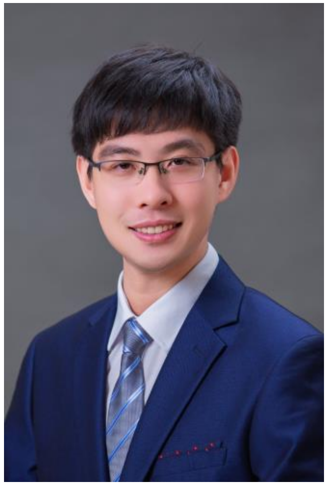
Web Science

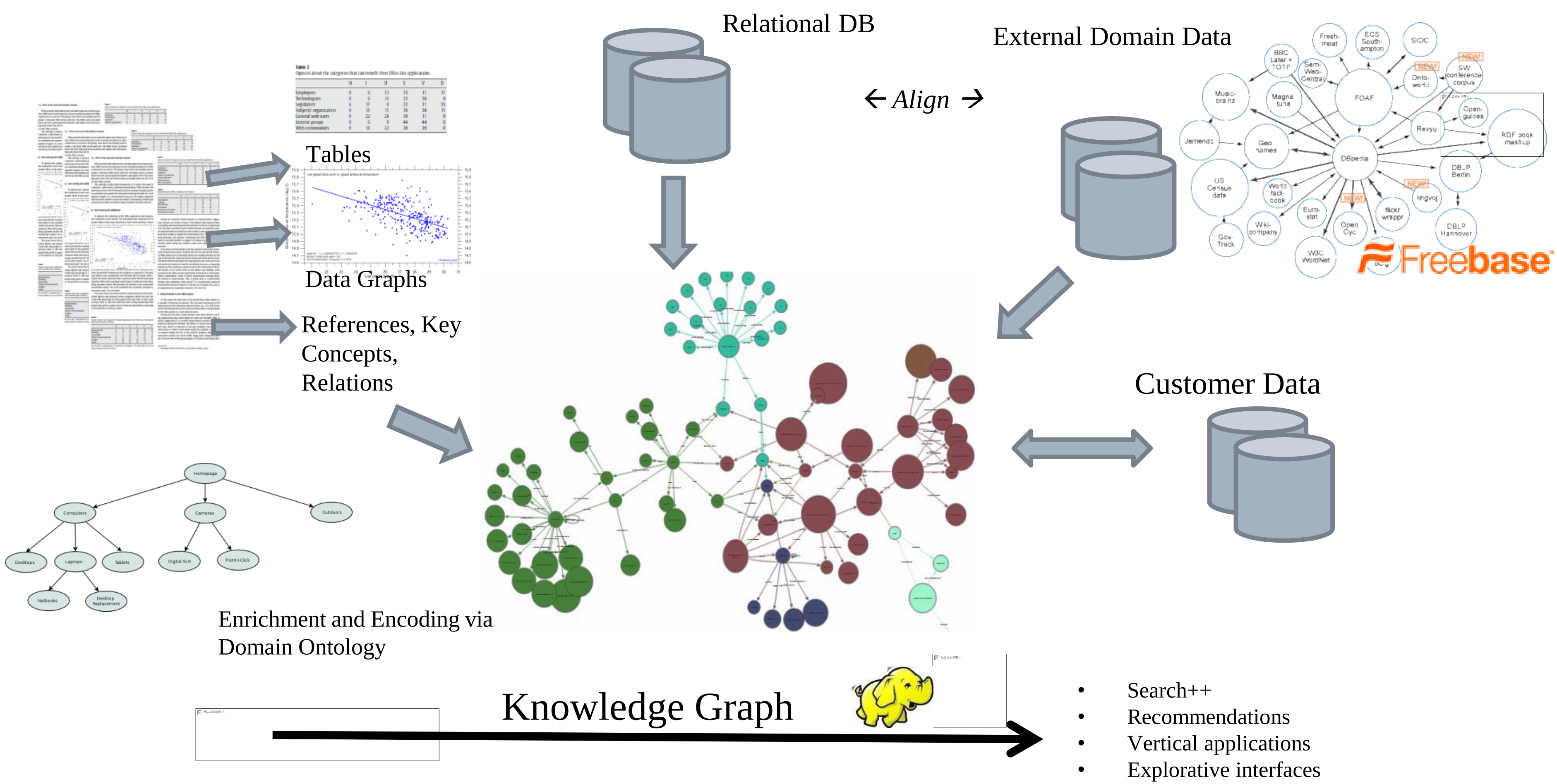
Prof. Tianxing Wu
School of Computer Science and Engineering
Southeast University

Email: tianxingwu@seu.edu.cn
Contact: 15077889931



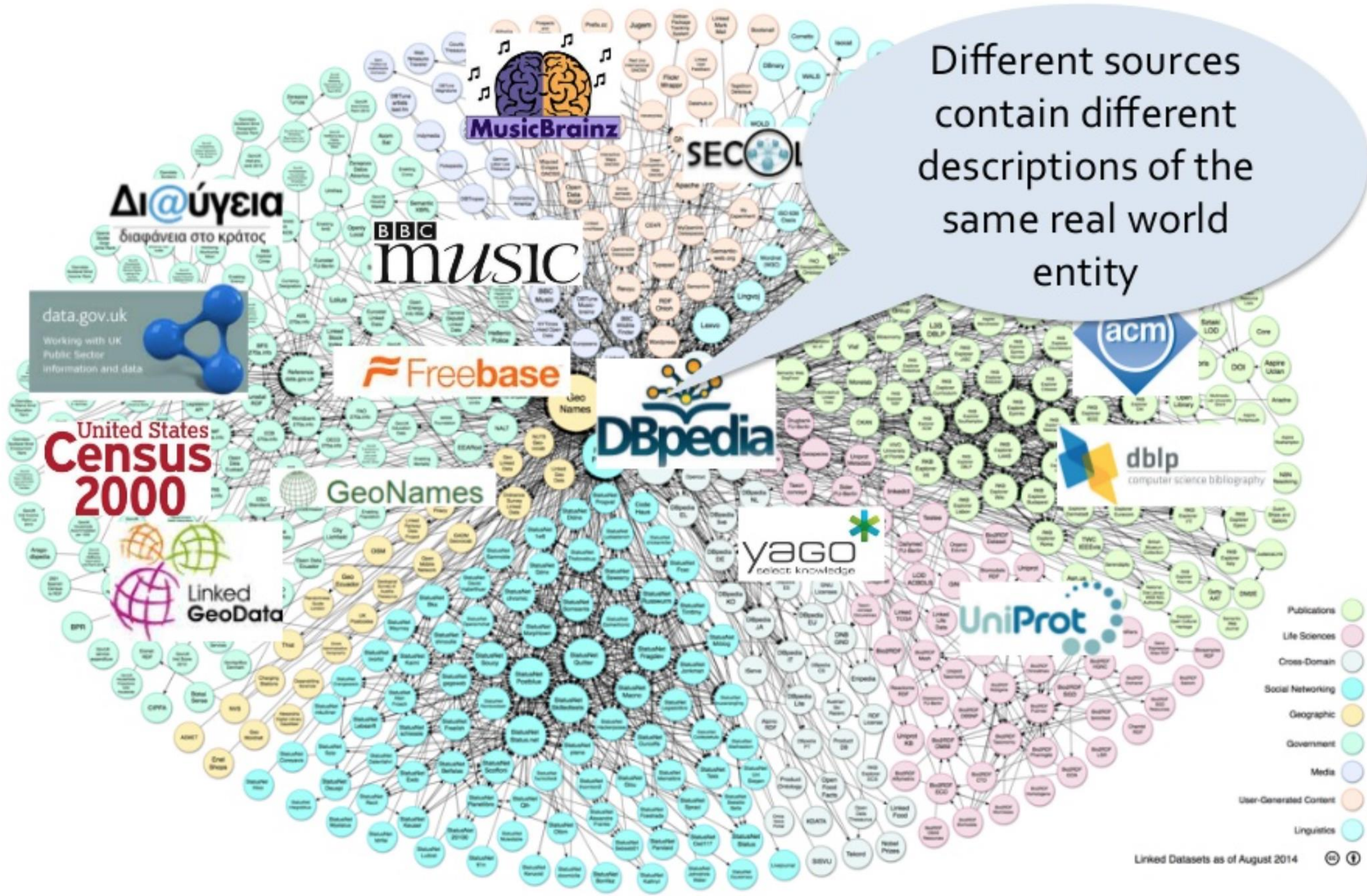
Knowledge Graph Alignment

Why do we need Knowledge Graph Alignment?



Knowledge graph construction needs to fuse the data from multiple sources!

Why do we need Knowledge Graph Alignment?



*Adapted from Suchanek & Weikum tutorial@SIGMOD 2013

Identifying the same entity with different descriptions from multiple sources is the key of knowledge graph alignment!

Why do we need Knowledge Graph Alignment?

"Basically, we're all trying to say the same thing."

Why do we need Knowledge Graph Alignment?

- The Same Entity in Multiple Sources

Videos about the St. of Liberty

Wiki pages about the St. of Liberty

General and lexicographical information about the St. of Liberty

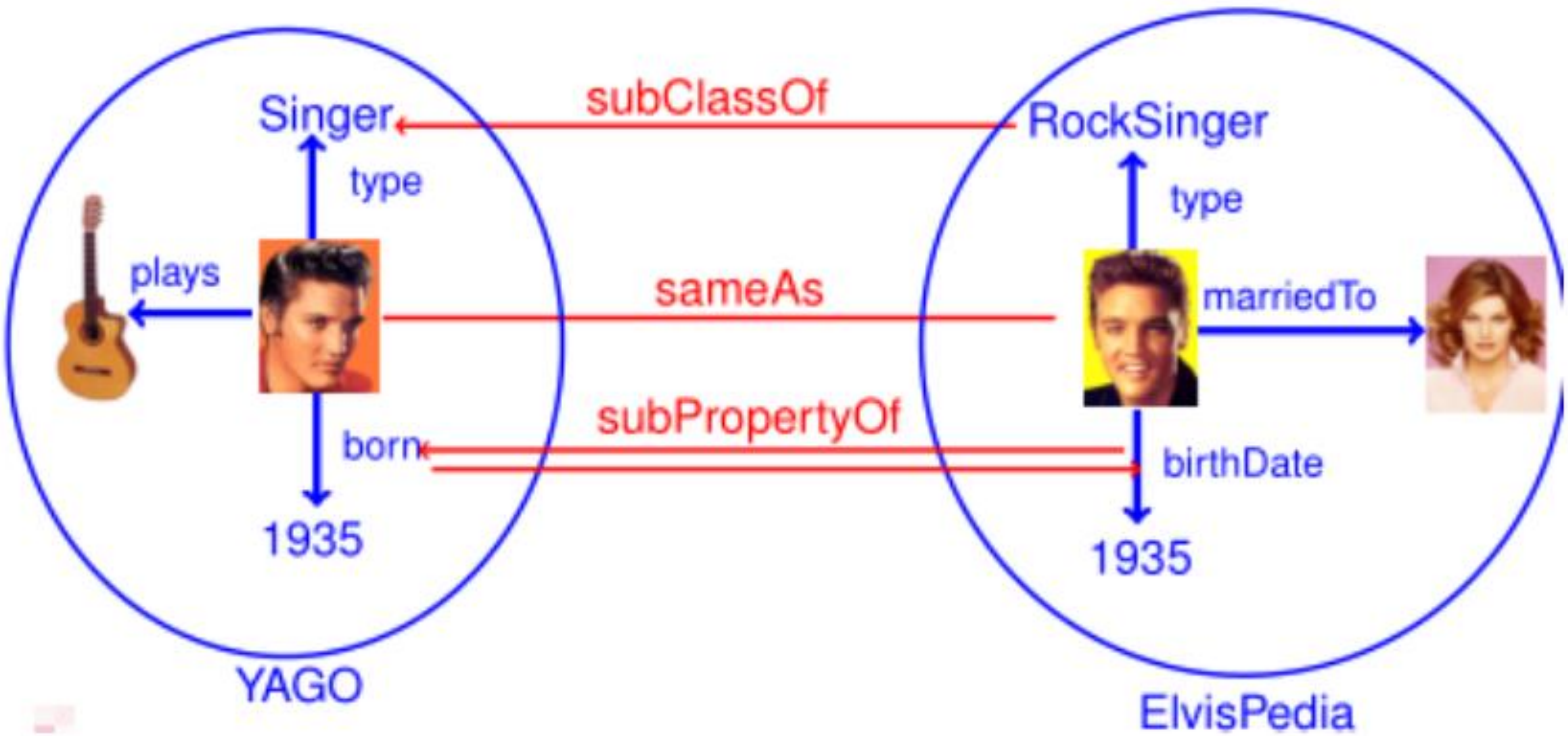
Images of the St. of Liberty

General information about the St. of Liberty

Articles about the St. of Liberty

Why do we need Knowledge Graph Alignment?

- The Same Entity in Multiple Sources
- Knowledge graph alignment identifies relationships between classes (equivalence, subClassOf, and etc.), properties (equivalence, subPropertyOf, and etc.), and instances (sameAs).



Why do we need Knowledge Graph Alignment?

- The Same Entity in Multiple Sources

Baidu Baike

baidu:大熊猫	
baidu:标签	“大熊猫”
baidu:拉丁学名	“Ailuropoda melanoleuca”
baidu:纲	baidu:哺乳纲
…	

Hudong Baike

hudong:大熊猫	
hudong:中文学名	“大熊猫”
hudong:二名法	“Ailuropoda melanoleuca”
hudong:纲	hudong:哺乳纲
…	

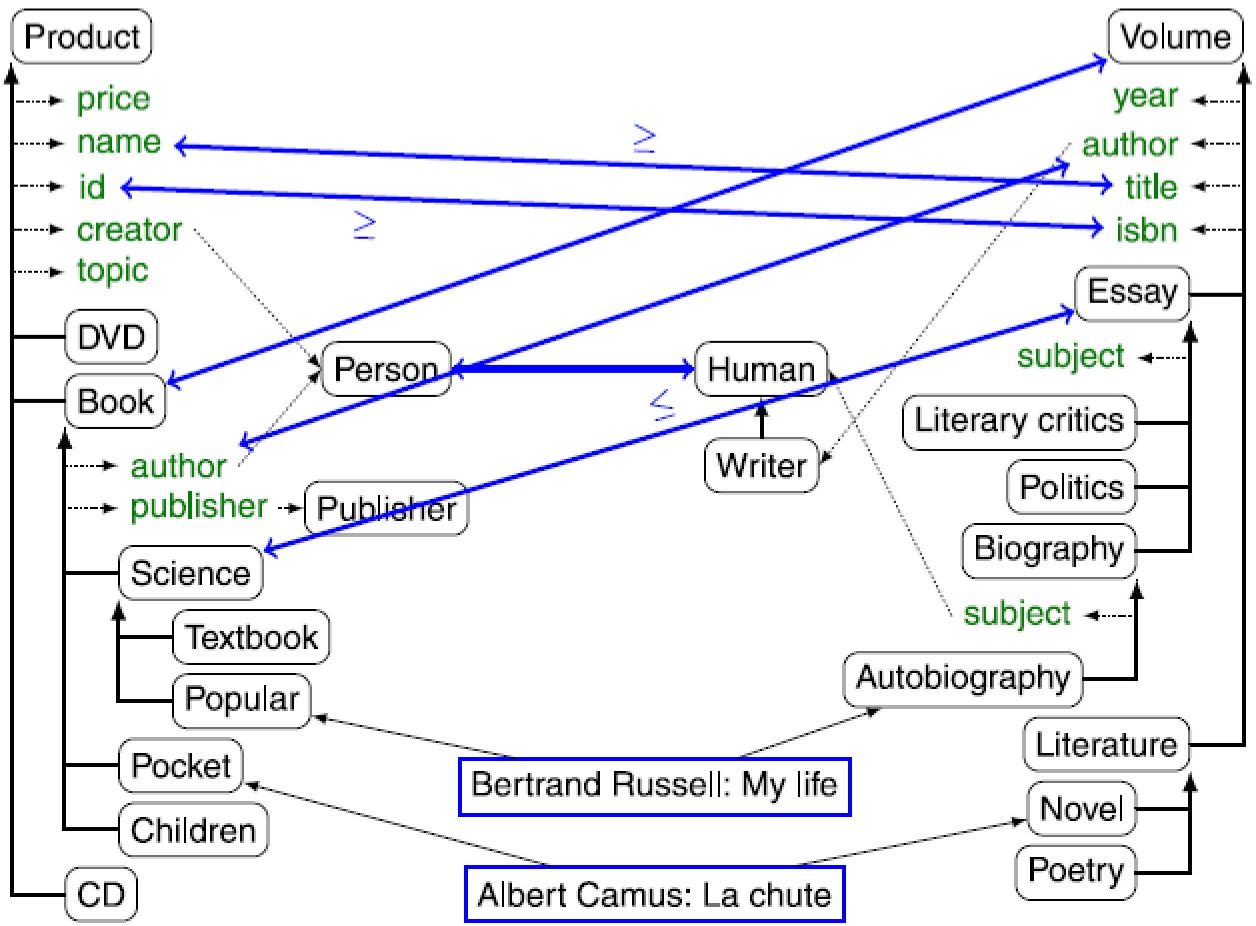
Knowledge Graph Alignment

- Knowledge Graph Alignment consists of:
  - ontology matching (i.e., schema matching),
  - instance matching.

Ontology Matching (本体匹配)：It is the process of finding correspondences (i.e., relationships) between classes (or properties) of different ontologies.

Ontology Matching

- Example:



Book = Volume
name ≥ title
id ≥ isbn
author = author
Person = Human
Science ≤ Essay

equivalence (=)
more general (⩾)
less general (⩽)

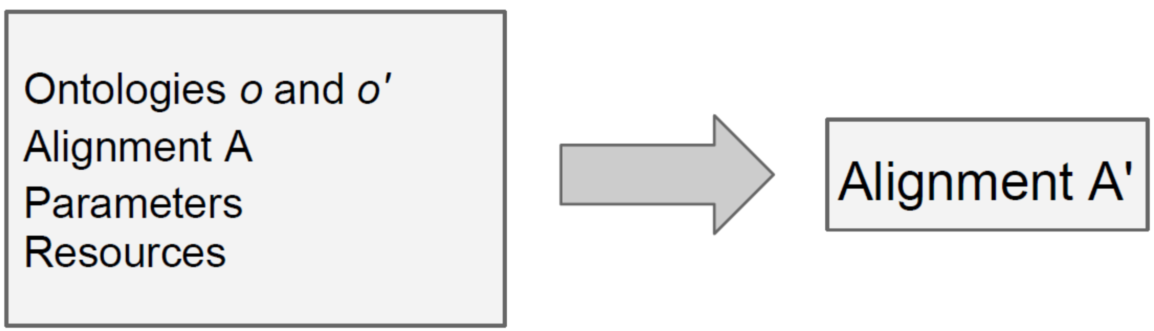
Benefits of Ontology Matching

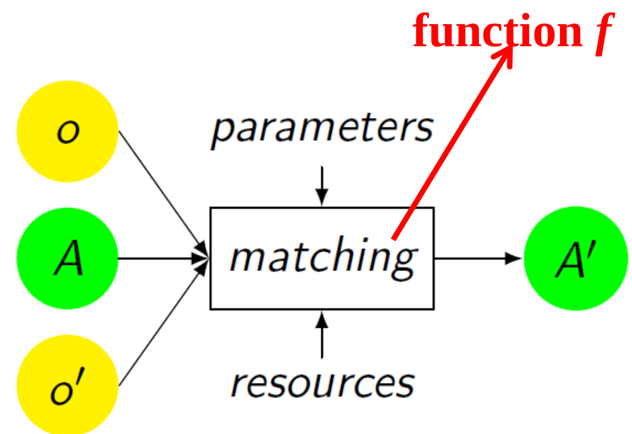
- Creating global ontologies from local ontologies
- Reuse information between ontologies
- Dealing with heterogeneity
- Queries across multiple distributed resources

Ontology Matching Process

Definition (Matching process) The matching process can be seen as **a function *f*** which, from a pair of ontologies to match *o* and *o*, an input alignment *A*, a set of parameters *p* and a set of resources *r*, returns an alignment *A'* between these ontologies:

$$A' = f(o, o', A, p, r)$$





parameters: weight, threshold…

resources: common sense knowledge, domain-specific thesauri…

Ontology Matching Techniques

- Element-level matching techniques
  - Analysing entities or instances in isolation
  - Ignoring their relations with other entities or their instances
- Structure-level techniques
  - Analysing how entities or their instances appear together in a structure (e.g. by representing ontologies as a graph)

Element-level Matching Techniques: String-based

- Prefix
  - takes as input two strings and checks whether the first string starts with the second one
  - net = network; but also hot = hotel
- Suffix
  - takes as input two strings and checks whether the first string ends with the second one
  - ID = PID; but also word = sword

Element-level Matching Techniques: String-based

Edit distance - **Levenshtein distance is used here**

- takes as input two strings and calculates the number of edition operations, (e.g., insertions, deletions, substitutions) of characters required to transform one string into another
- normalized by length of the maximum string

Dis(NKN, Nikon)= $\mathrm{N}\cancel{\mathrm{i}}\mathrm{K}\cancel{\mathrm{o}}\mathrm{N}/5 = 2/5 = 0.4$

Dis(editeur, editor)= $\mathrm{edit}\frac{e}{o}\cancel{\mathrm{u}}\mathrm{r}/7 = 2/7 = 0.43$

Element-level Matching Techniques: String-based

- N-gram
  - takes as input two strings and calculates the number of common n-grams (i.e., sequences of n characters) between them, normalized by $max(length(string1), length(string2))$

Example:
trigrams(nikon) = {nik, iko, kon}
trigrams(nike) = {nik, ike}

sim(nikon, nike)= 1/3

Exercise

Compute the trigram based similarity between two strings University and Universe.

Element-level Matching Techniques: Language-based

- Tokenization
  - parses names into tokens by recognizing punctuation, cases
  - Hands-Free_Kits $\rightarrow$ $\langle$ hands, free, kits $\rangle$
- Lemmatization
  - analyses morphologically tokens in order to find all their possible basic forms
  - Kits $\rightarrow$ Kit

Element-level Matching Techniques: Language-based

- Elimination
  - discards "empty" tokens that are articles, prepositions, conjunctions, etc.
  - a, the, by, type of, their, from

Element-level Matching Techniques: Resource-based

- WordNet
  - $A \sqsubseteq B$ if A is a hyponym of B
    - Brand $\sqsubseteq$ Name
  - $A = B$ if they are synonyms
    - Quantity $=$ Amount
  - $A \perp B$ if they are antonyms or the siblings in the part of hierarchy
    - Microprocessors $\perp$ PC Board

Element-level Matching Techniques: Constraint-based

- Datatype comparison
  - *integer* $<$ *real*
  - $date \in [1/4/2005 \ 30/6/2005] < date[year = 2005]$
  - $\{a, c, g, t\}[1 - 10] < \{a, c, g, u, t\}+$
- Multiplicity comparison
  - $[1 \ 1] < [0 \ 10]$

Multiplicity:[minCardinality maxCardinality] of a property

Can be turned into a distance by estimating the ratio of domain coverage of each datatype.

Structure-level Matching Techniques

- **Graph-based Techniques:**
  - consider the input ontologies as labeled graphs;
  - if two nodes from two ontologies are similar, their neighbors may also be somehow similar (similar subclasses, superclasses, and properties).

- **Taxonomy-based Techniques:**
  - are also graph algorithms which consider only is-a relations between classes;
  - is-a links connect terms that are already similar, therefore their neighbors may be somehow similar.

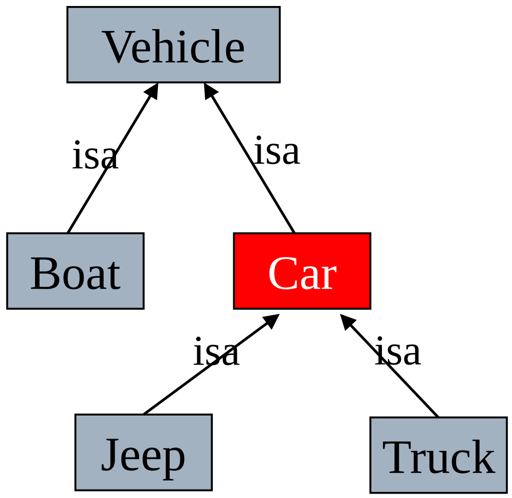
Structure-level Matching Techniques

- Model-based Techniques
  - handle the input ontologies based on its semantic interpretation (e.g, model-theoretic semantics);
  - if two classes (or properties) are the same, then they share the same interpretation.

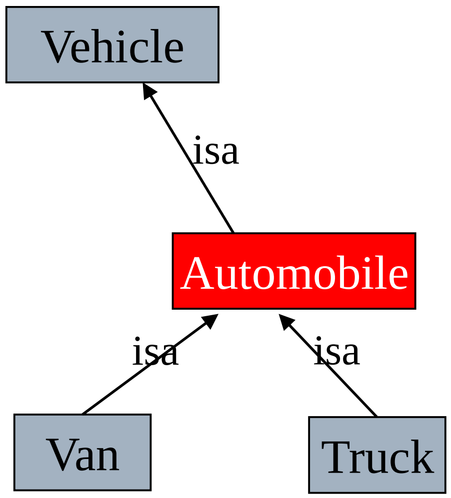
- Instance-based Techniques:
  - compare sets of instances of classes to decide if these classes match or not (i.e., exist equivalence or subClassOf relations).

Exercise

Compute the Jaccard similarity between the first-order neighbor classes of the classes Car and Automobile from different ontologies.



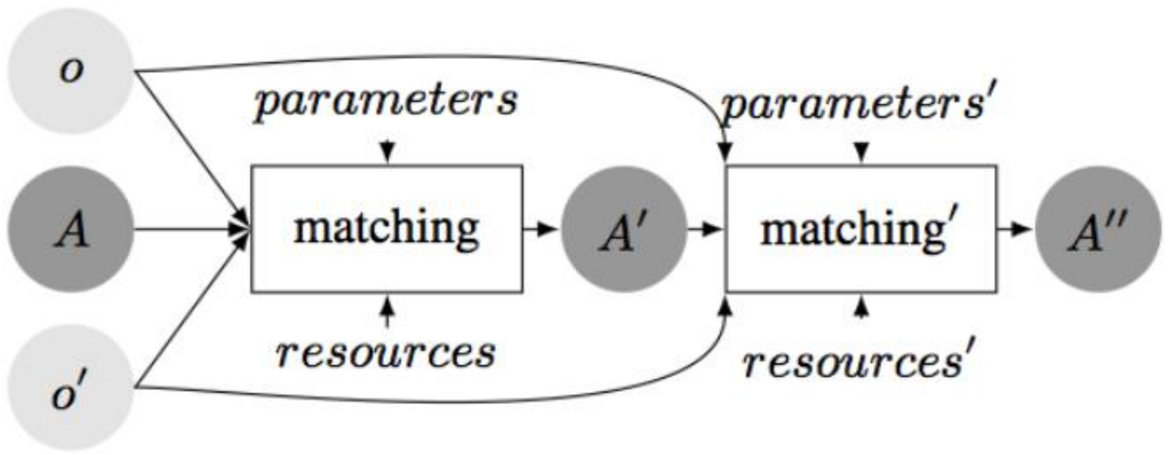
Ontology 1



Ontology 2

Matcher Composition

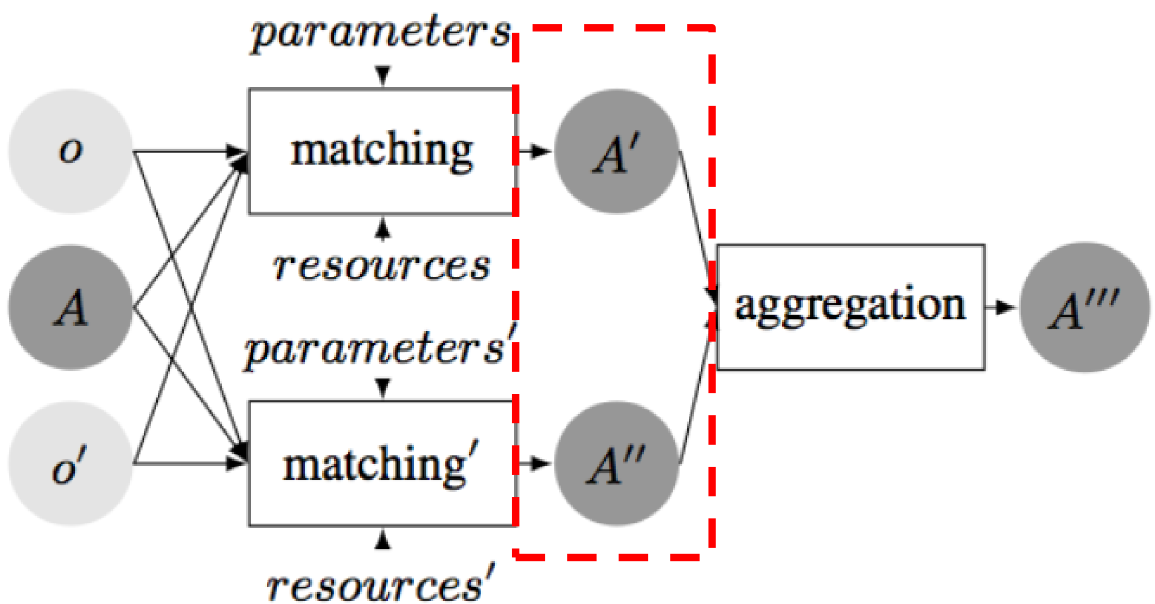
- Sequential composition of matchers



Problem: error accumulation and low coverage.

Matcher Composition

- Parallel composition of matchers



  - e.g. A single similarity measure composed by the similarity obtained from their names, the similarity of their superclasses, the similarity of their instances and that of their properties

A Real-World Case: Book Ontology Matching

Given two ontologies O_1 and O_2, generate candidate matched classes by pairing any two classes from the two ontologies. A pair of candidate matched classes is denoted as (C_{1k}, C_{2P}), and note that (C_{2p}, C_{1k}) is a different pair since we need to measure the asymmetric similarities between classes.

- String Similarity:

$$CLsim(C_{1k}, C_{2p}) = \frac{LCS(l(C_{1k}), l(C_{2p}))}{\left| l(C_{1k}) \right|}$$

where LCS means the length of the longest common substring, $l(\cdot)$ returns the label of the class, and $|\cdot|$ returns the length of the input label.

Example:

```
ABABC
  |||
 BABCA
  |||
  ABCBA
```

the longest common substring: ABC
length: 3

A Real-World Case: Book Ontology Matching

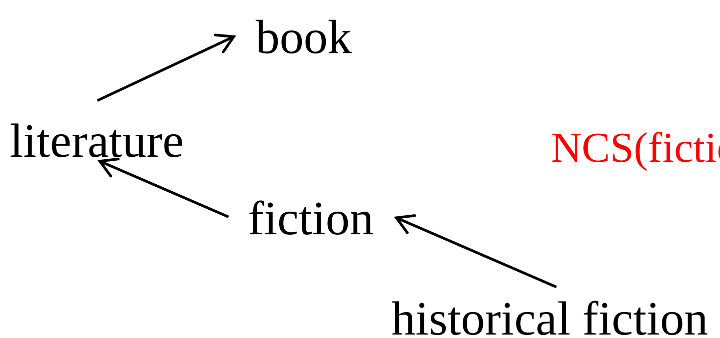
Given two ontologies O_1 and O_2, generate candidate matched classes by pairing any two classes from the two ontologies. A pair of candidate matched classes is denoted as (C_{1k}, C_{2P}), and note that (C_{2p}, C_{1k}) is a different pair since we need to measure the asymmetric similarities between classes.

- Neighbor Class Set Similarity:

$$NCSsim(C_{1k}, C_{2p}) = \frac{|NCS(C_{1k}) \cap NCS(C_{2p})|}{|NCS(C_{1k})|}$$

where $|\mathrm{NCS}(C_{1k}) \cap \mathrm{NCS}(C_{2p})|$ is the size of the intersection of $\mathrm{NCS}(C_{1k})$ and $\mathrm{NCS}(C_{2p})$, $\mathrm{NCS}(\cdot)$ returns the set of first-order neighbor classes in the given ontology.

Example:

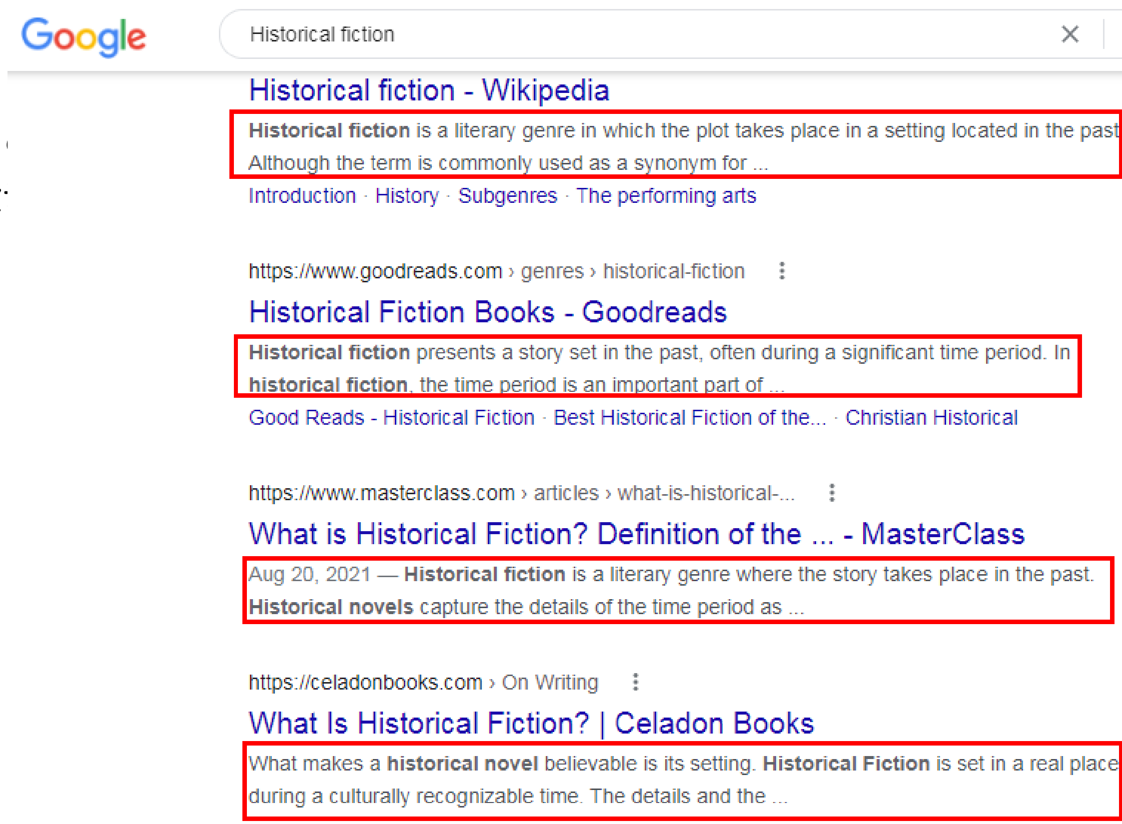


NCS(fiction)={literature, historical fiction}

A Real-World Case: Book Ontology Matching

Given two ontologies O_1 and O_2, generate candidate matched classes by pairing any two classes from the two ontologies. A pair of candidate matched classes is denoted as (C_{1k}, C_{2P}), and note that (C_{2p}, C_{1k}) is a different pair since we need to measure the asymmetric similarities between classes.

- Textual Context Similarity.
  - We submit the label $l(C)$ of a
    snippets as the textual contex



A Real-World Case: Book Ontology Matching

Given two ontologies O_1 and O_2, generate candidate matched classes by pairing any two classes from the two ontologies. A pair of candidate matched classes is denoted as (C_{1k}, C_{2P}), and note that (C_{2p}, C_{1k}) is a different pair since we need to measure the asymmetric similarities between classes.

- Textual Context Similarity.
  - We submit the label $l(C)$ of a class C to a search engine to acquire different snippets as the textual context;
  - In the top-k returned snippets of Web pages, the words co-occurred with $l(C)$ in the same sentence are extracted;
  - After removing the stopwords and the words with low frequency (e.g., less than 3), TF-IDF is adopted for weighting each word u:

$$w_u = tf_u \cdot idf_u$$

A Real-World Case: Book Ontology Matching

Given two ontologies O_1 and O_2 , generate candidate matched classes by pairing any two classes from the two ontologies. A pair of candidate matched classes is denoted as (C_{1k}, C_{2P}), and note that (C_{2p}, C_{1k}) is a different pair since we need to measure the asymmetric similarities between classes.

- Textual Context Similarity.
  - The textual context vector representation of a class C is denoted as: $TC(C)=<w_1(C), w_2(C),\ldots, w_n(C)>$, and n is the number of all words.
  - The textual context similarity between classes C_{1k} and C_{2p} is computed as:

$$TCsim(C_{1k}, C_{2p}) = \frac{\sum_{v=1}^{n} TC(C_{1k})_v \cdot TC(C_{2p})_v}{\sum_{v=1}^{n} TC(C_{1k})_v^{\ 2}}$$

A Real-World Case: Book Ontology Matching

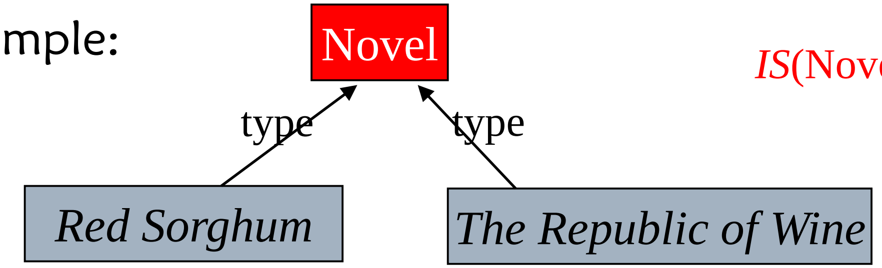
Given two ontologies O_1 and O_2 , generate candidate matched classes by pairing any two classes from the two ontologies. A pair of candidate matched classes is denoted as (C_{1k}, C_{2P}), and note that (C_{2p}, C_{1k}) is a different pair since we need to measure the asymmetric similarities between classes.

- Instance Set Similarity:

$$ISsim(C_{1k}, C_{2p}) = \frac{\left|IS(C_{1k}) \cap IS(C_{2p})\right|}{\left|IS(C_{1k})\right|}$$

where $IS(\cdot)$ returns the instance set of the class, and we can identify the same book instances using the ISBN number in the book domain.

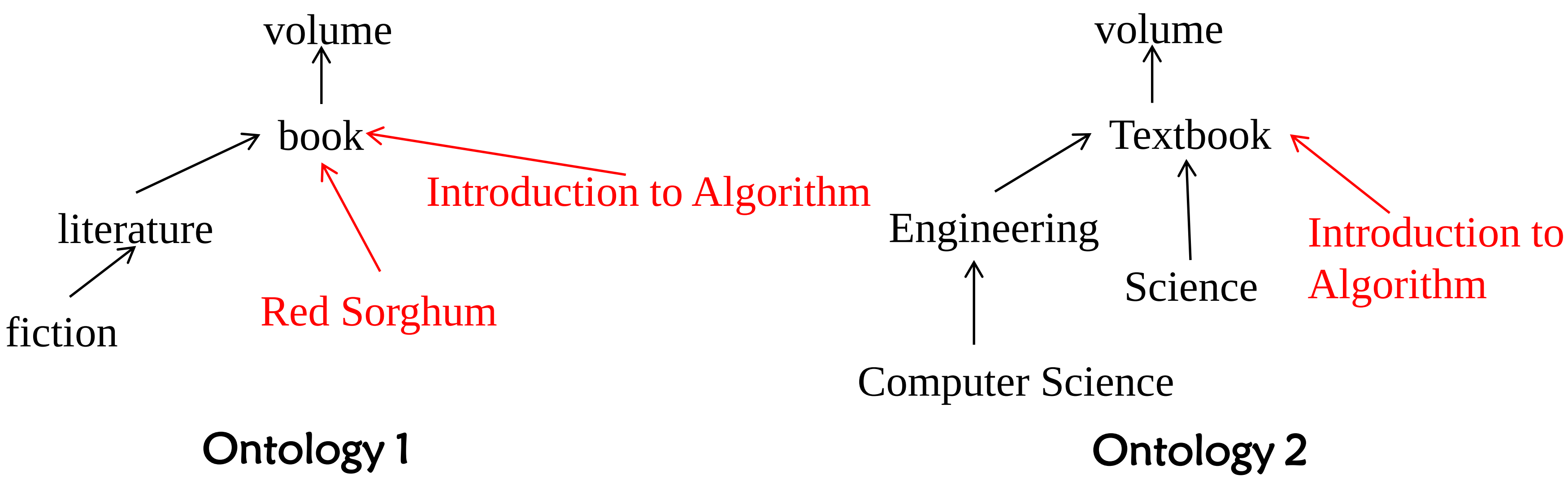
Example:



IS(Novel)={Red Sorghum, The Republic of Wine}

Exercise

Given two ontologies as follows, compute the String Similarity, Neighbor Class Set Similarity, Instance Set Similarity (introduced in the real-world case: book ontology matching) **on the class pairs (book, Textbook) and (Textbook, book), respectively.**



A Real-World Case: Book Ontology Matching

Given two ontologies O_1 and O_2 , generate candidate matched classes by pairing any two classes from the two ontologies. A pair of candidate matched classes is denoted as (C_{1k}, C_{2P}), and note that (C_{2p}, C_{1k}) is a different pair since we need to measure the asymmetric similarities between classes.

- Question: now we have String Similarity, Neighbor Class Set Similarity, Textual Context Similarity, and Instance Set Similarity, please tell which one belongs to the element-level matching techniques? Which one is a structure-level matching technique?

A Real-World Case: Book Ontology Matching

Given two ontologies O_1 and O_2, generate candidate matched classes by pairing any two classes from the two ontologies. A pair of candidate matched classes is denoted as (C_{1k}, C_{2P}), and note that (C_{2p}, C_{1k}) is a different pair since we need to measure the asymmetric similarities between classes.

- **Aggregate these similarities by a semi-supervised learning strategy: self-training, for binary classification on subClassOf relations:**
  - In each iteration, self-training accepts the labeled data as training data and learns a classifier.
  - Then the classifier is applied to the unlabeled data and adds class pairs of high confidence to the labeled data to train a new classifier for the next iteration.
  - The whole process will terminate if the difference between the predicted labels of the class pairs given by classifiers in the two consecutive iterations is smaller than a threshold or the maximal number of iterations is achieved.

A Real-World Case: Book Ontology Matching

Given two ontologies O_1 and O_2 , generate candidate matched classes by pairing any two classes from the two ontologies. A pair of candidate matched classes is denoted as (C_{1k}, C_{2P}), and note that (C_{2p}, C_{1k}) is a different pair since we need to measure the asymmetric similarities between classes.

- **Aggregate these similarities by a semi-supervised learning strategy: self-training, for binary classification on subClassOf relations:**
  - The binary classifier can use SVM, Random Forest, Neural Networks, and etc.
  - In each iteration, rules are applied to filter out misclassified relations.

  RULE 1:
  Given two book classes C_{1k} and C_{2p} , if the label string $l(C_{1k})$ is the suffix of the label string $l(C_{2p})$, and $l(C_{2p})$ does not contain "与", "和", and "&", then C_{2p} is the subclass of C_{1k}.

  Example: 企业管理 subClassOf 管理

A Real-World Case: Book Ontology Matching

Given two ontologies O_1 and O_2 , generate candidate matched classes by pairing any two classes from the two ontologies. A pair of candidate matched classes is denoted as (C_{1k}, C_{2P}), and note that (C_{2p}, C_{1k}) is a different pair since we need to measure the asymmetric similarities between classes.

- **Aggregate these similarities by a semi-supervised learning strategy: self-training, for binary classification on subClassOf relations:**
  - The binary classifier can use SVM, Random Forest, Neural Networks, and etc.
  - In each iteration, domain-specific rules are applied to filter out misclassified relations

RULE 2:
Given two book classes C_{1k} and C_{2p} , if the label string $l(C_{2p})$ contains "与" or "和" or "&" , then using these symbols as separators to segment the label string $l(C_{2p})$. If one of the segmented strings and $l(C_{1k})$ are the same, then C_{1k} is the subclass of C_{2p}.

Example: 计算机 subClassOf 计算机与互联网

A Real-World Case: Book Ontology Matching

Given two ontologies O_1 and O_2 , generate candidate matched classes by pairing any two classes from the two ontologies. A pair of candidate matched classes is denoted as (C_{1k}, C_{2P}), and note that (C_{2p}, C_{1k}) is a different pair since we need to measure the asymmetric similarities between classes.

- **Aggregate these similarities by a semi-supervised learning strategy: self-training, for binary classification on subClassOf relations:**
  - With generated subClassOf relations, how to get equivalent classes?

OAEI

OAEI（Ontology Alignment Evaluation Initiative）本体对齐竞赛，用来评估各种本体对齐算法，以达到评估，比较，交流以及促进本体对齐工作的目的。OAEI每年举办一次，结果公布在官网上。

http://oaei.ontologymatching.org/

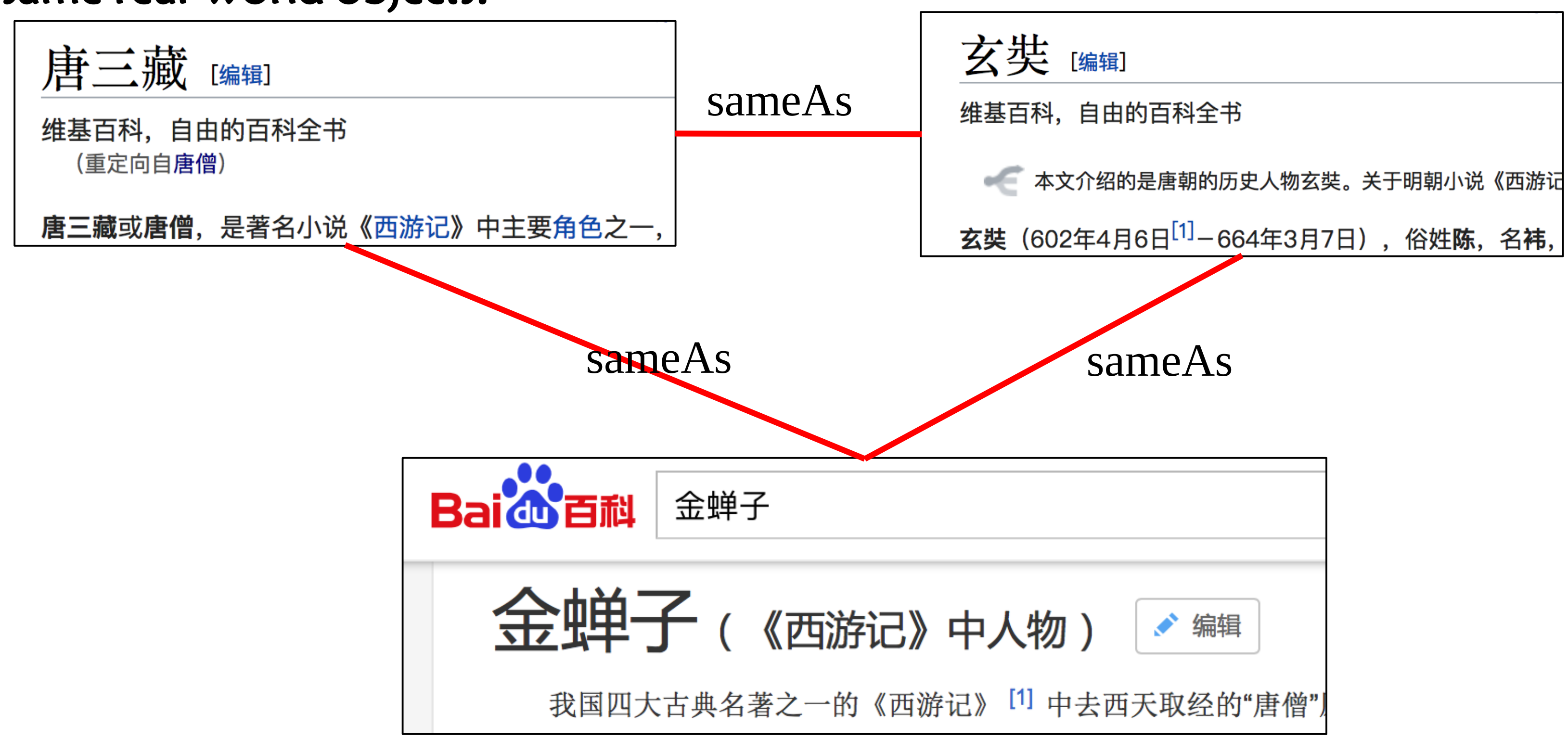
OAEI

序号	关注问题	关注点
1	Anatomy	解剖
2	conference	会议
3	Multifarm	不同语言会议数据
4	Interactive matching evaluation	交互式的匹配评估
5	Large Biomedical Ontologies	生物
6	Disease and Phenotype	疾病及症状匹配
7	Process Model Matching	在更多具体的领域上，系统性能测试
8	Instance Matching	实例匹配
9	HOBBIT Link Discovery	新增的，地理

Knowledge Graph Alignment

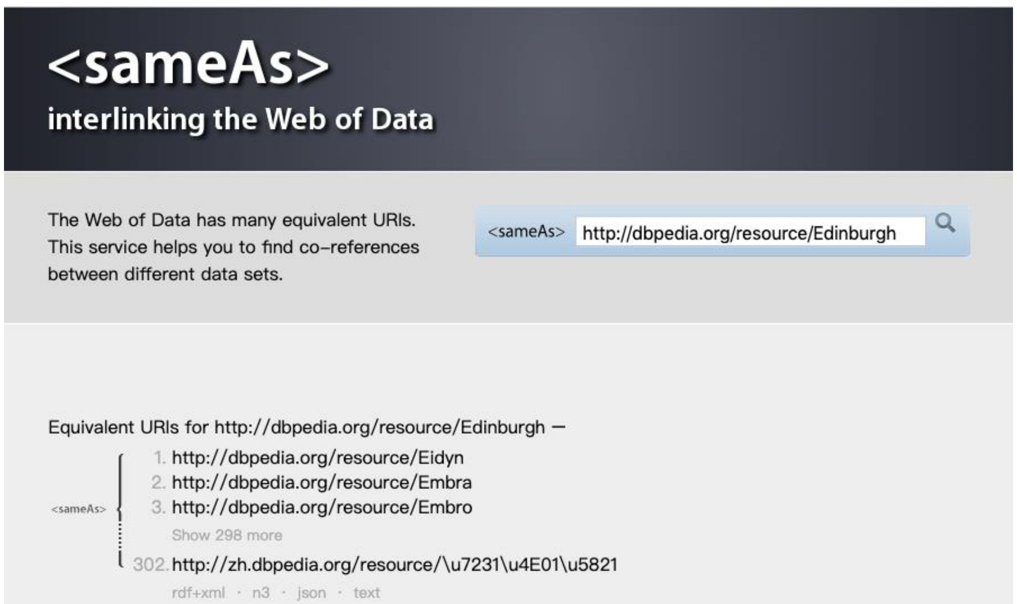
- Knowledge Graph Alignment consists of:
  - ontology matching (i.e., schema matching),
  - instance matching.

Instance Matching (实例匹配)：It is the process of finding different instances of the same real-world objects.



sameAs.org

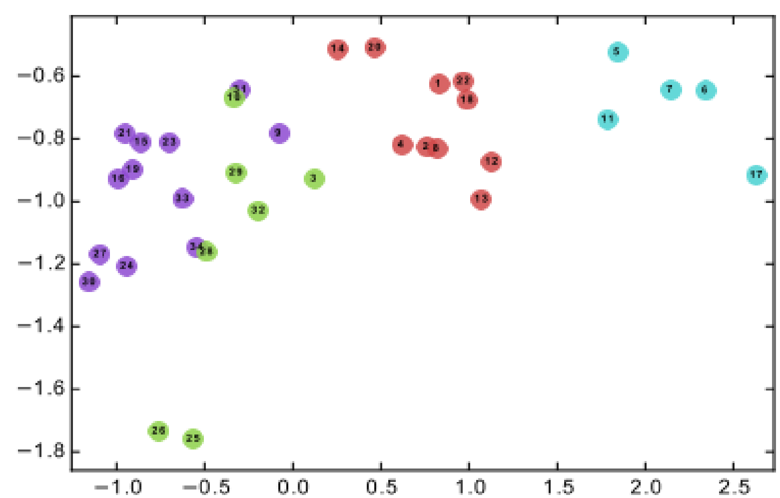
Currently serving 203,953,936 URIs which relate to over 53,054,359 apparently distinct entities.



http://go.bio2rdf.org/ http://purl.org/hcls/
http://moustaki.org/ http://rdf.dmoz.org/
http://doapstore.org/ http://dbpedia.org/
http://rdf.geospecies.org/ http://www.yr-bcn.es/pmika/
http://umbel.org/ http://downloads.dbpedia.org/
http://www.opencyc.org/ http://hcls.deri.org/
http://lingvoj.org/ http://www.cs.vu.nl/STITCH/rameau/
http://rkbexplorer.com/ http://airports.dataincubator.org/
http://telegraphis.net/ http://ontologi.es/rail/stations
http://data.linkedct.org/ http://discogs.dataincubator.org/
http://www.bbc.co.uk/music/ http://linkedgeodata.org/
http://data.nytimes.com/ http://bnb.data.bl.uk
http://d-nb.info http://data.bibsys.no
http://nektar.oszk.hu http://dbpedia.org
http://id.loc.gov http://id.ndl.go.jp
http://stitch.cs.vu.nl

Instance Matching with Knowledge Graph Embedding

- Embedding maps discrete variables to continuous vector representations;
- Embedding learning techniques has achieved great progress in CV, NLP, Speech Recognition, and etc.;
- Knowledge Graph Embedding aims to map entities and relations to continuous vector representations.

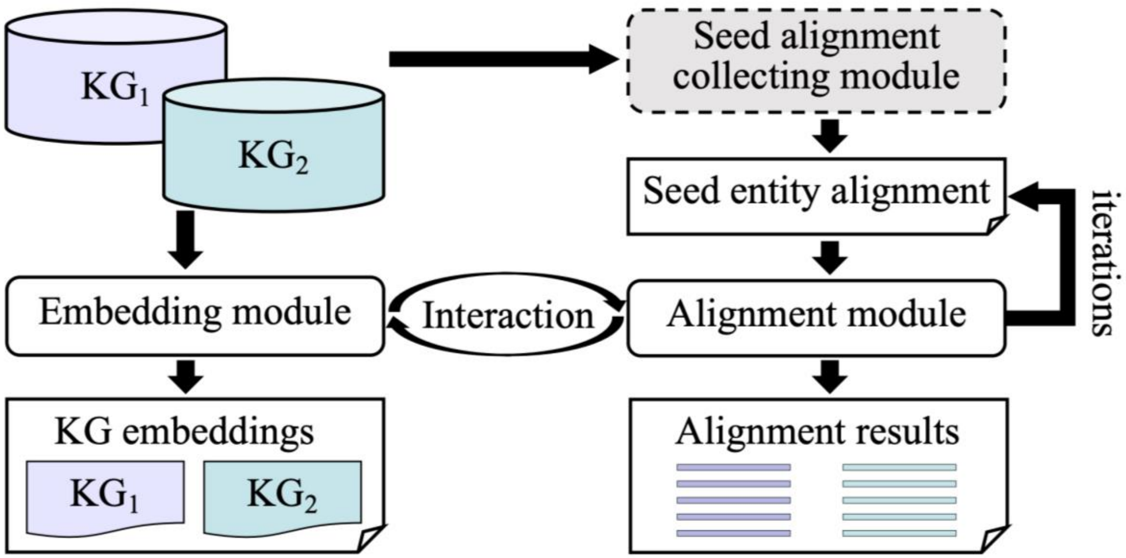


Instance Matching with Knowledge Graph Embedding

Conventional approaches are challenged by the symbolic, linguistic and schematic heterogeneity of independently-created KGs

Embedding-based approaches measure entity similarities based on entity embeddings

- Three key modules
  - KG embedding
  - Alignment inference
  - How they interact



Instance Matching with Knowledge Graph Embedding

23 approaches (incomplete)

Relation embedding
- Triple
- Path
- Neighborhood subgraph

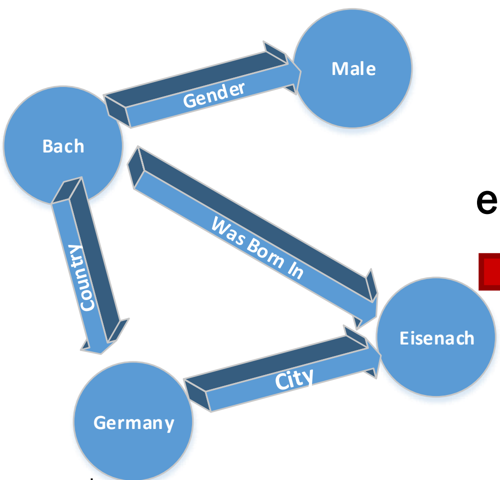
Attribute embedding
- Attribute correlation
- Literal

	Embedding		Alignment	Interaction	
	Relation	Att.	Emb. distance	Combination	Learning
MTransE [10]	Triple	-	Euclidean	Transformation	Superv.
IPTransE [93]	Path	-	Euclidean	Sharing	Semi-
JAPE [72]	Triple	Att.	Cosine	Sharing	Superv.
BootEA [73]	Triple	-	Cosine	Swapping	Semi-
KDCoE [9]	Triple	Literal	Euclidean	Transformation	Semi-
NTAM [44]	Triple	-	Cosine	Swapping	Superv.
GCNAlign [81]	Neighbor	Att.	Manhattan	Calibration	Superv.
AttrE [77]	Triple	Literal	Cosine	Sharing	Superv.
IMUSE [28]	Triple	Literal	Cosine	Sharing	Superv.
SEA [57]	Triple	-	Cosine	Transformation	Superv.
RSN4EA [24]	Path	-	Cosine	Sharing	Superv.
GMNN [85]	Neighbor	Literal	Cosine	Swapping	Superv.
MuGNN [8]	Neighbor	-	Manhattan	Calibration	Superv.
OTEA [58]	Triple	-	Euclidean	Transformation	Superv.
NAEA [94]	Neighbor	-	Cosine	Swapping	Superv.
AVR-GCN [88]	Neighbor	-	Euclidean	Swapping	Superv.
MultiKE [90]	Triple	Literal	Cosine	Swapping	Superv.
RDGCN [83]	Neighbor	Literal	Manhattan	Calibration	Superv.
KECG [42]	Neighbor	-	Euclidean	Calibration	Superv.
HGCN [84]	Neighbor	Literal	Euclidean	Calibration	Superv.
MMEA [68]	Triple	-	Cosine	Sharing	Superv.
HMAN [87]	Neighbor	Literal	Euclidean	Calibration	Superv.
AKE [47]	Triple	-	Euclidean	Transformation	Superv.

https://github.com/THU-KEG/Entity_Alignment_Papers

Instance Matching with Knowledge Graph Embedding (Example: MTransE)

Knowledge graph embedding: TransE



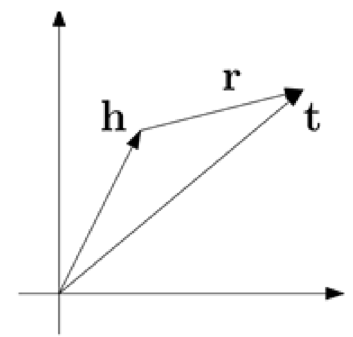
encode

Bach	(0.036, −0.12, ..., 0.323)
City	(0.102, 0.671, ..., −0.101)
Germany	(0.138, 0.551, ..., 0.222)
...	

Embeddings

TransE

- $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$

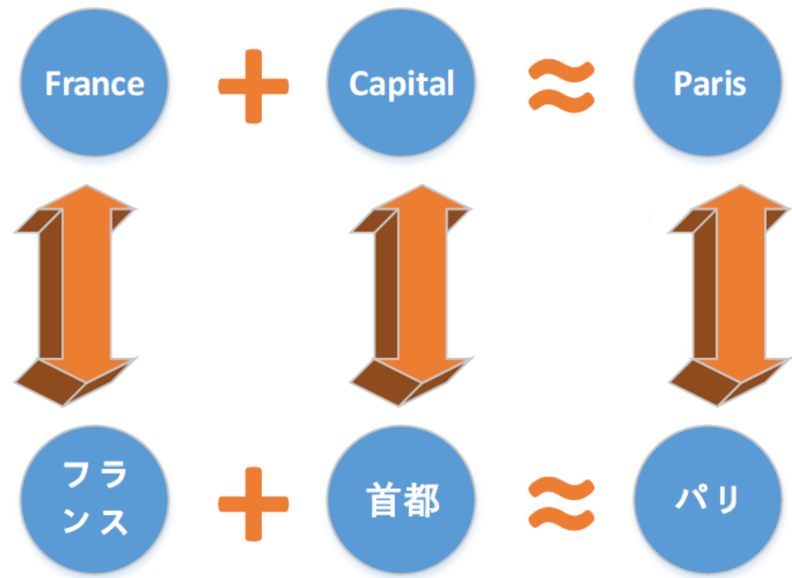


It is a monolingual scenario.

MTransE extends TransE in Multilingual Scenarios.

What does MTransE use and enable?

- Corpora: (partially-aligned) multilingual KGs
- Enabling: inferable embeddings of multilingual semantics
- Can be applied to:
  - Knowledge alignment
  - Cross-lingual Q&A
  - Multilingual chat-bots
  - ...



MTransE

MTransE Model Components

- *Knowledge model*

$$S_K = \sum_{L\in\{L_i,L_j\}} \sum_{T\in G_L} ||\mathbf{h} + \mathbf{r} - \mathbf{t}||$$

All triples in the knowledge graph in different languages

loss functions

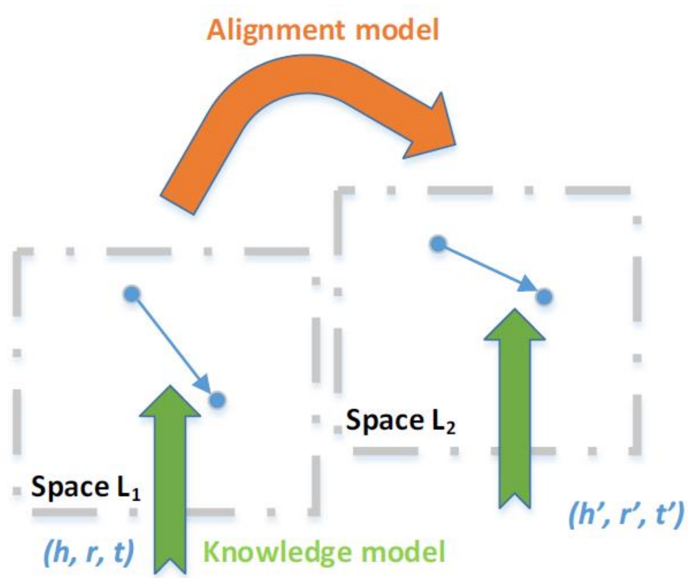
- *Alignment model*

$$S_A = \sum_{(T,T')\in\delta(L_i,L_j)} S_a(T,T')$$

All aligned triples

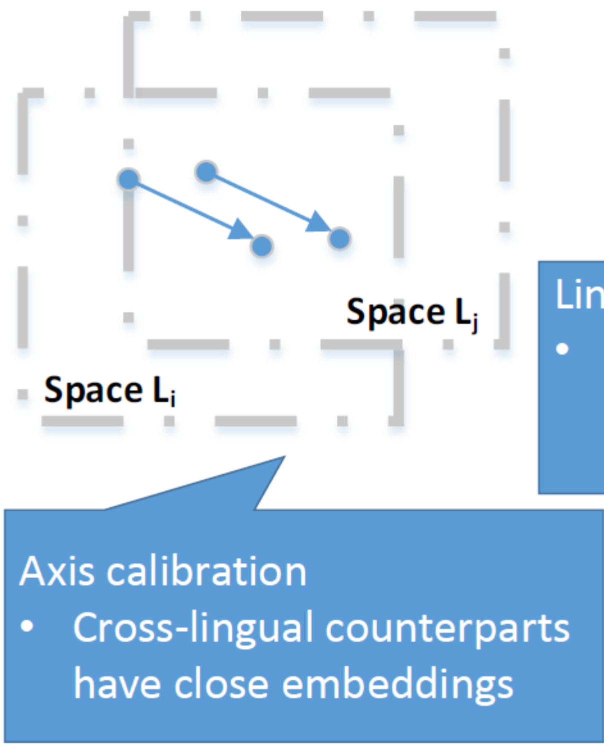
- *Objective of learning*
  - Minimizing $J(\theta) = S_K + \alpha S_A$

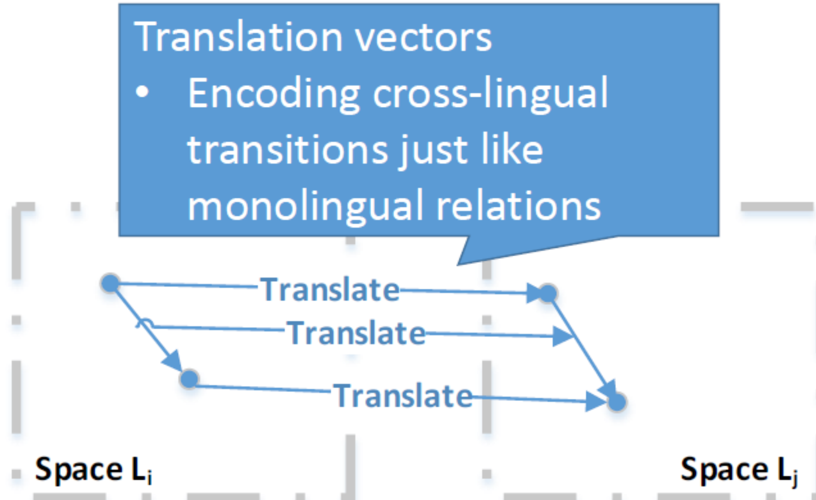
final loss function

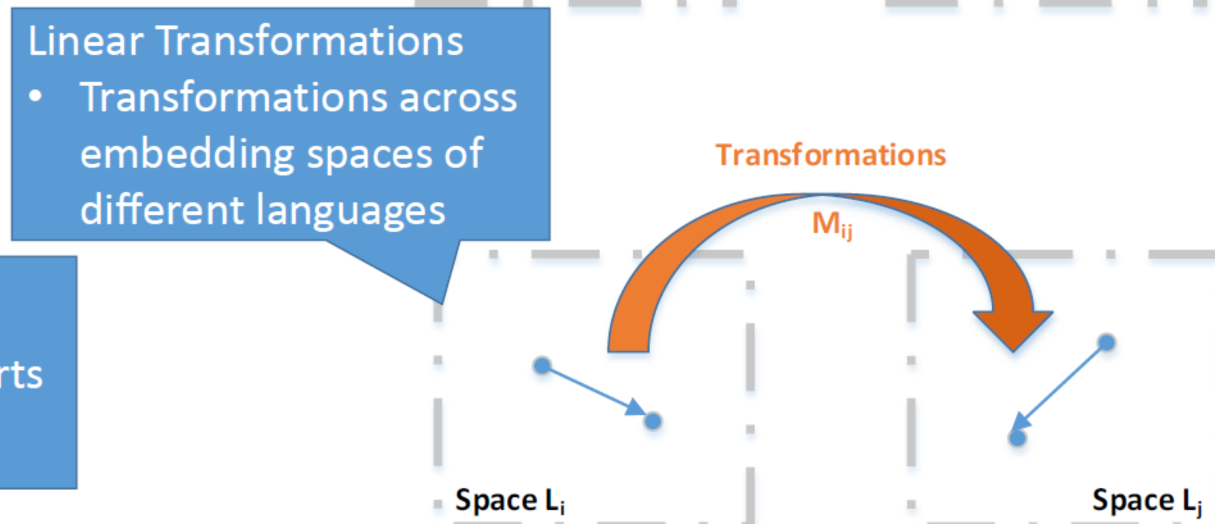


MTransE

Different alignment techniques







MTransE

Alignment Scores and Five Model Variants

- Var_i combines the i^{th} alignment model with the knowledge model

Variant	Alignment Score	Remark
Var_1	$S_{a_1} = \lVert \boldsymbol{h} - \boldsymbol{h}' \rVert + \lVert \boldsymbol{t} - \boldsymbol{t}' \rVert$	
Var_2	$S_{a_2} = \lVert \boldsymbol{h} - \boldsymbol{h}' \rVert + \lVert \boldsymbol{r} - \boldsymbol{r}' \rVert + \lVert \boldsymbol{t} - \boldsymbol{t}' \rVert$	
Var_3	$S_{a_3} = \lVert \boldsymbol{h} + \boldsymbol{v}^e_{ij} - \boldsymbol{h}' \rVert + \lVert \boldsymbol{r} + \boldsymbol{v}^r_{ij} - \boldsymbol{r}' \rVert + \lVert \boldsymbol{t} + \boldsymbol{v}^e_{ij} - \boldsymbol{t}' \rVert$	$\boldsymbol{v}^e_{ij} = -\boldsymbol{v}^e_{ji},\ \boldsymbol{v}^r_{ij} = -\boldsymbol{v}^r_{ji}$
Var_4	$S_{a_4} = \lVert \boldsymbol{M}^e_{ij}\boldsymbol{h} - \boldsymbol{h}' \rVert + \lVert \boldsymbol{M}^e_{ij}\boldsymbol{t} - \boldsymbol{t}' \rVert$	$\boldsymbol{M}^e_{ij} \in \mathbb{R}^{k \times k},\ \boldsymbol{M}^r_{ij} \in \mathbb{R}^{k \times k}$
Var_5	$S_{a_5} = \lVert \boldsymbol{M}^e_{ij}\boldsymbol{h} - \boldsymbol{h}' \rVert + \lVert \boldsymbol{M}^r_{ij}\boldsymbol{r} - \boldsymbol{r}' \rVert + \lVert \boldsymbol{M}^e_{ij}\boldsymbol{t} - \boldsymbol{t}' \rVert$	

Axis Calibration

Translation Vector

Linear Transforms

Instance Matching with Rules

- The Same Entity in Multiple Sources

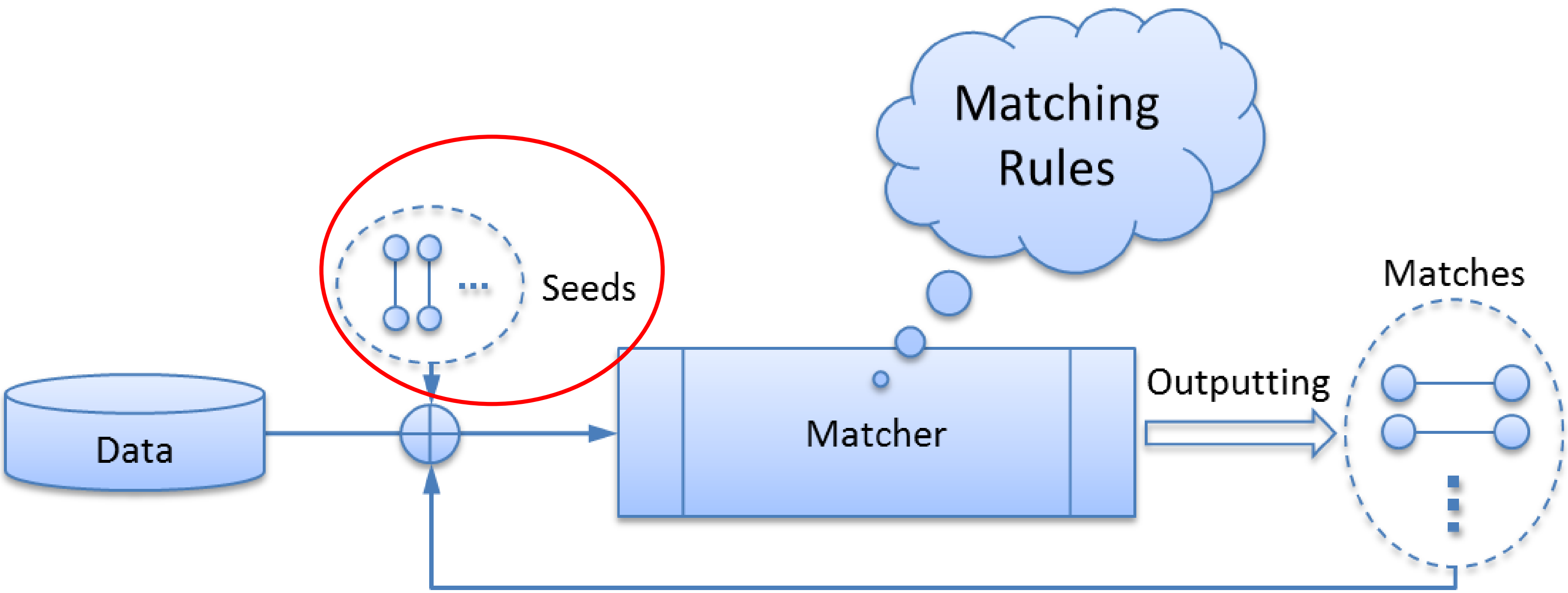
Baidu Baike

baidu:大熊猫	
baidu:标签	“大熊猫”
baidu:拉丁学名	“Ailuropoda melanoleuca”
baidu:纲	baidu:哺乳纲
…	

Hudong Baike

hudong:大熊猫	
hudong:中文学名	“大熊猫”
hudong:二名法	“Ailuropoda melanoleuca”
hudong:纲	hudong:哺乳纲
…	

Instance Matching with Rules



- **Automatically discovering and refining dataset-specific matching rules in iterations**
  - Deriving these rules by finding the most discriminative data characteristics for a given data source pair.

Instance Matching with Rules

Seeds - Lightweight Entity Matching

Punctuation Cleaning:
Space Shuttle Endeavour ≈ Space Shuttle “Endeavour”

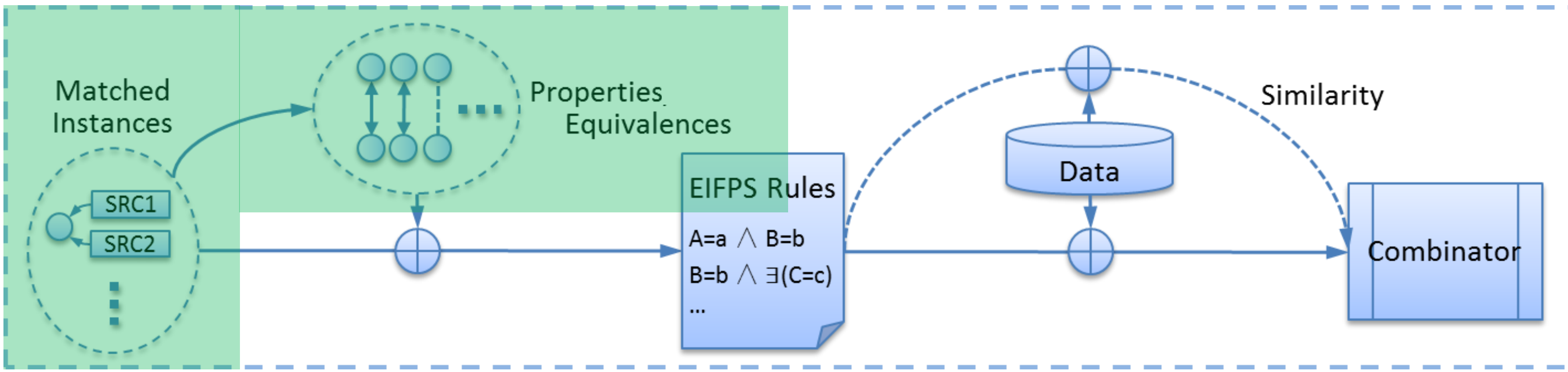
Redirects Information:
A redirects to B means A and B are synonyms

…

肖申克的救赎 = 《肖申克的救赎》
海尔波普彗星 = 海尔·波普彗星 = 海尔-波普彗星
奋进号航天飞机 = “奋进号” 航天飞机

Instance Matching with Rules

- Mining Properties Equivalences

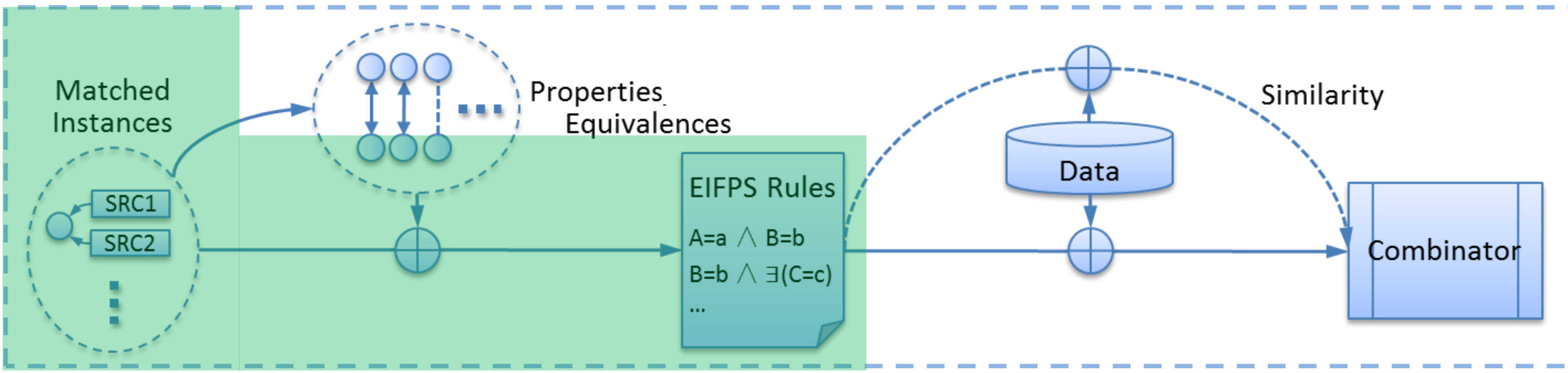


- For each pair of existing matched instances, their property-value pairs are merged.

Values	Property_1	Property_2
"大熊猫"	baidu:标签	hudong:中文学名
"Ailuropoda melanoleuca"	baidu:拉丁学名	hudong:二名法
"白鳍豚"	baidu:标签	hudong:中文学名
"桂花"	baidu:标签	hudong:中文学名
…	…	…

Instance Matching with Rules

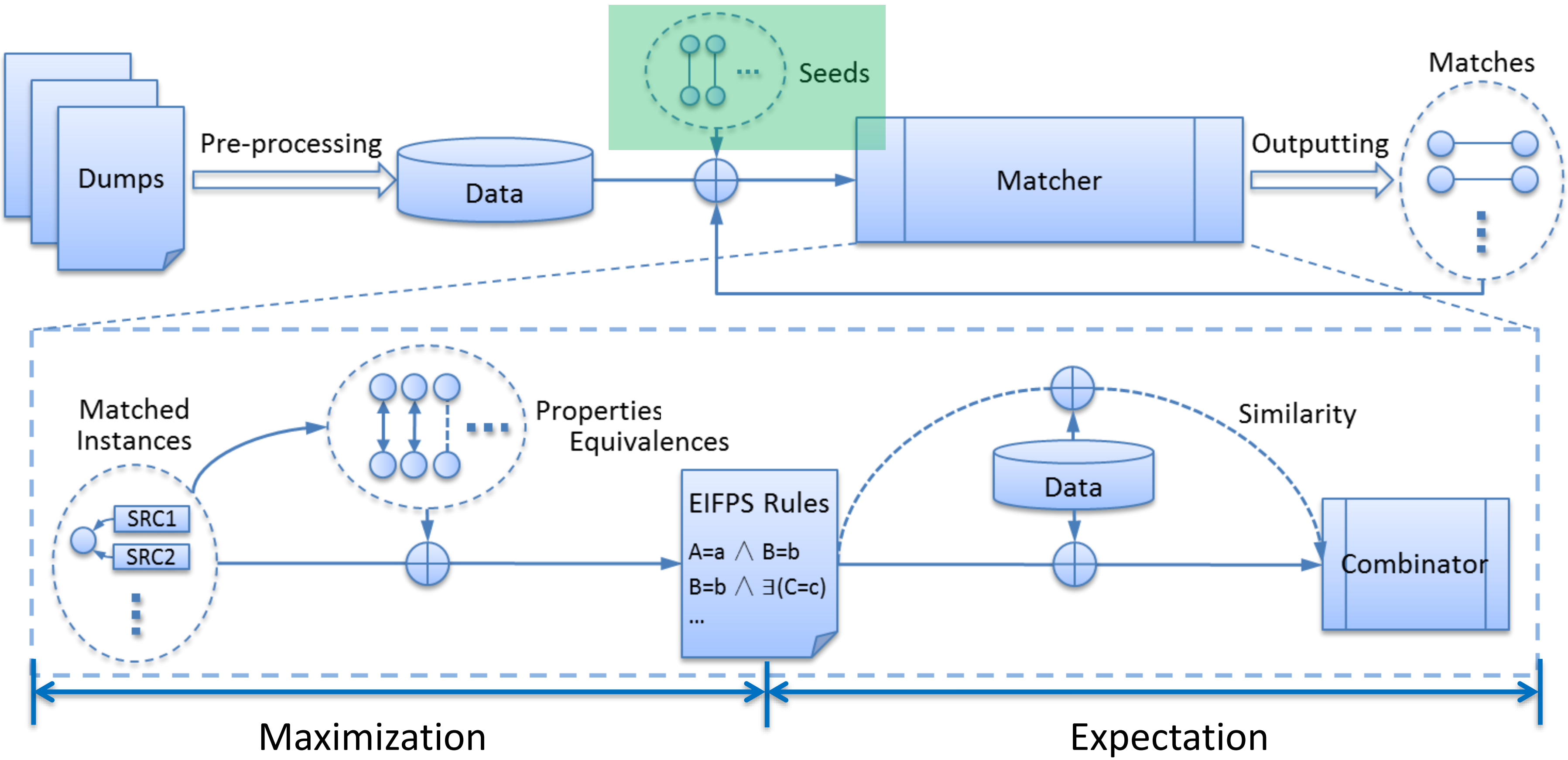
- Mining Matching Rules



- Matching rule (frequent set mining):
  - baidu:x and hudong:x are matched, iff.
  - valueOf(baidu:标签) = valueOf(hudong:中文学名)
  - and
  - valueOf(baidu:拉丁学名) = valueOf(hudong:二名法)
  - and
  - valueOf(baidu:纲) = valueOf(hudong:纲)

Instance Matching with Rules

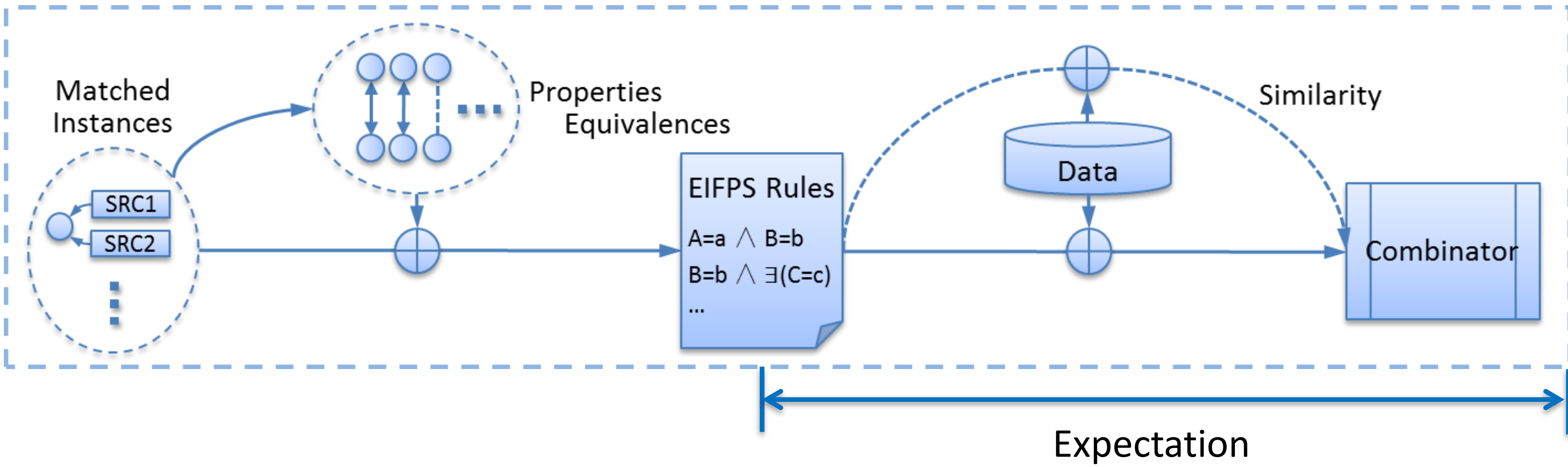
- The Wrapper Algorithm



- The wrapper is an implementation of Expectation-Maximization iterations.

Instance Matching with Rules

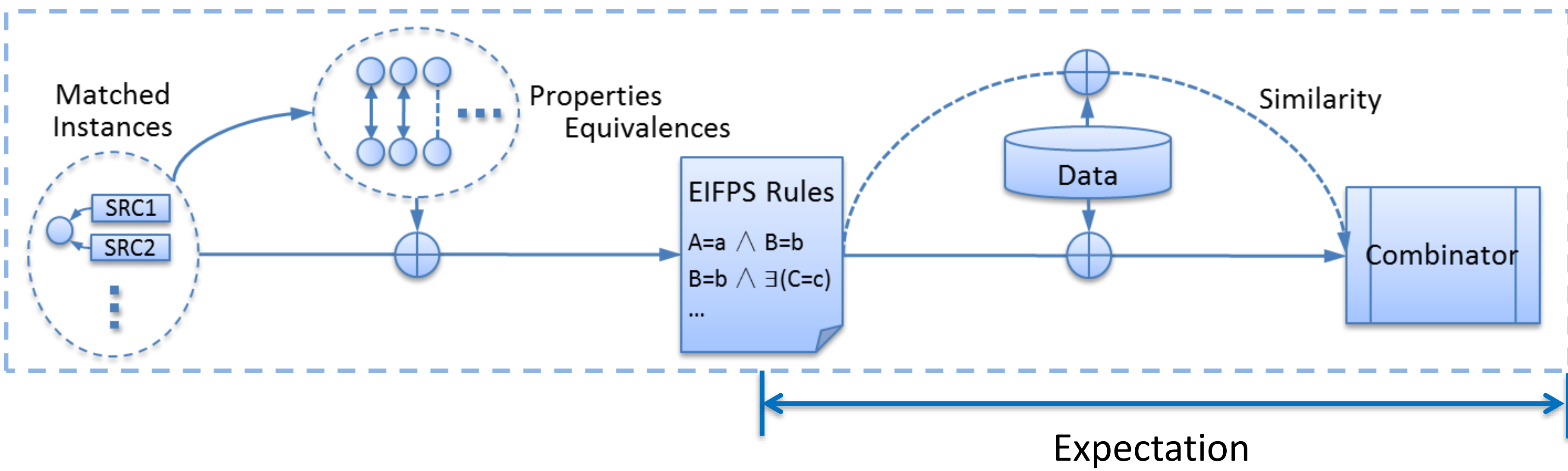
- The E-step



- The E-step estimates the missing data (matches) using the observed data and the current estimate for the parameters (matching rules).

Instance Matching with Rules

- The M-step



- The M-step computes parameters maximizing the likelihood function as the data estimated in E-step are used in lieu of the actual missing data.
  - M: matches
  - Θ: parameters

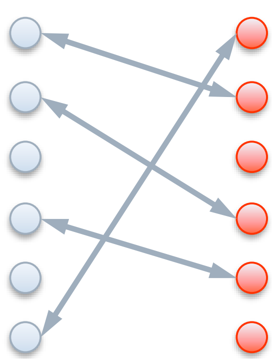
$$L(\boldsymbol{\theta}; \boldsymbol{M}) = \Pr(\boldsymbol{M} | \boldsymbol{\theta}).$$

Instance Matching with Rules

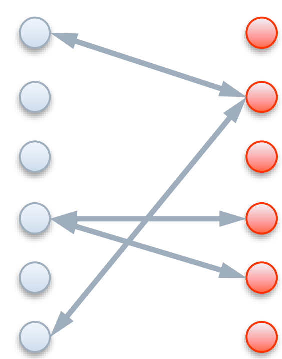
- The Likelihood Function

$$L(\boldsymbol{\theta}; \boldsymbol{M}) \approx \frac{|\mathrm{ConnectedComponent}(M)|}{|\mathrm{Edge}(M)|}$$

- Assuming that no equivalent instances exist in a single data source, we can infer that an instance is equivalent to at most one another from the other data source.
- Incorrect matches in M may result in a node connecting to more than one other node, which is contrary to the assumption.



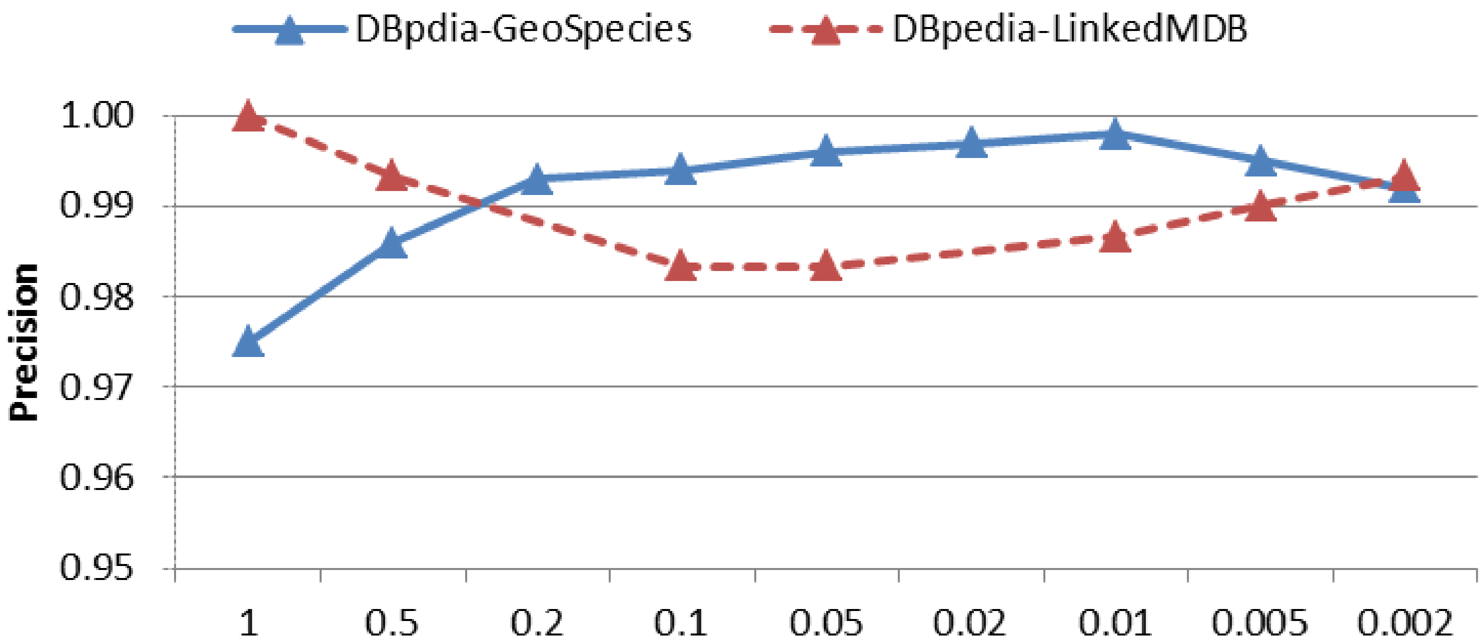
P=4/4=1



P=2/4=0.5

Instance Matching with Rules

- Precisions



- Sampling a certain number of output matches.
- The X-axis indicates the proportions of selected seeds in complete reference matches.

Question

What are the advantages and disadvantages of the embedding-based method and rule-based method for instance matching?

Thanks!