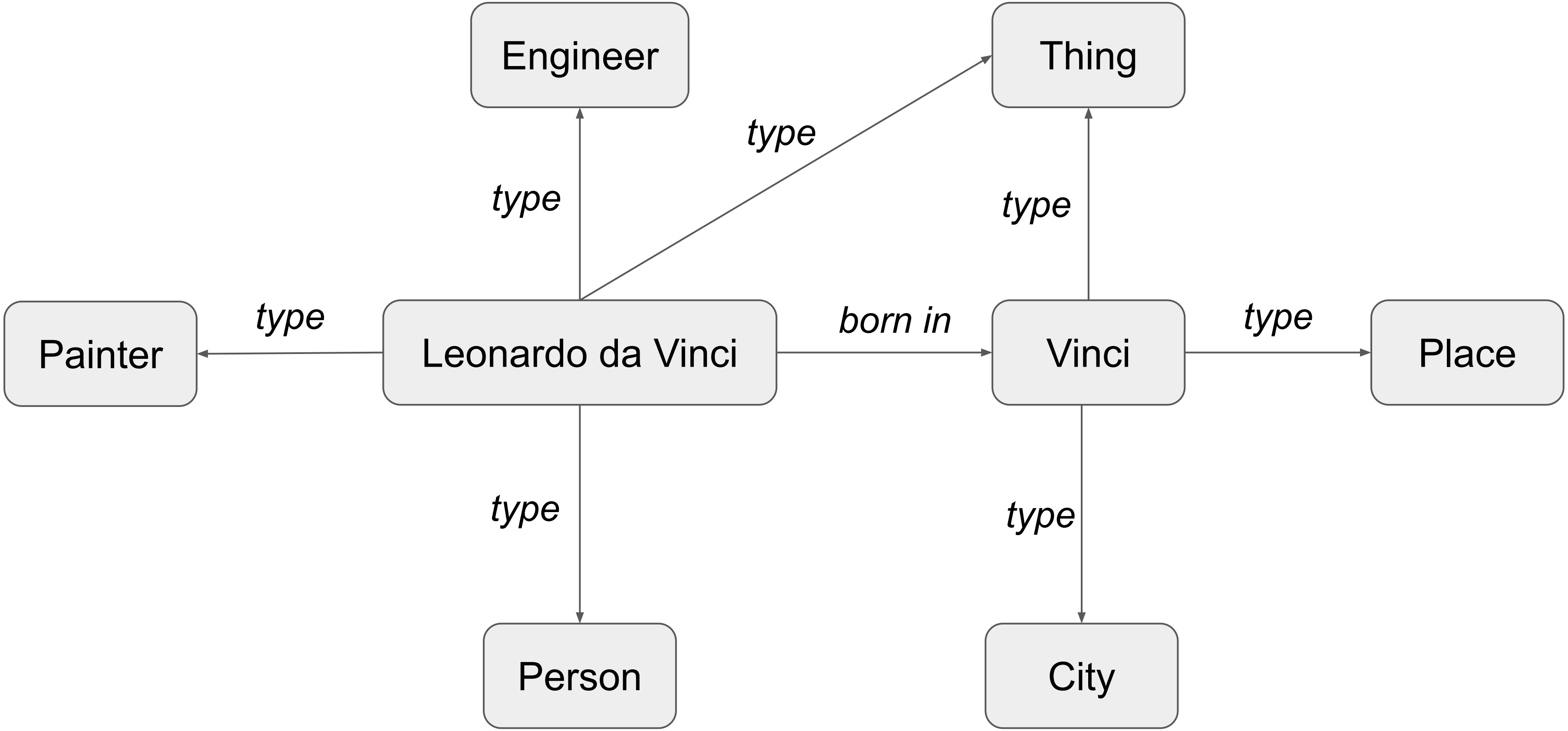
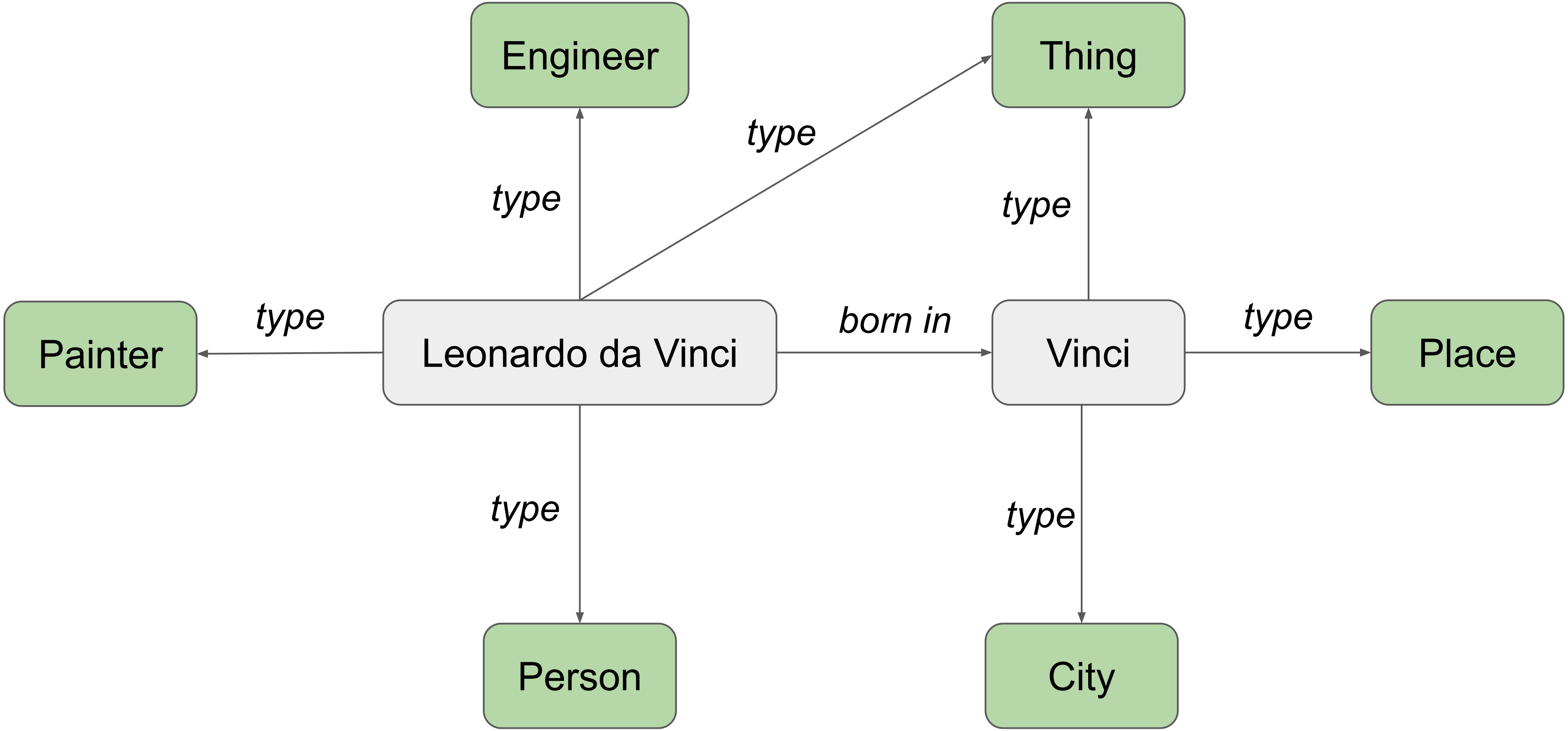
A Simple Method for Inducing Class Taxonomies in Knowledge Graphs

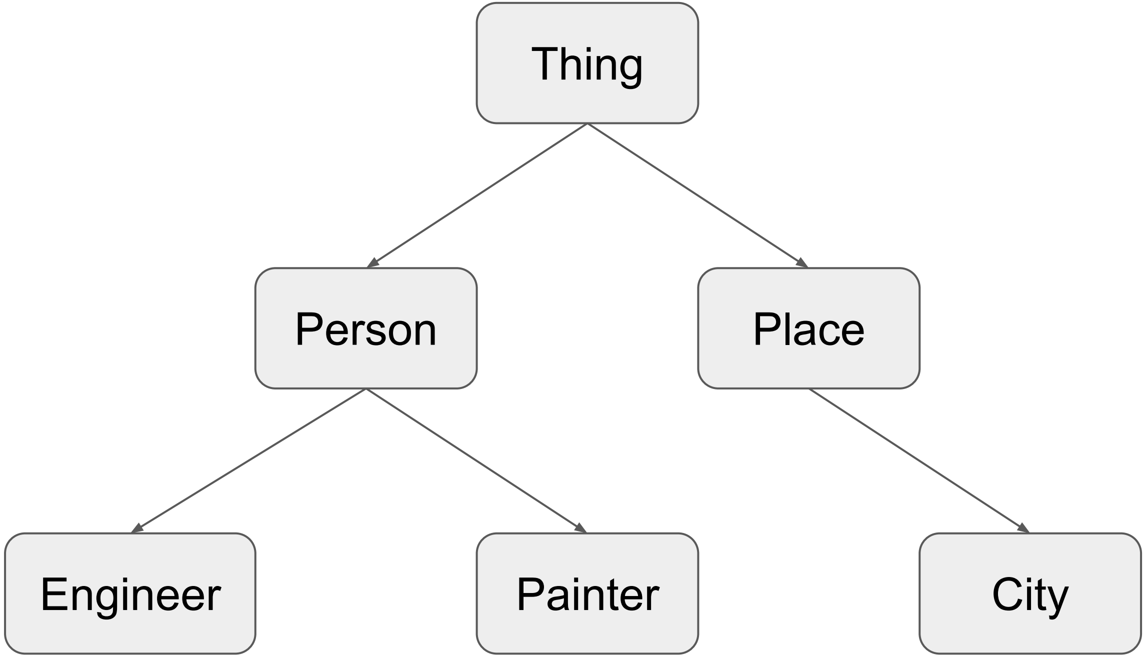
Marcin Pietrasik and Marek Reformat

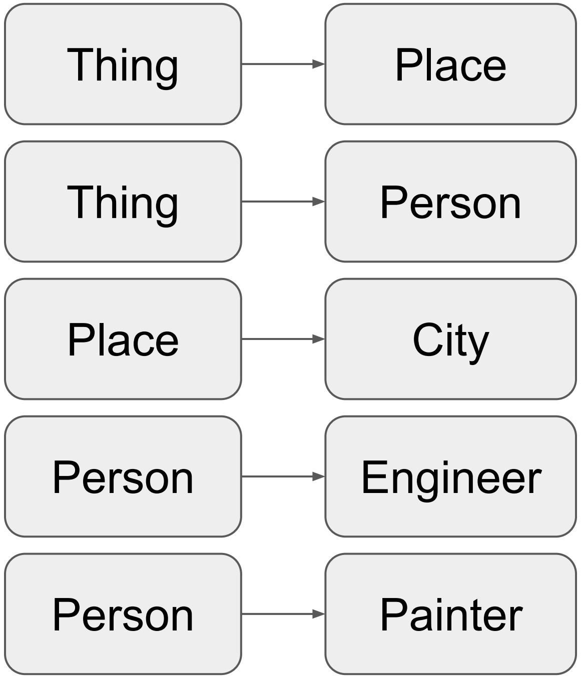
Motivation

- Many knowledge graphs capture their entities' type information by linking them to classes.
    - *eg.* entities in DBpedia are related to their classes through the *rdf:type* relation
- Hierarchical relations between these classes are often defined by a set of subsumption axioms.
    - eg. in DBpedia, *dbo:PopulatedPlace* subsumes *dbo:Country*
- Can we induce these subsumption axioms from just the knowledge graph?


Engineer
Thing
type
type
type
Painter
type
Leonardo da Vinci
born in
Vinci
type
Place
type
type
Person
City


Engineer
Thing
type
type
type
Painter
type
Leonardo da Vinci
born in
Vinci
type
Place
type
type
Person
City


Thing
Person
Place
Engineer
Painter
City



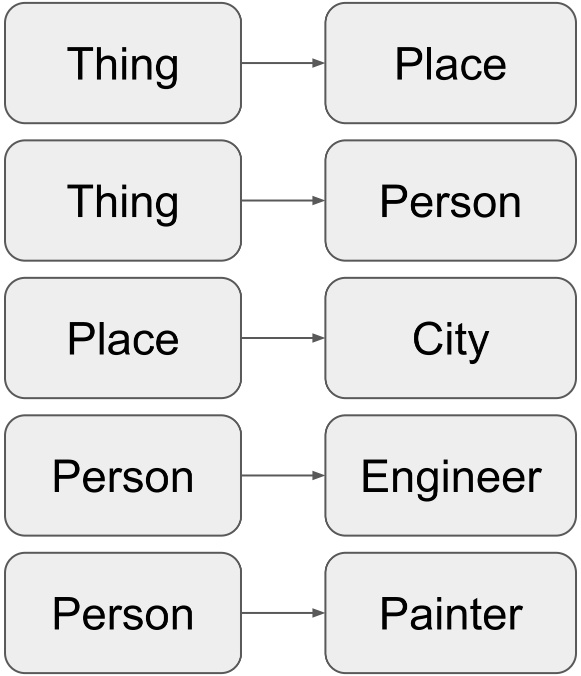
Thing → Place

Thing → Person

Place → City

Person → Engineer

Person → Painter



Thing → Place

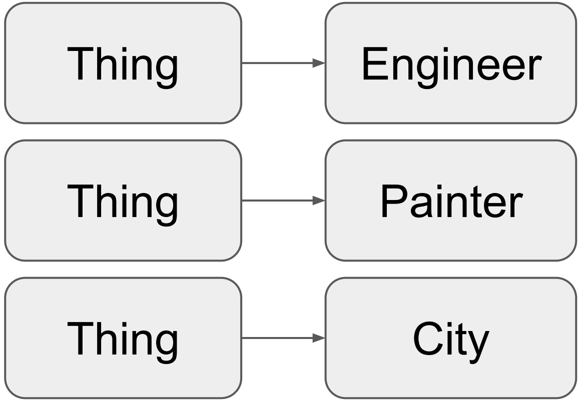
Thing → Person

Place → City

Person → Engineer

Person → Painter

By transitivity we can then deduce



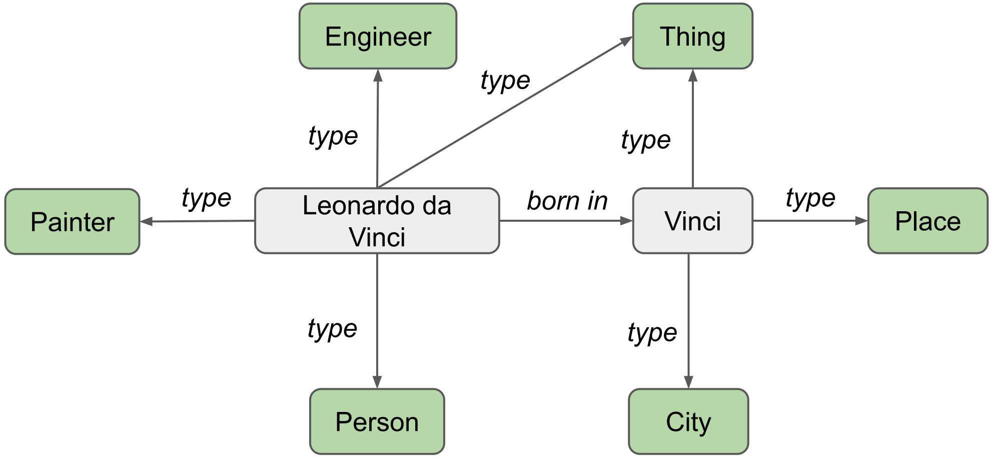
Thing → Engineer

Thing → Painter

Thing → City

Preliminaries

- We notice that in most knowledge graphs, entities are related to their classes on one property. We can therefore drop the relation without losing information, resulting in a tuple.
  - For instance, the triple <*Leonardo da Vinci, type, Painter*> is reduced to the tuple <*Leonardo da Vinci, Painter*>
- In this view, we can think of classes as tags which annotate an entity.
  - For instance, *Leonardo da Vinci* is annotated by *Thing, Person, Painter,* and *Engineer*.
- This view is common in Natural Language Processing where documents have words that describe them.



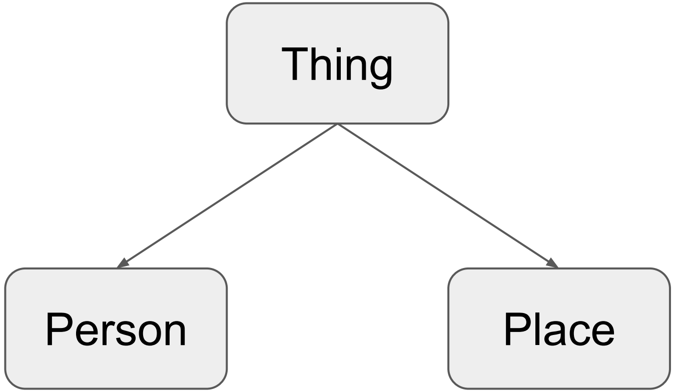
Entity/Document	**Classes/Tags/Words**
Leonardo da Vinci	Thing Person Painter Engineer
Vinci	Thing Place City

Preliminaries

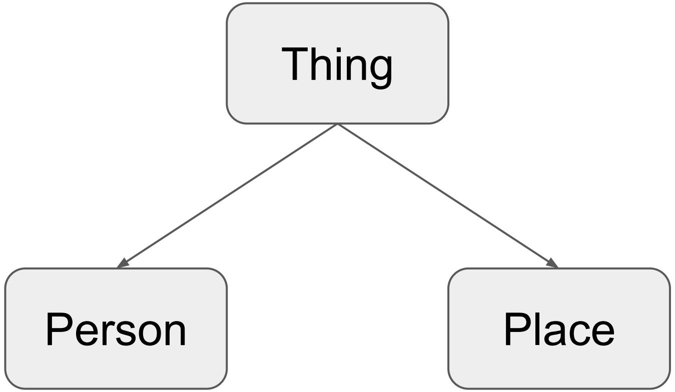
- We need to make two assumptions about the nature of knowledge graphs:
  - Classes higher up in the taxonomy annotate more entities than classes lower in the taxonomy.
    - For instance the class *animal* will annotate more entities than the class *dog*.
  - Subclasses are more likely to co-appear with their superclasses than with classes they are not subclasses of.
    - For instance the class *dog* will co-appear with the class *mammal* than with the class *reptile*.
- If our assumptions hold, we can induce a taxonomy by leveraging the frequencies and co-occurrences of class annotations.

Proposed Solution

1. Sort classes by generality.
   a. Generality is calculated by counting how often a class annotates an entity. Classes with higher generality annotate more entities.
2. Initialize root of taxonomy as the most general class.
3. In order of decreasing generality, add classes by making them the child of the class in the taxonomy it is most similar to.
   a. Similarity is calculated as the extent to which classes are likely to annotate the same entities.



Class	**Generality**
Engineer	0.1
Painter	0.2
City	0.4

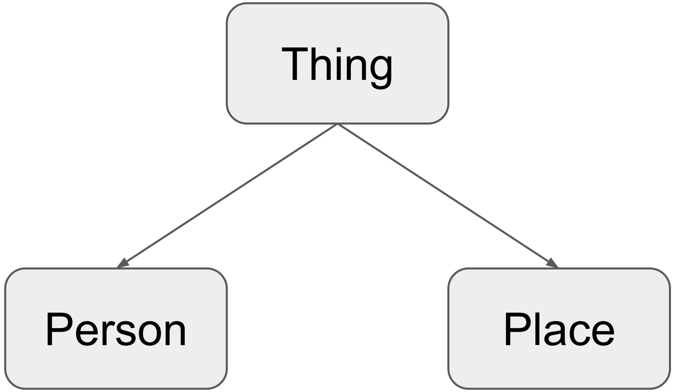


Thing

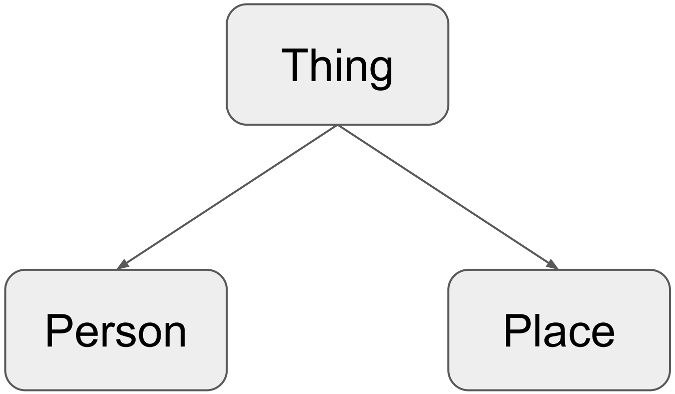
Person

Place

Class	Generality
Engineer	0.1
Painter	0.2
City	0.4



Class	Similarity w/ City
Thing	1
Person	0.5
Place	5

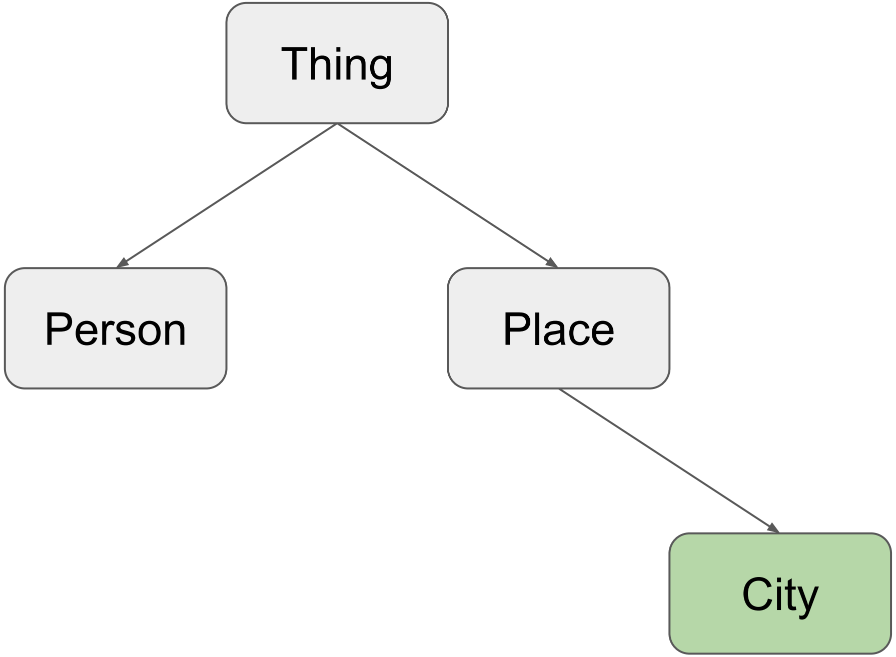


Thing

Person

Place

Class	Similarity w/ City
Thing	1
Person	0.5
Place	5


Thing
Person
Place
City

Evaluation Procedure

- We evaluate our method by inducing a class taxonomy on knowledge graphs and comparing them against each graph's gold standard taxonomy.
- The accuracy in inducing correct subsumption axioms is measured by the F_1 score.
- We do this on three real-world datasets.

Datasets

- The Life dataset consists of the taxonomic classification of 100,000 living organisms.
- The DBpedia dataset was obtained by querying DBpedia for 50,000 entities and their classes in the DBpedia ontology.
- The WordNet dataset includes types from DBpedia which also appear in the WordNet lexical database. This dataset consists of 50,000 entities.

Results

	Life	**DBpedia**	**WordNet**
Heymann and Garcia-Molina	-	0.7982 ± 0.0149	0.5918 ± 0.0114
Schmitz	0.8423 ± 0.0000	0.8013 ± 0.0000	0.7943 ± 0.0000
Out Method	0.8625 ± 0.0040	0.8824 ± 0.0052	0.7144 ± 0.0069

Scalability

- Worst case time complexity is $O(DV^2)$ where D is the number of entities and V is the number of classes.
- Average case time complexity if $O(DA^2)$ where A is the average number of classes that annotate an entity.
- Runtime of unoptimized Python implementation running on single core Intel i7 processor on 100,000 entity DBpedia dataset is ~1.6 seconds.

Conclusion

- We proposed a new method of inducing class taxonomies in knowledge graphs.
- Results obtained using our method outperform other methods used in Natural Language Processing.
- Code for our paper can be found at: https://github.com/mpietrasik/smict