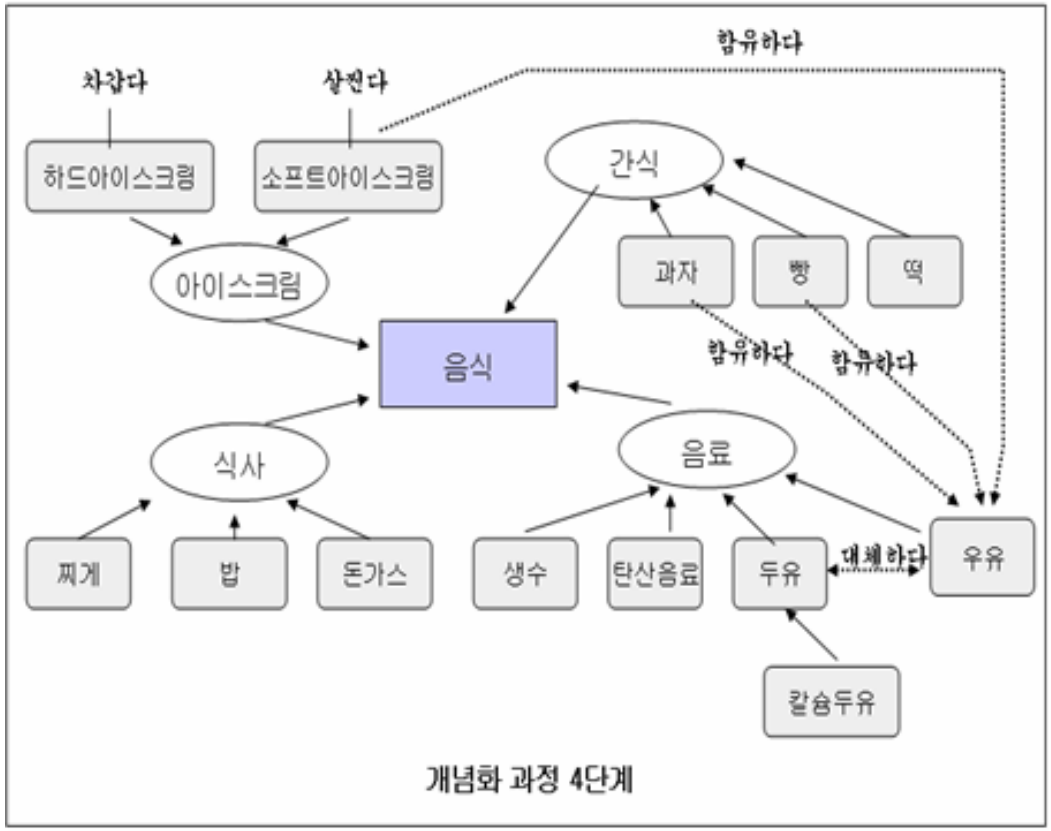
Large-scale Taxonomy Induction Using Entity and Word Embeddings

Ristoski, Petar, et al. "Large-scale taxonomy induction using entity and word embeddings." *Proceedings of the International Conference on Web Intelligence*. ACM, 2017.

Introduction

❖ Ontology?

- A knowledge representation framework consisting of classes, relational properties, and instances

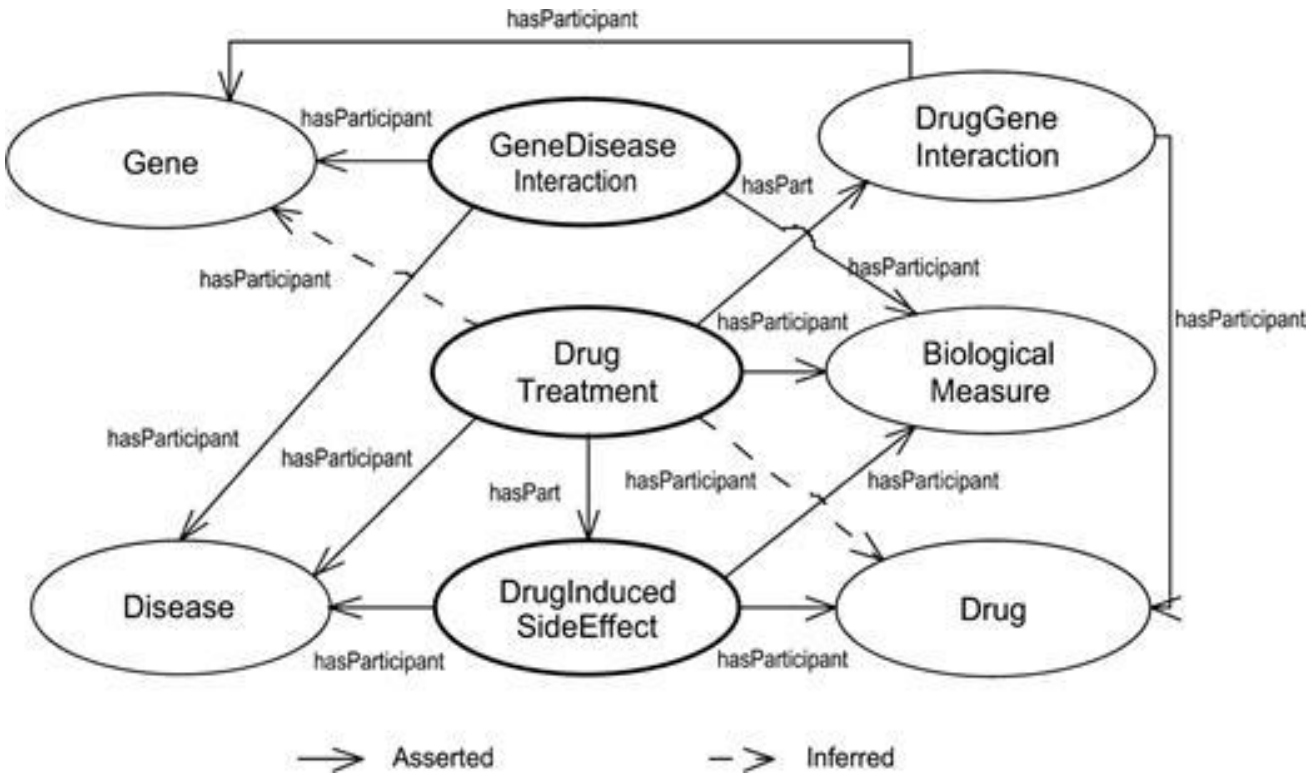


Hierarchical concepts and properties

Introduction

❖ Ontology?

- A knowledge representation framework consisting of classes, relational properties, and instances



Knowledge graph (instance)

Introduction

- A basic building block of an ontology is a **class**, where the classes are organized with **"is-a" relations** (or class subsumption axioms).

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg${peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+ \sqsubseteq$ ancestor
functionalProperty	$\top \sqsubseteq \leqslant 1P$	$\top \sqsubseteq \leqslant$1hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leqslant 1P^-$	$\top \sqsubseteq \leqslant$1hasSSN$^-$

[그림8. OWL의 Axiom]

Introduction

- A basic building block of an ontology is a **class**, where the classes are organized with **"is-a" relations** (or class subsumption axioms).

- **Manually** curating a class hierarchy for a given knowledge graph is **time consuming** and requires a high cost.

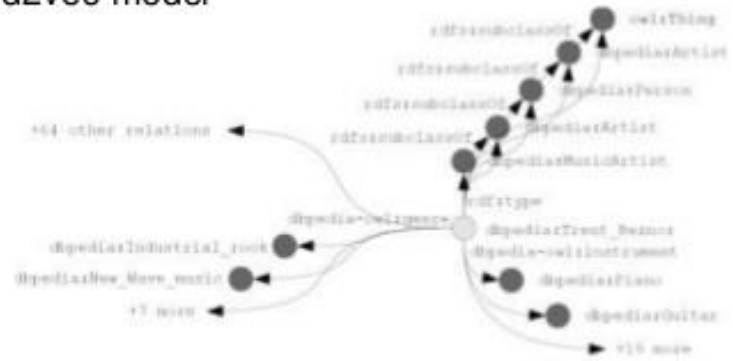
- Proposing the ***TIEmb*** **approach** for **automatic unsupervised class subsumption axiom extraction** from knowledge bases using **entity and text embeddings (RDF2vec)**.
  - ✓ RDF : a basic standard format (model) for ontological expression

Introduction

❖ RDF2vec

Graph Walks RDF2vec

- For each entity in the graph:
  - Extract a subgraph with depth *d*
  - Extract walks on the subgraph
  - Build word2vec model



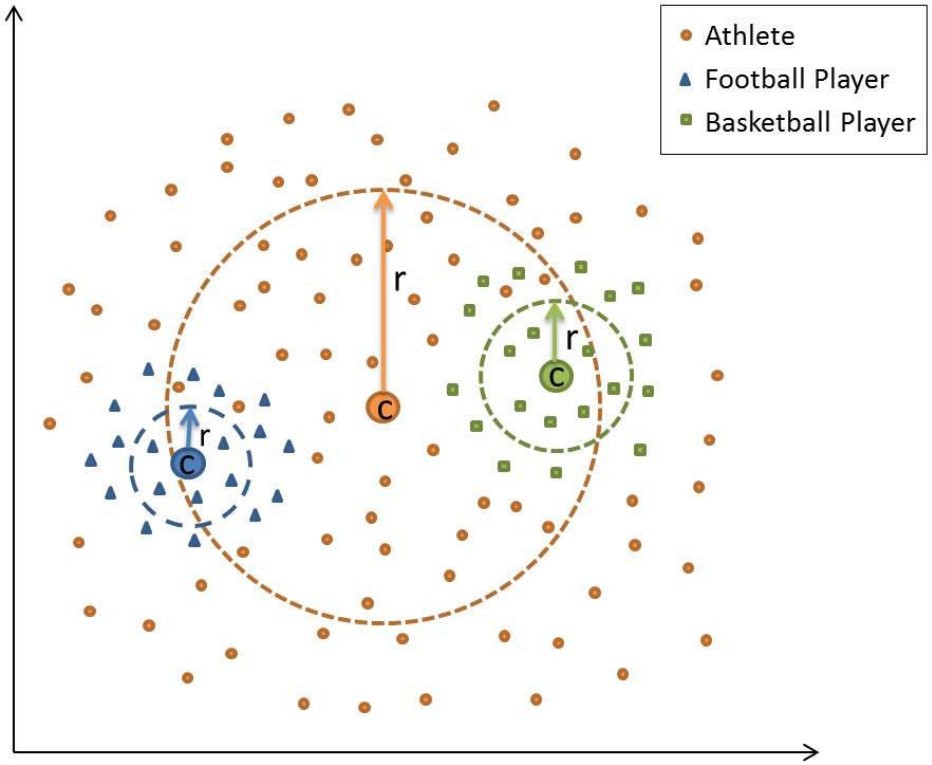
dbr:Trent_Reznor -> dbo:associatedBand -> dbr:Exotic_Birds -> dbo:bandMember -> dbr:Chris_Vrenna

dbr:Trent_Reznor -> dbo:genre - > dbr:Dark_ambient -> dbo:instrument -> dbr:Field_recording

Introduction

❖ The underlying assumptions

1. **the majority of all instances of the same class are positioned close to each other** in the embedded space.
2. each class in the knowledge base can be represented as a cluster of the instances in the embedded space, defined with a centroid and an average radius.
3. clusters that completely or partially subsume each other, indicate class subsumption axiom.



Approach

Algorithm 1: Algorithm for class subsumption axioms extraction from a knowledge base

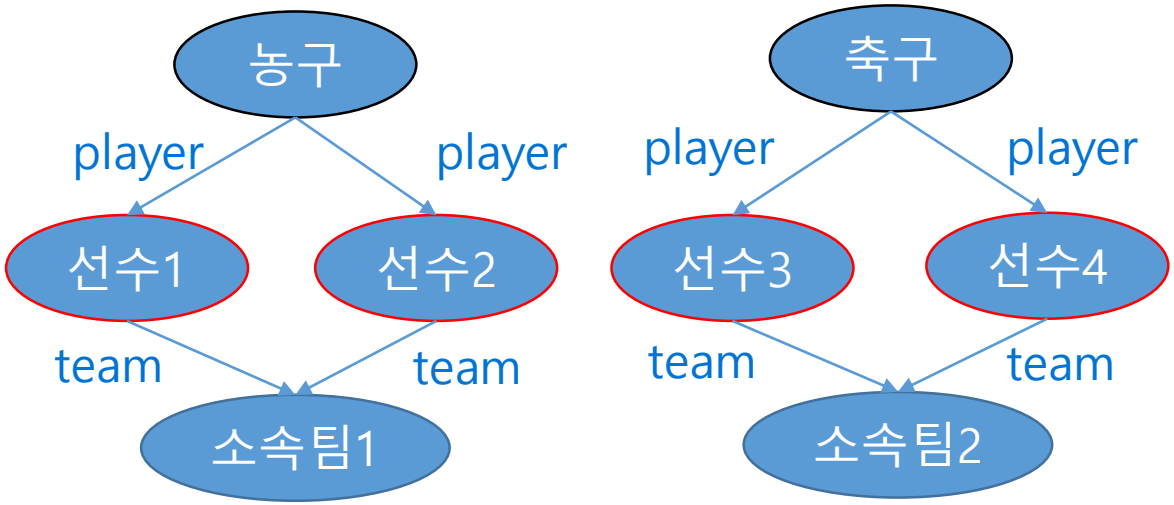
Data: KB: Knowledge base, V_{KB}: Knowledge base embeddings for the knowledge base KB
Result: A: Set of class subsumption axioms

$A := \emptyset$
Calculate the centroid and radius for each class in the knowledge base
$C_{KB} = \{c \mid \exists i \; typeOf \; c \wedge i \in KB\}$
foreach *class* $c \in C_{KB}$ **do**
  $I_c = \{i \mid i \; typeOf \; c \wedge i \in KB \wedge \exists v_i \in V_{KB}\}$
  $c.centroid = \frac{1}{|I_c|} \sum_{i \in I_c} v_i$
  $c.radius = \sqrt{\frac{1}{|I_c|} \sum_{i \in I_c} (v_i - c.centroid)^2}$
end
Extract class subsumption axioms
foreach *class* $c_1 \in C$ **do**
  $A_{c_1} := \emptyset$
  foreach *class* $c_2 \in C$ **do**
    if $c_1 == c_2$ **then**
      continue
    end
    $distance_{c_1 c_2} = distance(c_1.centroid, \; c_2.centroid)$
    if $distance_{c_1 c_2} \leq c_2.radius$ *and* $c_1.radius < c_2.radius$ **then**
      $axiom = c_1 \sqsubseteq c_2$
      add $[axiom, distance]$ to A_{c_1}
    end
  end
  sort A_{c_1} in ascending order
  add $A_{c_1}.first()$ to A
end
Compute transitive closure
$change = true$
while $change == true$ **do**
  foreach *axiom* $a \in A$ **do**
    $change = false$
    $subClass = axiom.getSubClass()$
    $superClass = axiom.getSuperClass()$
    $superClassesAxioms = \{c, axiom \mid \exists axiom \in A \wedge axiom = superClass \sqsubseteq c\}$
    foreach *classAxiom* $s \in superClassesAxioms$ **do**
      if $s.class == subClass$ **then**
        remove $s.axiom$ from A
        continue
      end
      $axiom = subClass \sqsubseteq s$
      if $axiom \notin A$ **then**
        add $axiom$ to A
        $change = true$
      end
    end
  end
end
return A

Assumption

- entities which belong to the same class are positioned close to each other in the vector space

- instances of a more specific class should be positioned closer to each other on average than instances of a broader class.



Approach

Algorithm 1: Algorithm for class subsumption axioms extraction from a knowledge base

Data: KB: Knowledge base, V_{KB}: Knowledge base embeddings for the knowledge base KB
Result: A: Set of class subsumption axioms

$A := \emptyset$
Calculate the centroid and radius for each class in the knowledge base
$C_{KB} = \{c \mid \exists i \; typeOf \; c \wedge i \in KB\}$
foreach *class* $c \in C_{KB}$ **do**
  $I_c = \{i \mid i \; typeOf \; c \wedge i \in KB \wedge \exists v_i \in V_{KB}\}$
  $c.centroid = \frac{1}{|I_c|} \sum_{i \in I_c} v_i$
  $c.radius = \sqrt{\frac{1}{|I_c|} \sum_{i \in I_c} (v_i - c.centroid)^2}$
end
Extract class subsumption axioms
foreach *class* $c_1 \in C$ **do**
  $A_{c_1} := \emptyset$
  foreach *class* $c_2 \in C$ **do**
    if $c_1 == c_2$ **then**
      continue
    end
    $distance_{c_1 c_2} = distance(c_1.centroid, \; c_2.centroid)$
    if $distance_{c_1 c_2} \leq c_2.radius \; and \; c_1.radius < c_2.radius$ **then**
      $axiom = c_1 \sqsubseteq c_2$
      add $[axiom, distance]$ to A_{c_1}
    end
  end
  sort A_{c_1} in ascending order
  add $A_{c_1}.first()$ to A
end
Compute transitive closure
$change = true$
while $change == true$ **do**
  foreach *axiom* $a \in A$ **do**
    $change = false$
    $subClass = axiom.getSubClass()$
    $superClass = axiom.getSuperClass()$
    $superClassesAxioms = \{c, axiom \mid \exists axiom \in A \wedge axiom = superClass \sqsubseteq c\}$
    foreach *classAxiom* $s \in superClassesAxioms$ **do**
      if $s.class == subClass$ **then**
        remove $s.axiom$ from A
        continue
      end
      $axiom = subClass \sqsubseteq s$
      if $axiom \notin A$ **then**
        add $axiom$ to A
        $change = true$
      end
    end
  end
end
return A

Two inputs

1) a knowledge base as its input, which contains a set of instances, where each instance has one or more class types

2) the knowledge base embeddings, where each instance is represented as n-dimensional feature vector

Output

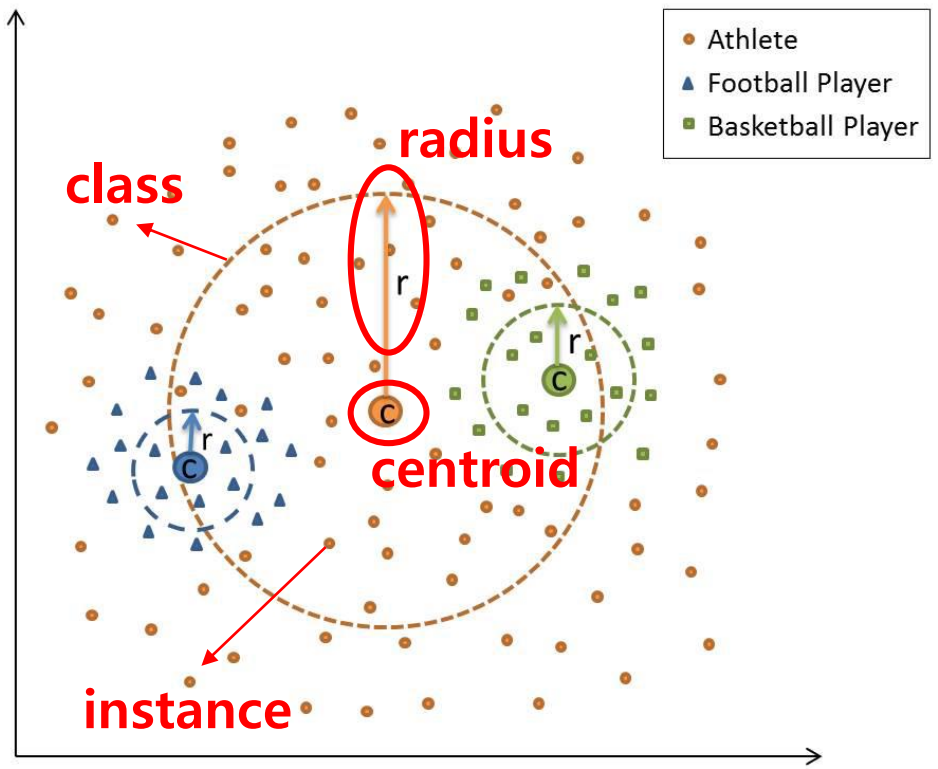
- a set of class subsumption axioms

Approach

Algorithm 1: Algorithm for class subsumption axioms extraction from a knowledge base

Data: KB: Knowledge base, V_{KB}: Knowledge base embeddings for the knowledge base KB
Result: A: Set of class subsumption axioms

$A := \emptyset$
\# Calculate the centroid and radius for each class in the knowledge base
$C_{KB} = \{c \mid \exists i \; typeOf \; c \wedge i \in KB\}$
foreach *class* $c \in C_{KB}$ **do**
  $I_c = \{i \mid i \; typeOf \; c \wedge i \in KB \wedge \exists v_i \in V_{KB}\}$
  $c.centroid = \frac{1}{|I_c|} \sum_{i \in I_c} v_i$
  $c.radius = \sqrt{\frac{1}{|I_c|} \sum_{i \in I_c} (v_i - c.centroid)^2}$
end
\# Extract class subsumption axioms
foreach *class* $c_1 \in C$ **do**
  $A_{c_1} := \emptyset$
  foreach *class* $c_2 \in C$ **do**
    if $c_1 == c_2$ **then**
      continue
    end
    $distance_{c_1 c_2} = distance(c_1.centroid, \; c_2.centroid)$
    if $distance_{c_1 c_2} \leq c_2.radius \; and \; c_1.radius < c_2.radius$ **then**
      $axiom = c_1 \sqsubseteq c_2$
      add $[axiom, distance]$ to A_{c_1}
    end
  end
  sort A_{c_1} in ascending order
  add $A_{c_1}.first()$ to A
end
\# Compute transitive closure
$change = true$
while $change == true$ **do**
  foreach *axiom* $a \in A$ **do**
    $change = false$
    $subClass = axiom.getSubClass()$
    $superClass = axiom.getSuperClass()$
    $superClassesAxioms = \{c, axiom \mid \exists axiom \in A \wedge axiom = superClass \sqsubseteq c\}$
    foreach *classAxiom* $s \in superClassesAxioms$ **do**
      if $s.class == subClass$ **then**
        remove $s.axiom$ from A
        continue
      end
      $axiom = subClass \sqsubseteq s$
      if $axiom \notin A$ **then**
        add $axiom$ to A
        $change = true$
      end
    end
  end
end
return A



RDF2vec space

Approach

Algorithm 1: Algorithm for class subsumption axioms extraction from a knowledge base

Data: KB: Knowledge base, V_{KB}: Knowledge base embeddings for the knowledge base KB
Result: A: Set of class subsumption axioms

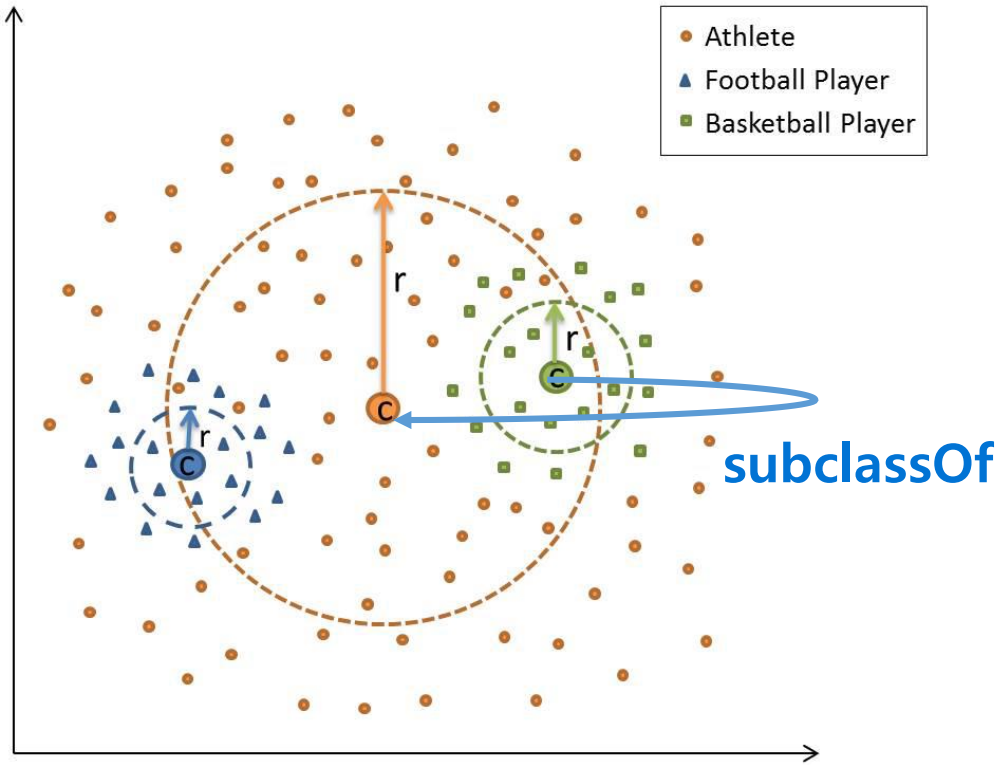
$A := \emptyset$
Calculate the centroid and radius for each class in the knowledge base
$C_{KB} = \{c \mid \exists i \; typeOf \; c \wedge i \in KB\}$
foreach *class* $c \in C_{KB}$ **do**
  $I_c = \{i \mid i \; typeOf \; c \wedge i \in KB \wedge \exists v_i \in V_{KB}\}$
  $c.centroid = \frac{1}{|I_c|} \sum_{i \in I_c} v_i$
  $c.radius = \sqrt{\frac{1}{|I_c|} \sum_{i \in I_c} (v_i - c.centroid)^2}$
end
Extract class subsumption axioms
foreach *class* $c_1 \in C$ **do**
  $A_{c_1} := \emptyset$
  foreach *class* $c_2 \in C$ **do**
    if $c_1 == c_2$ **then**
      continue
    end
    $distance_{c_1c_2} = distance(c_1.centroid, \; c_2.centroid)$
    if $distance_{c_1c_2} \leq c_2.radius$ *and* $c_1.radius < c_2.radius$ **then**
      $axiom = c_1 \sqsubseteq c_2$
      add $[axiom, distance]$ to A_{c_1}
    end
  end
  sort A_{c_1} in ascending order
  add $A_{c_1}.first()$ to A
end
Compute transitive closure
$change = true$
while $change == true$ **do**
  foreach *axiom* $a \in A$ **do**
    $change = false$
    $subClass = axiom.getSubClass()$
    $superClass = axiom.getSuperClass()$
    $superClassesAxioms = \{c, axiom \mid \exists axiom \in A \wedge axiom = superClass \sqsubseteq c\}$
    foreach *classAxiom* $s \in superClassesAxioms$ **do**
      if $s.class == subClass$ **then**
        remove $s.axiom$ from A
        continue
      end
      $axiom = subClass \sqsubseteq s$
      if $axiom \notin A$ **then**
        add $axiom$ to A
        $change = true$
      end
    end
  end
end
return A

- Class subsumption axiom (condition)

(distance$_{c1c2}$ < c2:radius) && (c1:radius < c2:radius)

*distance$_{c1c2}$ = distance *(c1:centroid; c2:centroid)*

select the axiom with the smallest distance



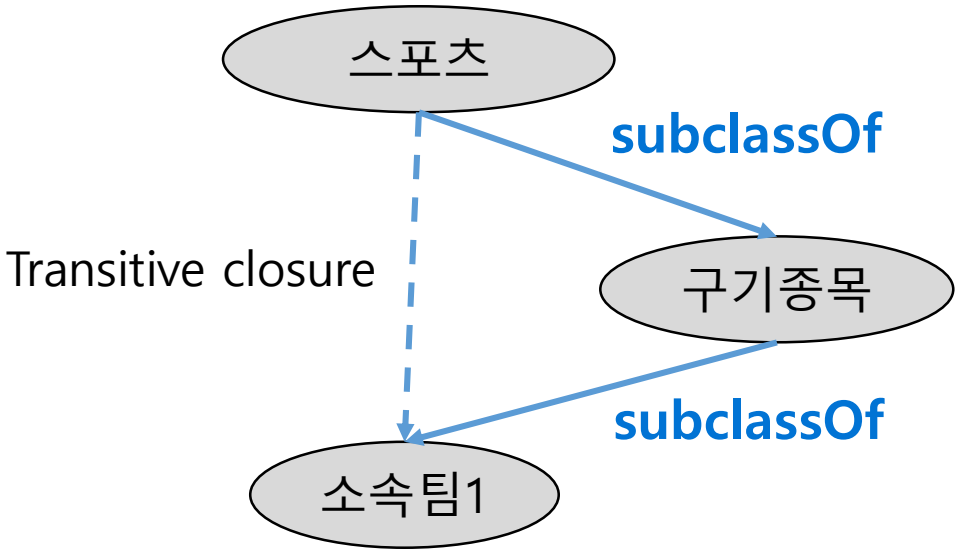
RDF2vec space

Approach

Algorithm 1: Algorithm for class subsumption axioms extraction from a knowledge base

Data: KB: Knowledge base, V_{KB}: Knowledge base embeddings for the knowledge base KB
Result: A: Set of class subsumption axioms

$A := \emptyset$
Calculate the centroid and radius for each class in the knowledge base
$C_{KB} = \{c \mid \exists i \; typeOf \; c \wedge i \in KB\}$
foreach *class* $c \in C_{KB}$ **do**
  $I_c = \{i \mid i \; typeOf \; c \wedge i \in KB \wedge \exists v_i \in V_{KB}\}$
  $c.centroid = \frac{1}{|I_c|} \sum_{i \in I_c} v_i$
  $c.radius = \sqrt{\frac{1}{|I_c|} \sum_{i \in I_c} (v_i - c.centroid)^2}$
end
Extract class subsumption axioms
foreach *class* $c_1 \in C$ **do**
  $A_{c_1} := \emptyset$
  foreach *class* $c_2 \in C$ **do**
    if $c_1 == c_2$ **then**
      continue
    end
    $distance_{c_1 c_2} = distance(c_1.centroid, \; c_2.centroid)$
    if $distance_{c_1 c_2} \leq c_2.radius \; and \; c_1.radius < c_2.radius$ **then**
      $axiom = c_1 \sqsubseteq c_2$
      add $[axiom, distance]$ to A_{c_1}
    end
  end
  sort A_{c_1} in ascending order
  add $A_{c_1}.first()$ to A
end
Compute transitive closure
$change = true$
while $change == true$ **do**
  foreach *axiom* $a \in A$ **do**
    $change = false$
    $subClass = axiom.getSubClass()$
    $superClass = axiom.getSuperClass()$
    $superClassesAxioms = \{c, axiom \mid \exists axiom \in A \wedge axiom = superClass \sqsubseteq c\}$
    foreach *classAxiom* $s \in superClassesAxioms$ **do**
      if $s.class == subClass$ **then**
        remove $s.axiom$ from A
        continue
      end
      $axiom = subClass \sqsubseteq s$
      if $axiom \notin A$ **then**
        add $axiom$ to A
        $change = true$
      end
    end
  end
end
return A



Experiment 1

- Dataset : 2016-04 DBpedia dataset and the corresponding ontology (classes that have at least one instance, resulting in 415 classes, 632 atomic class subsumption axioms)
- RDF2vec: 250 random walks, Skip-Gram (window 5, d 200)
- baseline : incoming, outgoing, in/out relation feature generation approaches

Table 1: Results for class subsumption axioms extraction using DBpedia ontology as a gold standard

Method	Precision	Recall	F-score
rel out	0.056	0.076	0.064
rel in	0.097	0.209	0.132
rel in & out	0.104	0.219	0.141
TIEmb	**0.594**	**0.465**	**0.521**

Experiment 1

- Error analysis (false positive)
  - extracts subsumption axioms for classes on the same level in the hierarchy, or siblings classes
    - ✓ e.g. *dbo:Bird* $\subseteq$ *dbo:Mammal*.
    - ✓ Reason : the centroids of the classes are positioned very close to each other in the embeddings space
  - Some would not necessarily be incorrect, but those axioms simply do not exist in the DBpedia ontology
    - ✓ e.g. *dbo:Senator* $\subseteq$ *dbo:OfficeHolder*

Experiment 2

- Dataset : WebIsA database
  - generated from the Common-Crawl by Hearst patterns
  - High coverage, low precision
  - Domain class: *Person*, *Place*

Dan Jurafsky

Hearst's Patterns for extracting IS-A relations

Hearst pattern	Example occurrences
X and other Y	...temples, treasuries, and other important civic buildings.
X or other Y	Bruises, wounds, broken bones or other injuries...
Y such as X	The bow lute, such as the Bambara ndang...
Such Y as X	...such authors as Herrick, Goldsmith, and Shakespeare.
Y including X	...common-law countries, including Canada and England...
Y , especially X	European countries, especially France, England, and Spain...

- Use DBpedia as filters
  - select all the subclasses of dbo:Person (184 in total) and dbo:Place (176 in total) in Dbpedia
  - select all the instances of these classes in WebIsADb
  - expand the set of classes by addingall siblings of the corresponding class
    - ✓ Ex. 'SoccerPlayer' (c) -> "Cristiano Ronaldo" (i) ->**'Star', 'Footballer'** … (c)
  - Use Dbpedia RDF2vec embeddings

Experiment 2

- Baseline : association rules (Apriori algorithm)
  - Support, confidence > 50

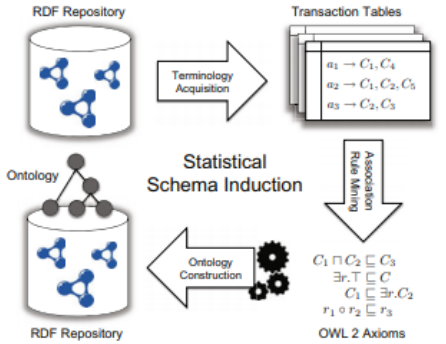


Fig. 1. Workflow of the Statistical Schema Induction framework

- Measurement : Dbpedia coverage, Extra Coverage, Precision (random 100)

Table 2: Results for the class subsumption axiom induction in the *Person* domain

Method	DBpedia Coverage	Extra Coverage(%)	Precision (random 100)
Association Rules	**0.44**	**31301.63**	0.25
RDF2Vec TIEmb	0.31	3594.44	**0.42**
GloVe TIEmb	0.29	1520.00	0.29

Table 3: Results for the class subsumption axiom induction in the *Place* domain

Method	DBpedia Coverage	Extra Coverage(%)	Precision (random 100)
Association Rules	**0.25**	**11029.71**	0.45
RDF2Vec TIEmb	0.21	3808.57	**0.54**
GloVe TIEmb	0.17	301.14	0.48

Experiment 2

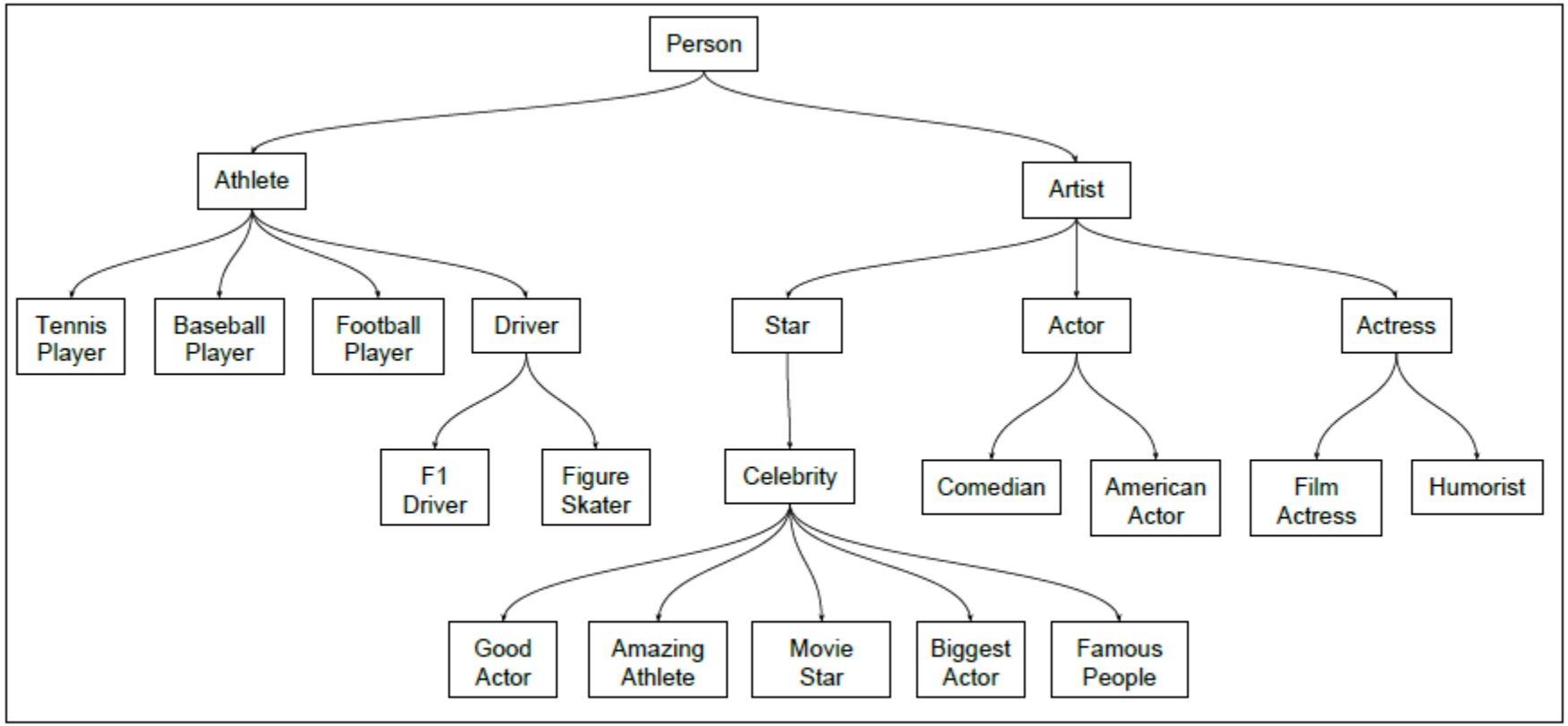


Figure 2: Excerpt of the *Person* top level hierarchy

Experiment 2

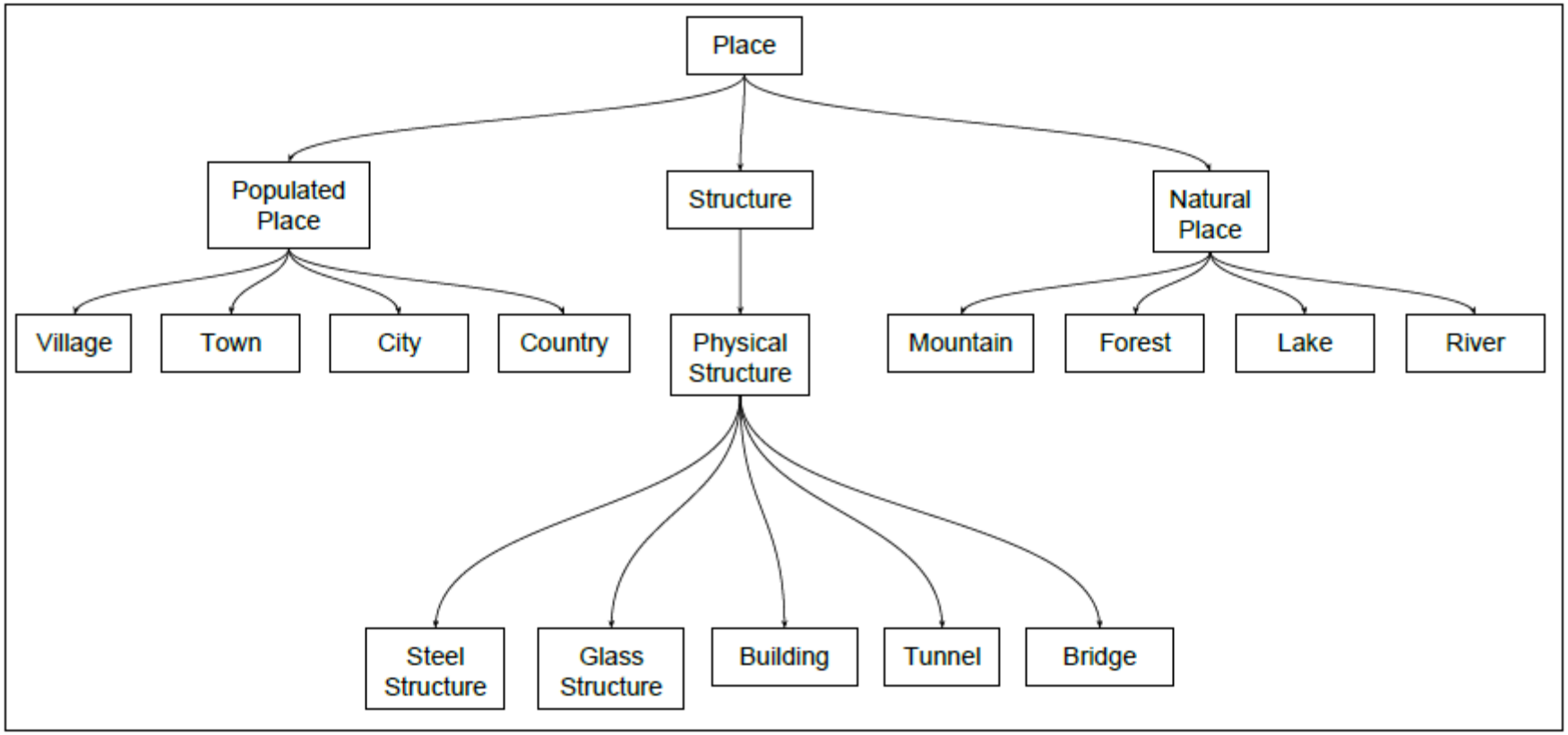


Figure 3: Excerpt of the *Place* top level hierarchy

Conclusion

- the approach cannot only be used for taxonomy induction, but also for the problem of type prediction

- interesting to see howembeddings coming from text and graphs can be combined reasonably.

- want to investigate how higher-level semantic knowledge, such as class restrictions or complementarity, can be mined using an embedding-based approach