Detect Missing Attributes for Entities in Knowledge Bases via Hierarchical Clustering

Bingfeng Luo, Huanquan Lu, Yigang Diao,
Yansong Feng and Dongyan Zhao
ICST, Peking University

Motivations

- **Entities often have missing attributes**
  - **Human-made KBs: human negligence**
  - **Auto-constructed KBs: incompleteness of source data, imperfectness of algorithm**
- **Detecting missing attributes is useful**
  - **Present possible missing attributes to open KB (like Wikipedia) editors**
  - **Rescore candidate triples proposed by relation extraction tools**

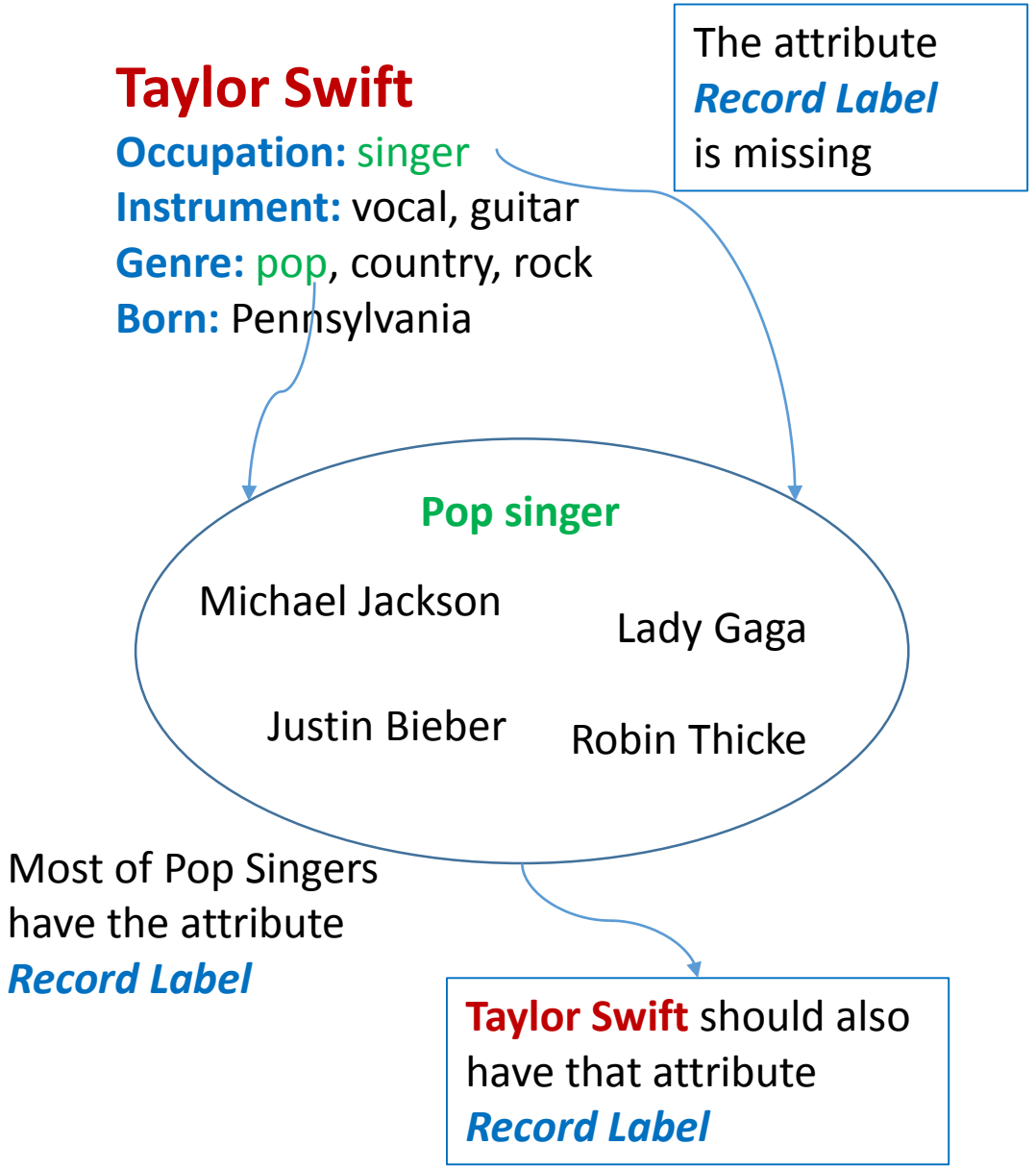
Taylor Swift (Wikipedia)

Occupation: singer
Instrument: vocal, guitar
Genre: pop, country, rock
Born: Pennsylvania

The attribute **Record Label** is missing!
(She works for Big Machine)

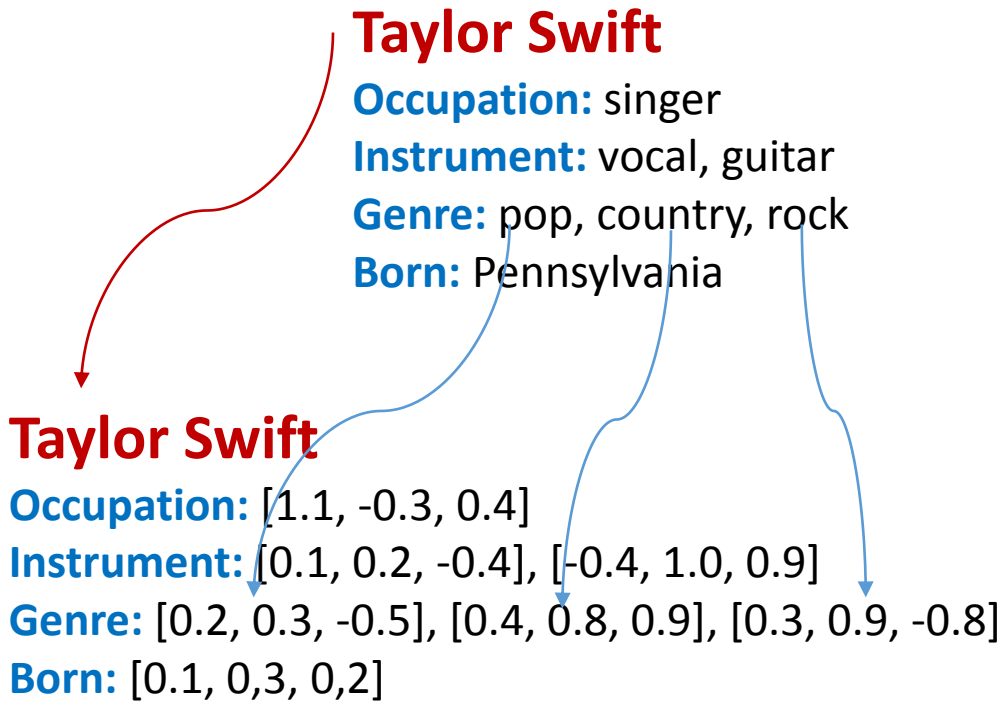
Overview

- **Basic Idea**
  - Entities in the same category may share some common attributes
- **Algorithm Framework**
  - Build a cluster system over the entities in KB
  - Apply our basic idea in each cluster to find missing attributes



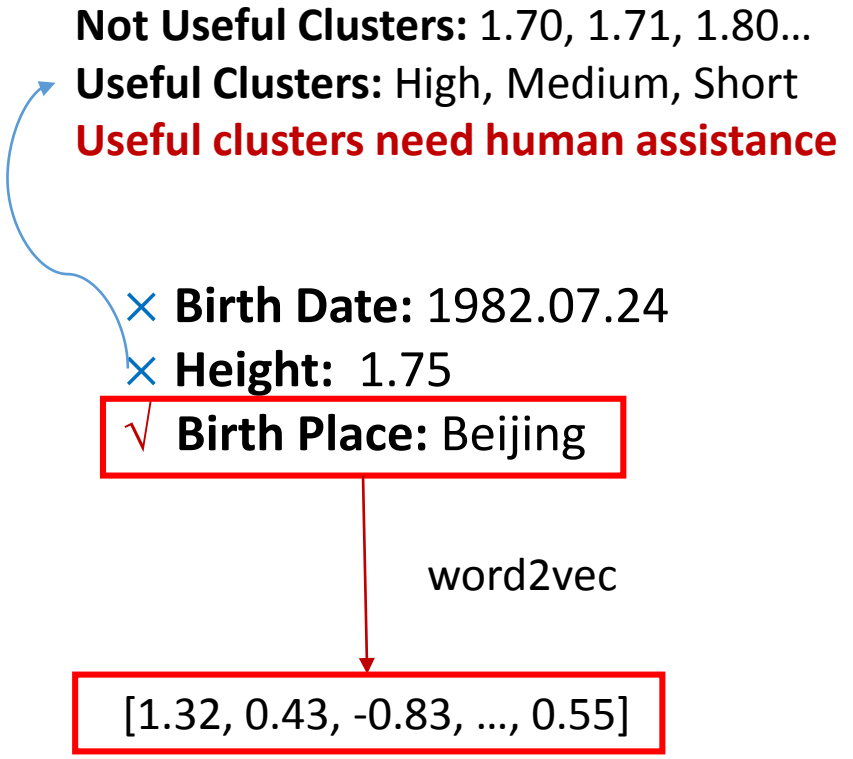
Building Clustering System

- **Entity Representation**
  - Each entity has several (attribute, value) pairs
  - The value of each attribute can be represented as a vector (explain later)
  - Each entity can be represented as a set of vectors, and each vector is an attribute value



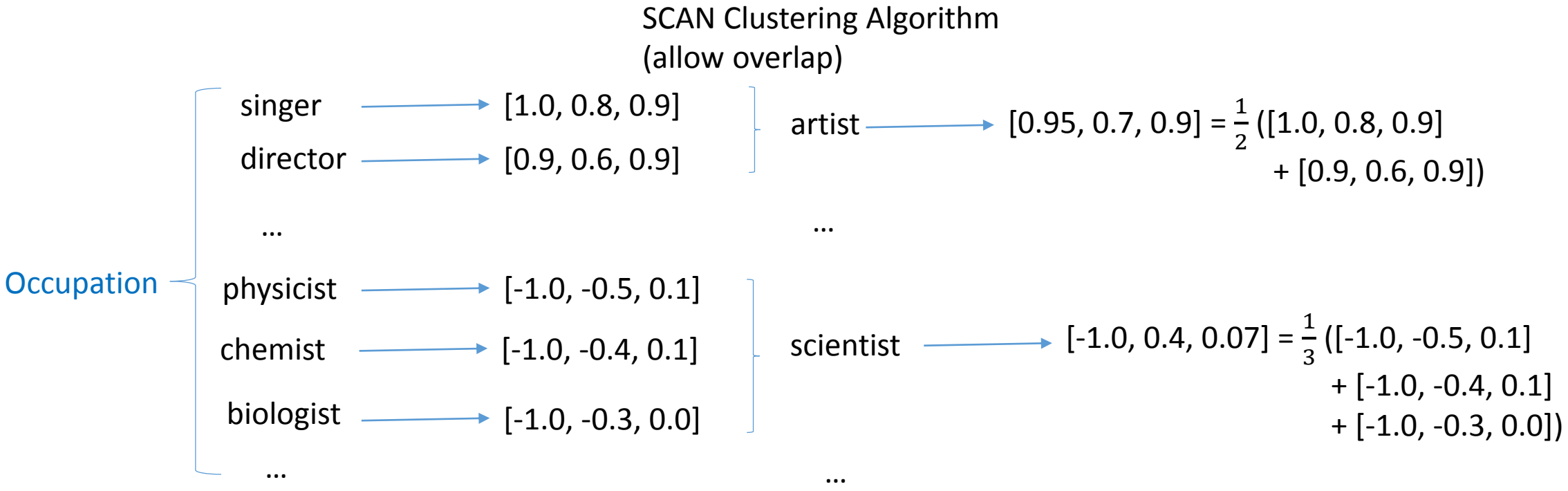
Building Clustering System

- **How to Acquire Attribute Value Vector?**
  - Numeric values and date values are not very useful
  - We only consider string values when clustering
  - Use word2vec to convert words or phrases in the attribute value into vectors
  - If a value have several words or phrases, will simple average them up



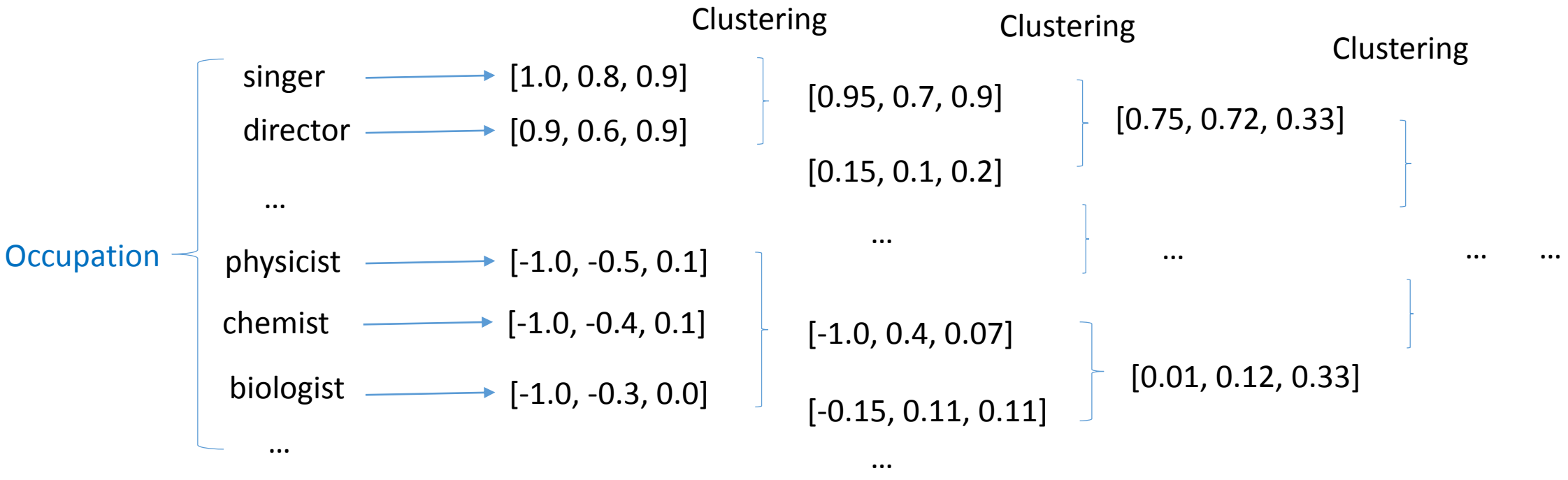
Building Clustering System

- **How to cluster entities?**
  - **Clustering entities directly is hard, since it contains so many vectors**
  - **Instead, we cluster attribute values within each attribute**



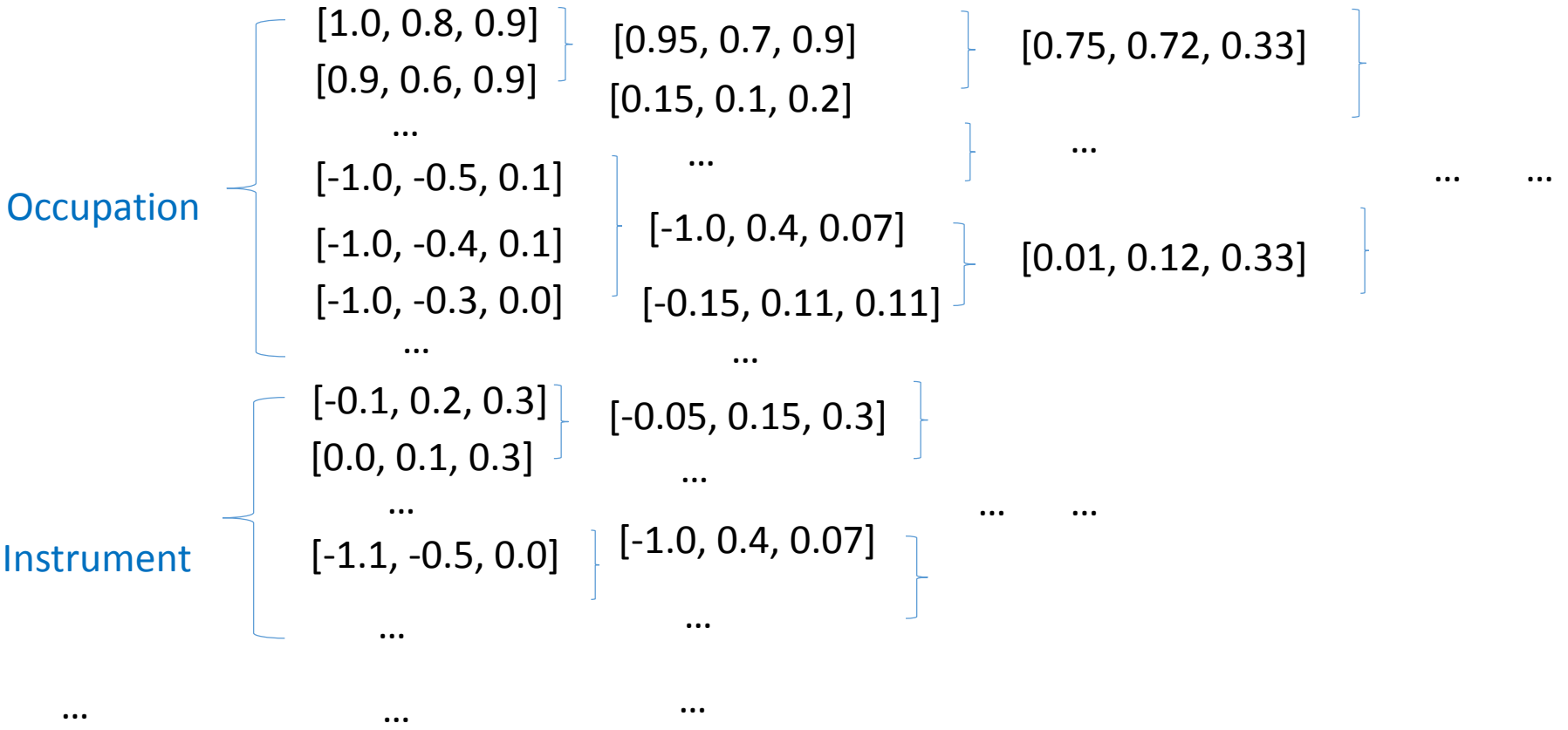
Building Clustering System

- **Clustering Within Attribute**
  - **Keep clustering to form hierarchical structure**
  - **The average height of the clustering system is about 4 layers**



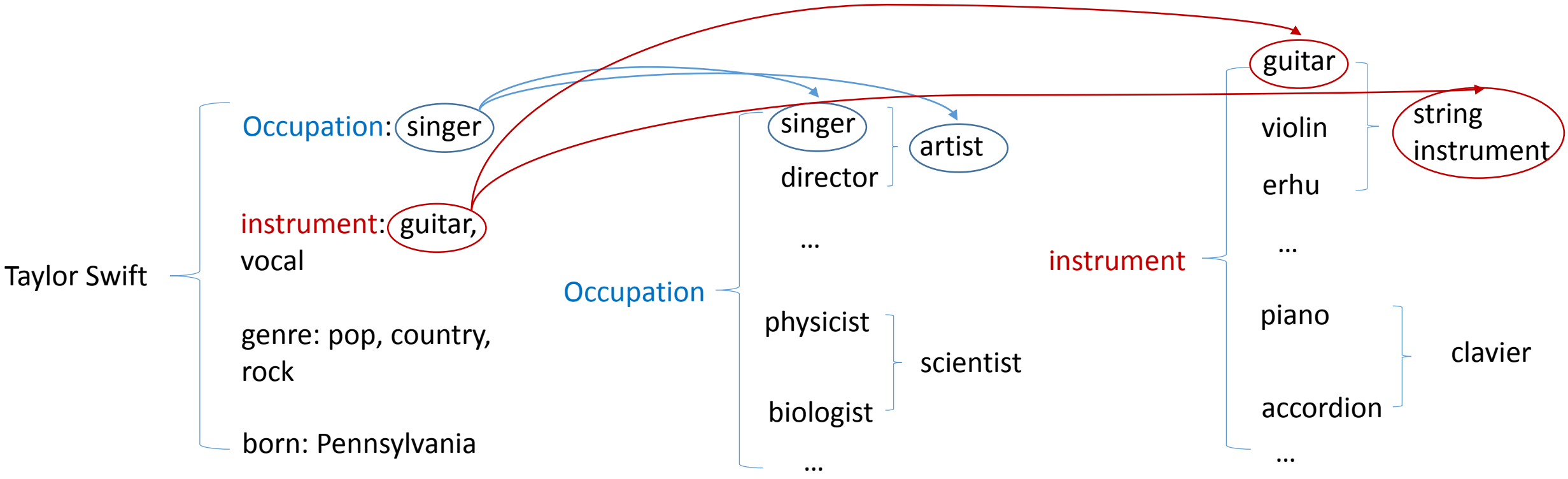
Building Clustering System

- **Clustering Within Attribute**
  - **Build clustering hierarchy within each attribute**



Building Cluster System

- **Assign entities to clusters**
  - **Assign entities to clusters according to its attribute value**

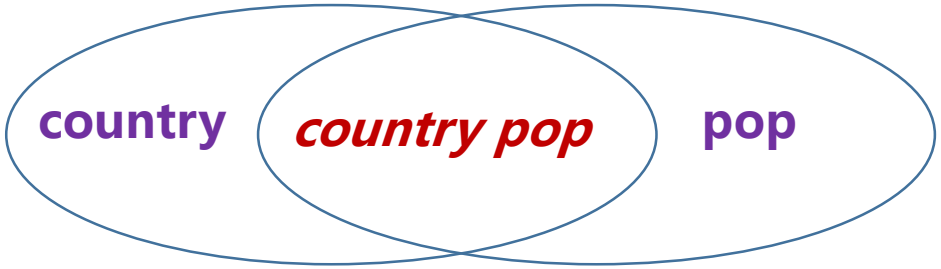


Building Cluster System

- **Intersection of different clusters**
  - The intersection of different clusters is also meaningful
  - We will keep the new cluster only when its size is large enough (contains a fair number of entities)

Within attribute:

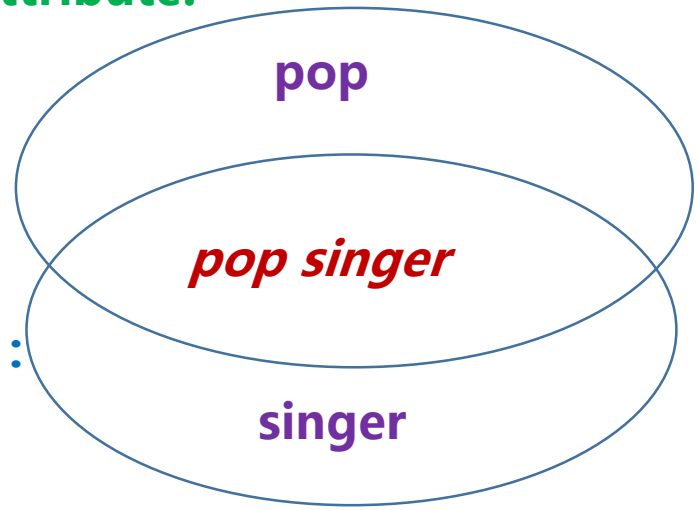
Genre :



Between attribute:

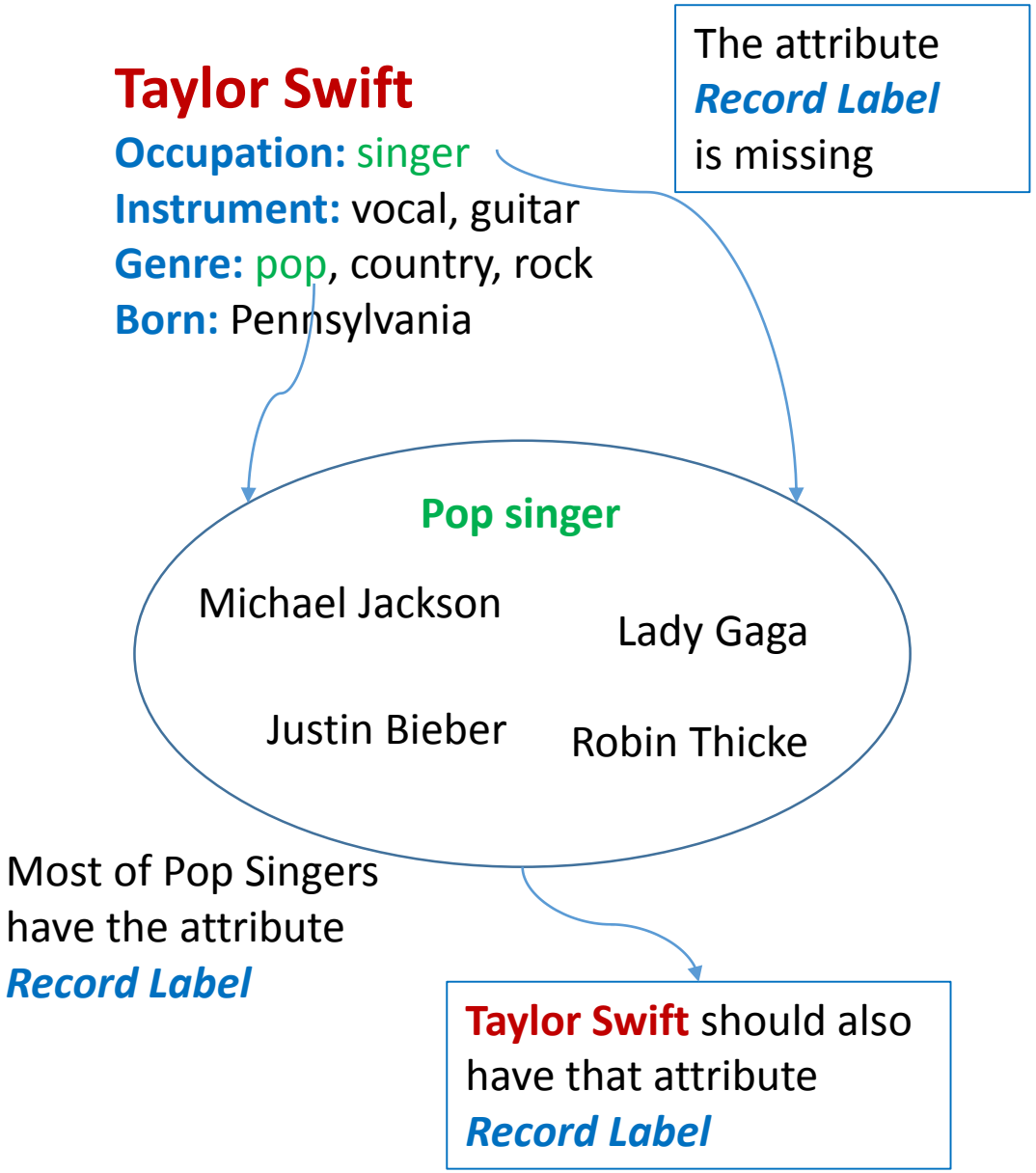
Genre :

Occupation :



Detecting missing attributes

- **Old entities**
  - **Exist in the KB** when building the cluster system
  - Already assigned clusters to them
  - Simply apply our basic idea



Detecting missing attributes

- **New entities**
  - **Not exist in the KB** when building the cluster system
  - Find clusters for them first

Justin Bieber
Occupation: singer
Instrument: vocal, guitar
Genre: pop, R&B
Born: London

Justin Bieber
Occupation: [1.1, -0.3, 0.4]
Instrument: [0.1, 0.2, -0.4], [-0.4, 1.0, 0.9]
Genre: [0.2, 0.3, -0.5], [0.1, 0.8, 0.2]
Born: [-0.1, 0,8, 0,9]

Vector Similarity Match

Within Occupation :

singer
[1.1, -0.3, 0.4]

Should also belong to cluster artist, and other clusters contains cluster singer

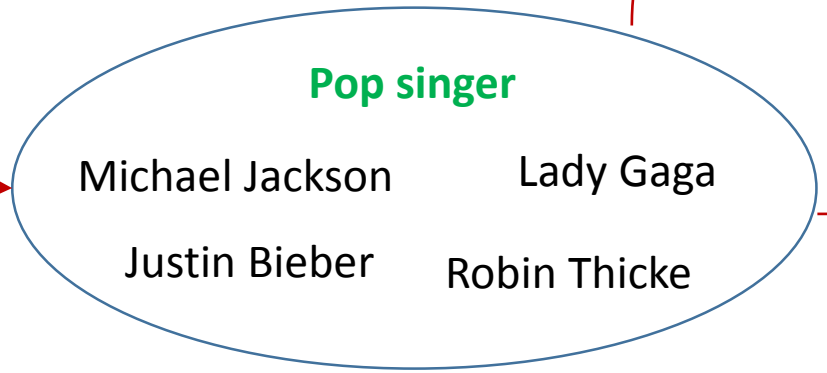
Summary

Represent entity as a set of vectors

Taylor Swift
Occupation: singer
Instrument: vocal, guitar
Genre: pop, country, rock
Born: Pennsylvania

Taylor Swift
Occupation: [1.1, -0.3, 0.4]
Instrument: [0.1, 0.2, -0.4], [-0.4, 1.0, 0.9]
Genre: [0.2, 0.3, -0.5], [0.4, 0.8, 0.9], [0.3, 0.9, -0.8]
Born: [0.1, 0,3, 0,2]

Assign clusters to entities



Generated From

Build clustering system within attributes AND interest these clusters

Occupation

[1.0, 0.8, 0.9]
[0.9, 0.6, 0.9]
[0.95, 0.7, 0.9]
[0.15, 0.1, 0.2]
[-1.0, -0.5, 0.1]
[-1.0, -0.4, 0.1]
[-1.0, -0.3, 0.0]
[-1.0, 0.4, 0.07]
[-0.15, 0.11, 0.11]

Instrument

[-0.1, 0.2, 0.3]
[0.0, 0.1, 0.3]
[-1.0, 0.4, 0.07]
[-1.1, -0.5, 0.0]

Detect missing attributes based on the Basic Idea

The attribute **record label** is missing

Experiment

- **Dataset**
  - 20,000 randomly sampled person entities in DBpedia
- **Evaluation**
  - First Method: randomly delete one the attribute of an entity, see if our method can find this attribute back or not
  - Second Method: human evaluation, see if the proposed missing attributes are reasonable or not

First Method

Taylor Swift

Occupation: singer
Instrument: vocal, guitar
Genre: pop, country, rock
Born: Pennsylvania
Record Label: Big Machine

DELETE!

Can our algorithm find the attribute Record Label back?

Second Method

Missing Attribute Proposed for Taylor Swift

√ **Record Label**
√ **Birth date**
× **Allegiance** (for military people)

Are these proposed missing attributes reasonable?

Experiment

- **Comparison method: Z. Abedjan and F. Naumann.(2013)**
  - First evaluation method
  - Top 5 means the algorithm proposes 5 most probable missing attributes, see if the deleted attribute is contained in
  - The metric is precision (if the deleted attribute is contained, then a match)

	Top1	Top5	top10
Our Method	84.43%	95.36%	96.05%
Abedjan & Naumann	NA	51.00%	71.40%

Experiment

- **Comparison of old entities and new entities**
  - First evaluation method
  - May be not fair when most of the proposed missing attributes are reasonable, except that the deleted one has a lower rank

	Top1	Top5	top10
Old Entities	84.43%	95.36%	96.05%
New Entities	33.86%	45.45%	46.07%

Not Fair Case:

Proposed Missing Attributes (Top 5):

1. …(good proposal)
2. …(good proposal)
3. …(good proposal)
4. …(not good)
5. …(good proposal)

…

18. Deleted Attribute

Experiment

- **Human Evaluation**
  - Randomly choose 1000 old entities and 1000 new entities
  - Propose all the attributes with a score higher than the threshold (no more than 10)

	Old Entity	New Entity
Precision	96.72%	97.09%

Conclusion

- **Performance**
  - Our method has a high precision, more than 95% suggested attributes are reasonable
  - Our method is good enough to be used in real world
- **Future Work**
  - Try Chinese data
  - Combine our method with relation extraction

Q & A