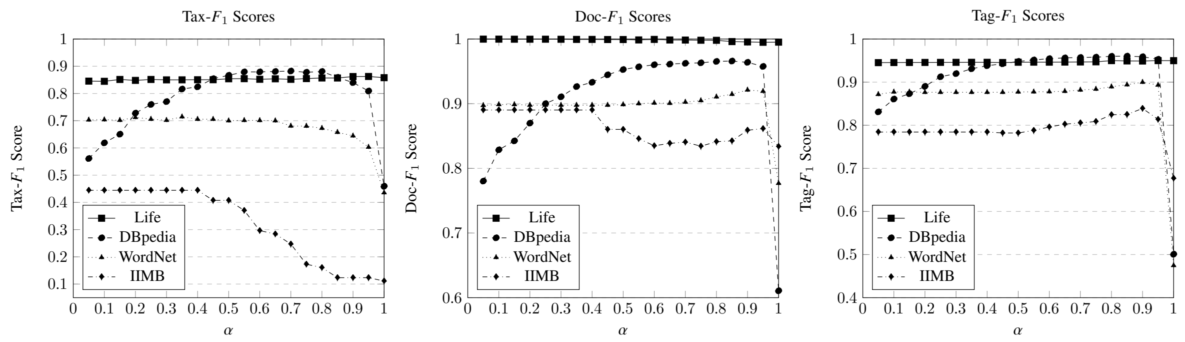

Fig. 1. Sensitivity to α as per Tax-F_1, Doc-F_1, and Tag-F_1 on the Life, DBpeida, WordNet, and IIMB datasets.

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.

5.1.4. IIMB

The IIMB dataset [34] is a benchmark created by the 2010 Ontology Alignment Evaluation Initiative to evaluate instance matching techniques and tools. The dataset contains 1416 documents and 4793 unique tags which describe facts about popular movies including: titles, genres, actors, locations, etc. The dataset is structured into five top-level tags to which all other tags belong: *Location*, *Language*, *Film*, *Creature*, and *Budget*. We manually added a root tag to anchor the dataset such that there are 82 unique tags in total.

5.2. Hyperparameter sensitivity

We evaluate our method's sensitivity to the decay factor, α, by performing a hyperparameter sweep on each of the four datasets. In this process, our method is applied five times on each dataset for α values starting at $\alpha = 0.05$ and increasing by increments of 0.05 up until $\alpha = 0.95$. 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 placing 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 four datasets ($F_1 < 0.1$). This is because when $\alpha = 0$, the similarity gets reduced to $\mathrm{S}_{a \to b} = \mathrm{D}_{a,b}/\mathrm{D}_b$ which has the effect of inducing shallow taxonomies with most tags as children of the root tag.

For class taxonomy induction, cursory inspection of the Tax-F_1 scores shows 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, 0.35, and 0.4 for Life, DBpedia, WordNet, and IIMB 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 Tax-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 Tax-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 Tax-F_1 scores of lower α values on WordNet and IIMB is our method's overall lower Tax-F_1 scores on these datasets. 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.

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

Method	Life	DBpedia	WordNet	IIMB
Heymann and Garcia-Molina	–	0.7982 ± 0.0159	0.5918 ± 0.0114	0.2025 ± 0.0068
Schmitz	0.8423 ± 0.0000	0.8013 ± 0.0000	0.7943 ± 0.0000	0.5211 ± 0.0000
Paulheim and Fümkranz[6] [10, 35]	–	0.1410	–	–
Ristoski et al.[6] [10]	–	0.5210	–	–
Völker and Niepert[6] [6]	–	0.9950	–	–
Our method	0.8625 ± 0.0040	0.8824 ± 0.0052	0.7144 ± 0.0069	0.4444 ± 0.0000

Broadly, the measures of performance of hierarchical clustering, Doc-F_1 and Tag-F_1, follow a similar pattern to Tax-F_1. The main reason for this is that hierarchical clustering is initialized by taxonomy induction. As such, errors present in the taxonomy propagate to the clustering procedure. We note two exceptions to this: Doc-F_1 and Tag-F_1 scores on the WordNet dataset; and Tag-F_1 scores on the IIMB dataset. The former exception does not show decline in clustering performance at $\alpha > 0.7$, despite initialization by lower quality taxonomies. We hypothesize that this is due the fact that many errors in the WordNet taxonomy occur at lower levels which get pruned during hierarchical clustering and therefore do not impact Doc-F_1 and Tag-F_1 scores. The latter exception only shows a decline in Doc-F_1 scores at $\alpha > 0.4$, with Tag-F_1 scores increasing. This is because higher α values induced a deeper hierarchy which introduces more errors when higher level clusters inherit the documents of their descendants. Unlike Doc-F_1, Tag-F_1 is resistant to these errors since only the highest F_1 score is considered in the pairwise comparison between tags and clusters.

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.

5.2.1. Taxonomy induction

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[6].

In general, all tag hierarchy methods achieve encouraging results and our method outperforms the others on two of the four datasets. We notice that since Tax-F_1 measures the balance between precision and recall values, this suggests that our method is 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* $\rightarrow$ *dbo:Instrumentalist*$\}$ when the correct order is $\{$*dbo:Instrumentalist* $\rightarrow$ *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* $\rightarrow$ *dbo:University*$\}$, $\{$*dbo:EducationalInstitution* $\rightarrow$ *dbo:College*$\}\}$ while our induced taxonomy gives the following:

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

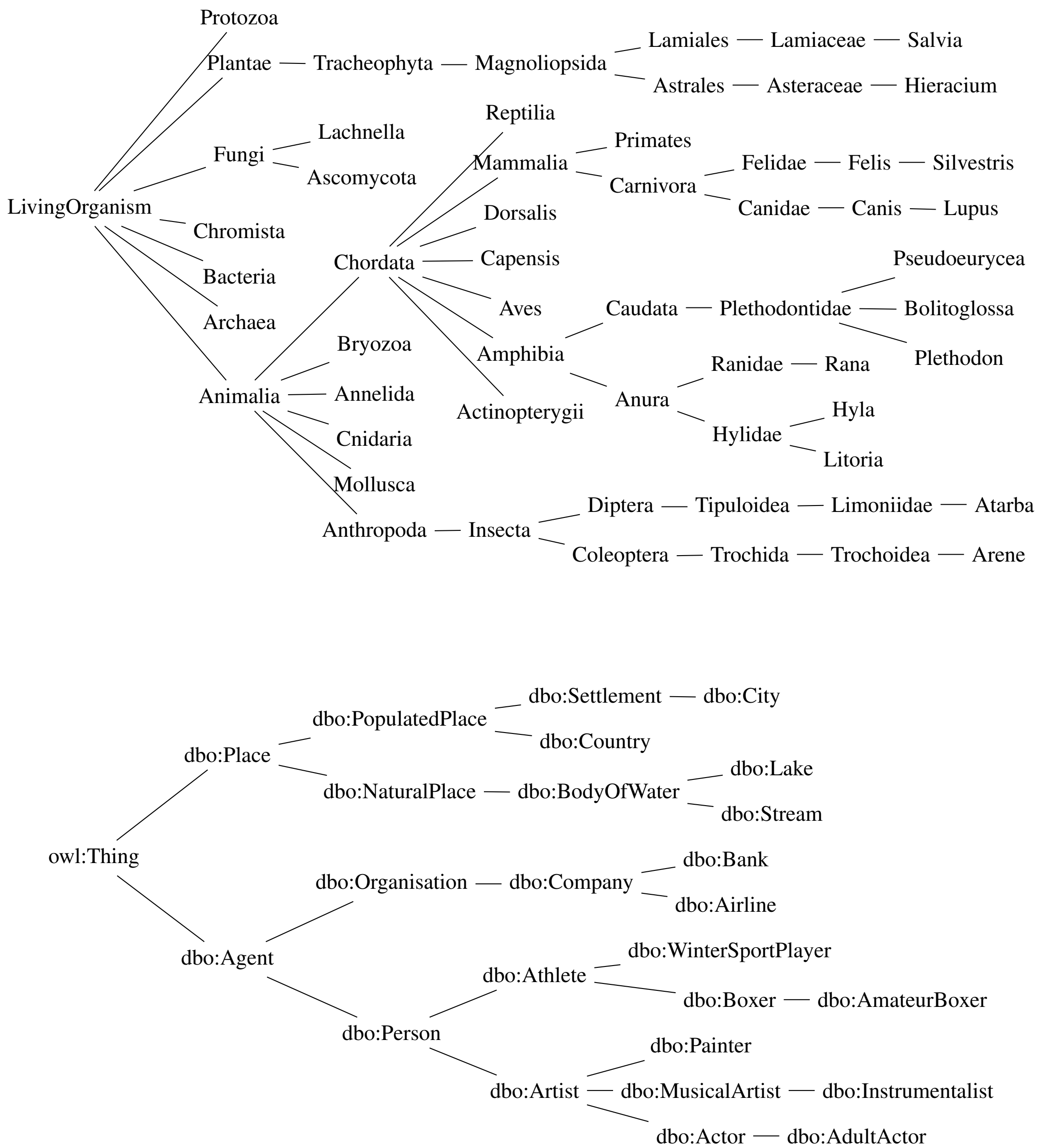

Fig. 2. Excerpts of the induced class taxonomies for the Life (top) and DBpedia (bottom) datasets. (Read left to right.)

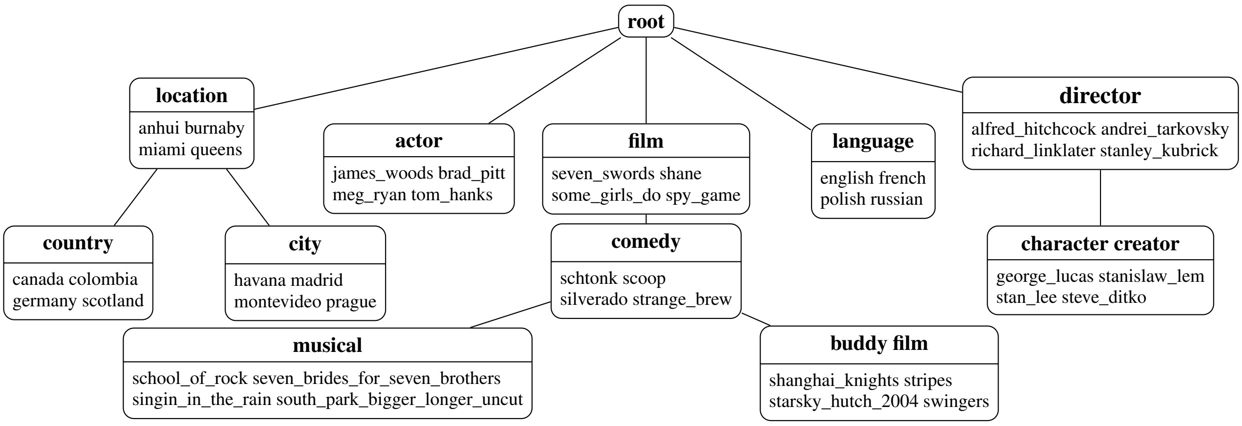

Fig. 3. Excerpt of the cluster hierarchy induced on the IIMB dataset. Node top indicates cluster's tag; bottom indicates cluster's constituent subjects.

Table 2

Hierarchical clustering results (mean ± standard deviation) on the Life, DBpedia, WordNet, and IIMB datasets.

Dataset	**Doc-F_1**	**Tag-F_1**
Life	0.9949 ± 0.0000	0.9499 ± 0.0000
DBpedia	0.9624 ± 0.0000	0.9572 ± 0.0000
WordNet	0.8977 ± 0.0000	0.8765 ± 0.0000
IIMB	0.8903 ± 0.0000	0.7843 ± 0.0000

$\{\{$*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.

5.2.2. Hierarchical clustering

As before, we apply our hierarchical clustering scheme on each of the four datasets at optimal α values as per the Tax-F_1 score. We repeat this process five times for each dataset and report the results in Table 2. We notice no variance in results despite our model's stochasticity in sorting tags. This is because the Doc-F_1 and Tag-F_1 metrics are insensitive to the ordering errors between parents and children discussed earlier. Furthermore, Doc-F_1 is higher than Tag-F_1 on all datasets. This suggests that our method is better at inducing clusters with strong inheritance properties and a high degree of consistency between cluster members and cluster annotation than it is at representing every class in the taxonomy with a cluster. Closer inspection of clustering errors shows that the majority of errors are the result of errors carried over from the taxonomy induction step. Specifically, the most common type of error is due to missing or incorrect ancestors in the paths of documents' clusters. Missing ancestors result in a false negative whereas incorrect ancestors result in a false positive, decreasing F_1 scores.

Figure 3 provides an excerpt of hierarchical clustering on the IIMB dataset. Recall that the induced class taxonomy showed a poor Tax-F_1 score on this dataset. Despite this, the hierarchical clustering obtained from this taxonomy scores highly on Doc-F_1 and Tag-F_1 and qualitative assessment confirm that it is well structured and coherent. This highlights problems with using gold standards, namely: there may be multiple valid ways of structuring a taxonomy; and there may be a disconnect between how the data ought to be structures and how it is structured. Both of these problems are manifest in the IIMB dataset.

5.3. Computational complexity analysis

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 taxonomy induction using our method is that of calculating the number of documents annotated by two tags, $D_{a,b}$, 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 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 four datasets.

Hierarchical clustering of documents involves a pairwise comparison between the documents and classes in the taxonomy. Thus, the time complexity of performing hierarchical clustering given the induced class taxonomy is $\mathcal{O}(|\mathcal{D}||\mathcal{V}|)$, allowing for fast execution even on large datasets. We note that the two metrics used for evaluating the clustering are relatively costly. Specifically, Doc-F_1 has a time complexity of $\mathcal{O}(|\mathcal{V}|)$ and Tag-F_1 has a time complexity of $\mathcal{O}(|\mathcal{V}^2|)$.

5.4. *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 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 3. 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 Tax-F_1 scores, the incremental increase in Tax-F_1 diminishes, and the scores plateau after 20,000 documents. However, relying on Tax-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 3

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

$\|\mathcal{D}\|$	$\|\mathcal{V}\|$	Triples	Optimal α	Time (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 demonstrated our method's ability to induce class hierarchies by applying it on four real-world datasets and evaluating it against their respective gold standard taxonomies. Finally, we showed how a class taxonomy may be used as the foundation for a simple hierarchical clustering scheme. This scheme was applied to the aforementioned datasets and evaluated on two metrics. Results demonstrate that our approach is capable of inducing high quality class taxonomies as well as hierarchical clusterings and can be reliable applied to large-scale knowledge graphs.

References

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

[2] J. Hoffart, F.M. Suchanek, K. Berberich and G. Weikum, YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia, *Artificial Intelligence* **194** (2013), 28–61.

[3] D. Vrandečić and M. Krötzsch, Wikidata: a free collaborative knowledge base (2014).

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge and J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, AcM, 2008, pp. 1247–1250.

[5] A. Singhal, Introducing the Knowledge Graph: things, not strings, 2012. https://www.blog.google/products/search/introducing-knowledge-graph-things-not/.

[6] J. Völker and M. Niepert, Statistical schema induction, in: *Extended Semantic Web Conference*, Springer, 2011, pp. 124–138.

[7] M. Nickel, V. Tresp and H.-P. Kriegel, Factorizing yago: scalable machine learning for linked data, in: *Proceedings of the 21st international conference on World Wide Web*, ACM, 2012, pp. 271–280.

[8] M. Nickel, V. Tresp and H.-P. Kriegel, A Three-Way Model for Collective Learning on Multi-Relational Data., 2011.

[9] M. Ankerst, M.M. Breunig, H.-P. Kriegel and J. Sander, OPTICS: ordering points to identify the clustering structure, in: *ACM Sigmod record*, Vol. 28, ACM, 1999, pp. 49–60.

[10] P. Ristoski, S. Faralli, S.P. Ponzetto and H. Paulheim, Large-scale taxonomy induction using entity and word embeddings, in: *Proceedings of the International Conference on Web Intelligence*, ACM, 2017, pp. 81–87.

[11] P. Heymann and H. Garcia-Molina, Collaborative creation of communal hierarchical taxonomies in social tagging systems, Technical Report, 2006.

[12] D. Benz, A. Hotho, S. Stützer and G. Stumme, Semantics made by you and me: Self-emerging ontologies can capture the diversity of shared knowledge (2010).

[13] P. Schmitz, Inducing ontology from flickr tags, in: *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland*, Vol. 50, 2006, p. 39.

[14] M. Sanderson and B. Croft, Deriving concept hierarchies from text, in: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, 1999, pp. 206–213.

[15] G. Solskinnsbakk and J.A. Gulla, A hybrid approach to constructing tag hierarchies, in: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, Springer, 2010, pp. 975–982.

[16] R. Agrawal and R. Srikant, Fast algorithms for mining association rules, in: *Proc. 20th int. conf. very large data bases, VLDB*, Vol. 1215, 1994, pp. 487–499.

[17] D.M. Blei, A.Y. Ng and M.I. Jordan, Latent dirichlet allocation, *Journal of machine Learning research* **3**(Jan) (2003), 993–1022.

[18] J. Tang, H.-f. Leung, Q. Luo, D. Chen and J. Gong, Towards ontology learning from folksonomies, in: *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[19] X. Li, H. Wang, G. Yin, T. Wang, C. Yang, Y. Yu and D. Tang, Inducing taxonomy from tags: An agglomerative hierarchical clustering framework, in: *International Conference on Advanced Data Mining and Applications*, Springer, 2012, pp. 64–77.

[20] S. Wang, D. Lo and L. Jiang, Inferring semantically related software terms and their taxonomy by leveraging collaborative tagging, in: *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, IEEE, 2012, pp. 604–607.

[21] H. Dong, W. Wang and F. Coenen, Learning Relations from Social Tagging Data, in: *Pacific Rim International Conference on Artificial Intelligence*, Springer, 2018, pp. 29–41.

[22] D.M. Roy, C. Kemp, V.K. Mansinghka and J.B. Tenenbaum, Learning annotated hierarchies from relational data, in: *Advances in neural information processing systems*, 2007, pp. 1185–1192.

[23] J.X. Chen and M.Z. Reformat, Learning categories from linked open data, in: *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Springer, 2014, pp. 396–405.

[24] G.J. Székely, M.L. Rizzo, N.K. Bakirov et al., Measuring and testing dependence by correlation of distances, *The annals of statistics* **35**(6) (2007), 2769–2794.

[25] S.K. Mohamed, Unsupervised Hierarchical Grouping of Knowledge Graph Entities, *arXiv preprint arXiv:1908.07281* (2019).

[26] C. Gu, G. Yin, T. Wang, C. Yang and H. Wang, A supervised approach for tag hierarchy construction in open source communities, in: *Proceedings of the 7th Asia-Pacific Symposium on Internetware*, ACM, 2015, pp. 148–152.

[27] W. Wang, P.M. Barnaghi and A. Bargiela, Probabilistic topic models for learning terminological ontologies, *IEEE Transactions on Knowledge and Data Engineering* **22**(7) (2009), 1028–1040.

[28] K. Liu, B. Fang and W. Zhang, Ontology emergence from folksonomies, in: *Proceedings of the 19th ACM international conference on Information and knowledge management*, ACM, 2010, pp. 1109–1118.

[29] F. Almoqhim, D.E. Millard and N. Shadbolt, Improving on popularity as a proxy for generality when building tag hierarchies from folksonomies, in: *International Conference on Social Informatics*, Springer, 2014, pp. 95–111.

[30] N. Chinchor, MUC-4 evaluation metrics, in: *Proceedings of the 4th conference on Message understanding*, Association for Computational Linguistics, 1992, pp. 22–29.

[31] O.T.N.D.B.N.K.P.M.B.T.D.R.E.D.W.N.E.v.Z.J.P.L.e. Roskov Y. Ower G., Species 2000 & ITIS Catalogue of Life, 2019 Annual Checklist. (2019).

[32] M. Döring, GBIF Type Specimen Names, 2017. https://doi.org/10.15468/sl9pyf.

[33] G.A. Miller, WordNet: a lexical database for English, *Communications of the ACM* **38**(11) (1995), 39–41.

[34] J. Euzenat, A. Ferrara, C. Meilicke, A. Nikolov, J. Pane, F. Scharffe, P. Shvaiko, H. Stuckenschmidt, O. Šváb-Zazamal, V. Svátek et al., Results of the ontology alignment evaluation initiative 2010, Technical Report, University of Trento, 2011.

[35] H. Paulheim and J. Fümkranz, Unsupervised generation of data mining features from linked open data, in: *Proceedings of the 2nd international conference on web intelligence, mining and semantics*, ACM, 2012, p. 31.