Semantic data mining for knowledge acquisition

Tutorial (K-CAP 2017)
Part II

Agnieszka Lawrynowicz
Poznan University of Technology

Planned schedule

- 14:00 – 15:30 Introduction, Semantic Data Mining Tasks and Pecularities
  - Introduction (20 min.)
  - Basics of semantic data mining (20 min.)
  - Tasks of semantic data mining (20 min.)
  - Pecularities of semantic data mining (30 min.)
- 15:30 – 16:00 Coffee break
- 16:00 – 18:00 Semantic Data Mining for Knowledge Acquisition, Hands-On
  - Semantic data mining for knowledge acquisition (60 min.)
  - Hands-on: LeoLOD Swift Linked Data Miner plugin for Protégé (60 min.)

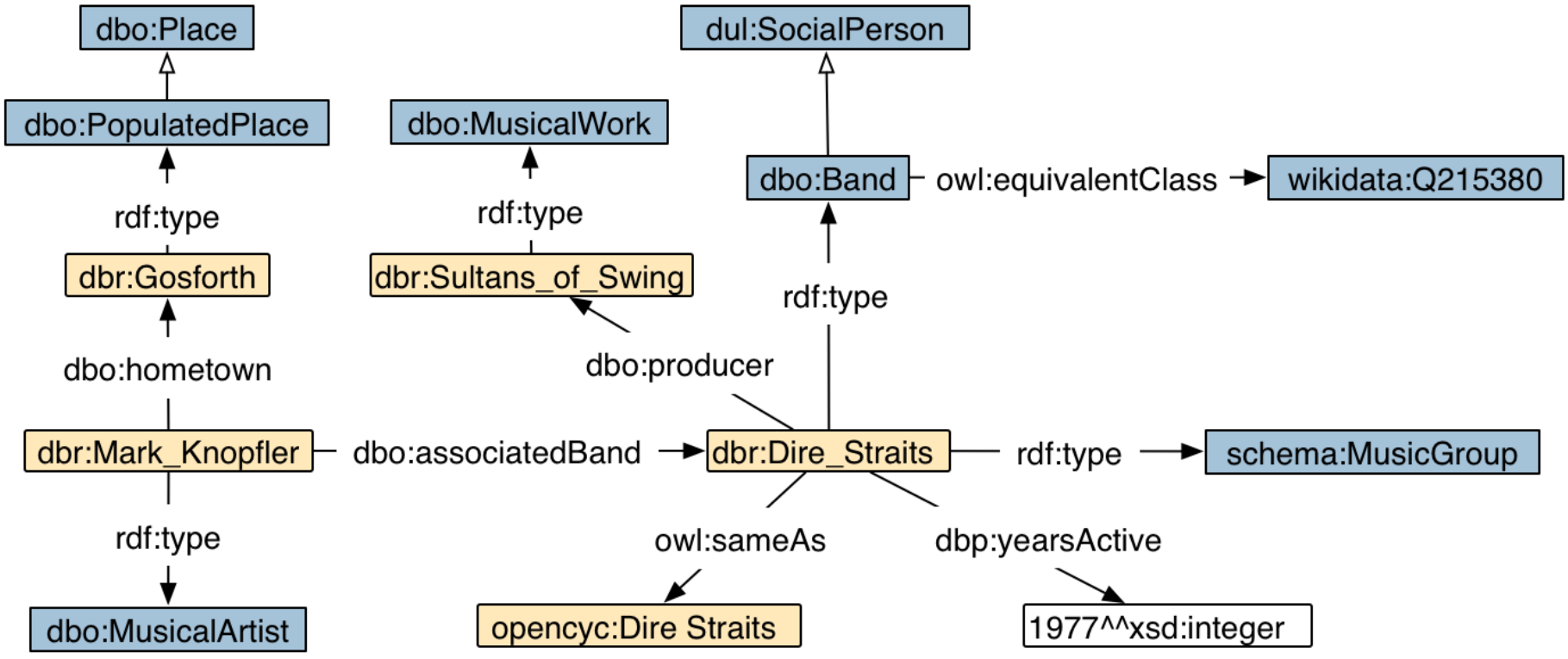
Knowledge graph

Definition

Knowledge graphs (KGs) are large, graph-structured knowledge bases which represent facts in the form of relationships between entities.

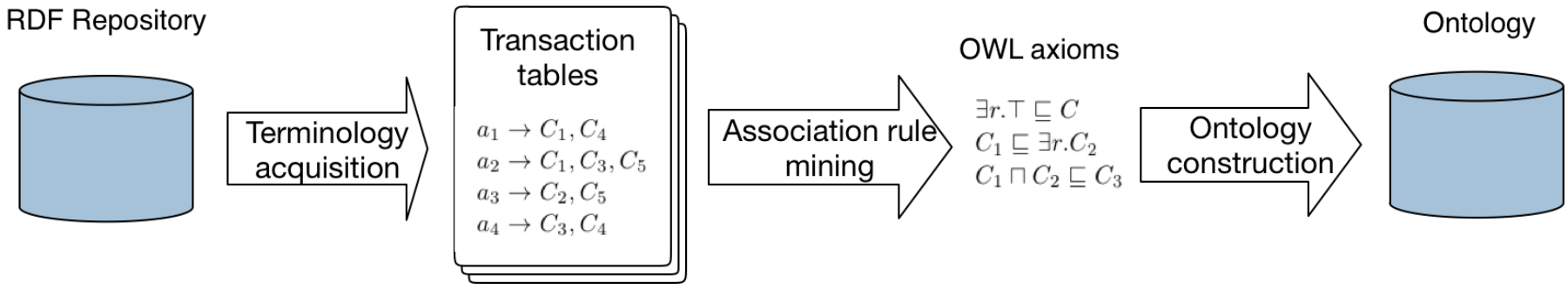
- The basic building blocks of a knowledge graph are:
  - entities, expressed via the nodes in a graph,
  - their properties (attributes), and relations connecting the nodes, expressed via the edges in the graph
- the entities may be (semantically) typed what is represented by the is-a relation between an entity and its type.
- some types, properties and relations smay be stored in a knowledge graph in a structured form of an ontology or a schema (TBox)
- knowledge graphs are concentrated on instance data (ABox) and the number of instances in a typical knowledge graph is huge

Knowledge graph: example



Statistical schema induction

Völker, Niepert (2011)



Statistical schema induction: Terminology acquisition

Völker, Niepert (2011)

- Select candidates for atomic classes, properties, individuals
- Assign identifiers to class and property expressions

```
SELECT DISTINCT ?x WHERE{
?y rdf:type ?x
}
```

```
SELECT DISTINCT ?x WHERE{
?y ?x ?z .
?z rdf:type ?zt
}
```

Statistical schema induction: Association rule mining

Völker, Niepert (2011)

1146	6330	3961	64		
3235	3788	2	66	64	
3235	3788	2	66	364	17
1146	6330	64			
1146	3788	64			

64 = Person
1146 = $\exists$ birthPlace.PopulatedPlace

$\{1146\} \rightarrow \{64\}$

$\exists$ birthPlace.PopulatedPlace $\sqsubseteq$ Person

Statistical schema induction: Ontology construction

Völker, Niepert (2011)

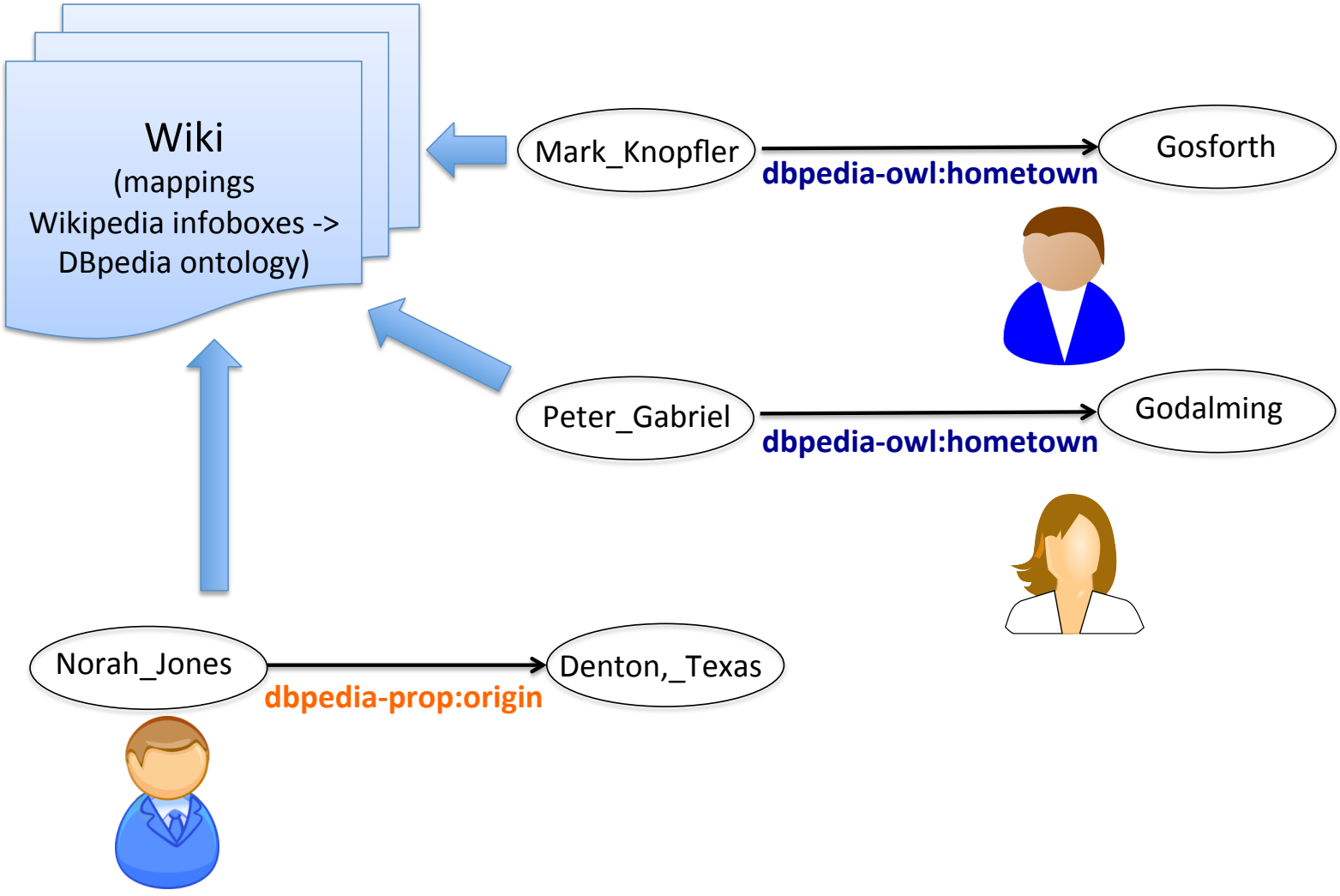
Axiom Type	Association Rule
$C \sqsubseteq D$	$\{C_i\} \rightarrow \{C_j\}$
$C \sqcap D \sqsubseteq E$	$\{C_i, C_j\} \rightarrow \{C_k\}$
$D \sqsubseteq \exists r.C$	$\{C_i\} \rightarrow \{\exists r_j.C_{jk}\}$
$\exists r.C \sqsubseteq D$	$\{\exists r_j.C_{jk}\} \rightarrow \{C_i\}$
$\exists r.\top \sqsubseteq C$	$\{\exists r_j.\top\} \rightarrow \{C_i\}$
$\exists r^{-1}.\top \sqsubseteq C$	$\{\exists r_j^{-1}.\top\} \rightarrow \{C_i\}$
$r \sqsubseteq s$	$\{r_i\} \rightarrow \{r_j\}$

Learning class disjointness

Völker, Fleischhacker, Stuckenschmidt (OTM 2011, JWS 2015)

- negative association rules $A \rightarrow \neg B$
- additional concepts that represent the complements of each concept included into transaction tables
- if the corresponding SPARQL query detects an instance not belonging to a concept A the membership in the complement concept ($\neg A$) is recorded in the transaction table

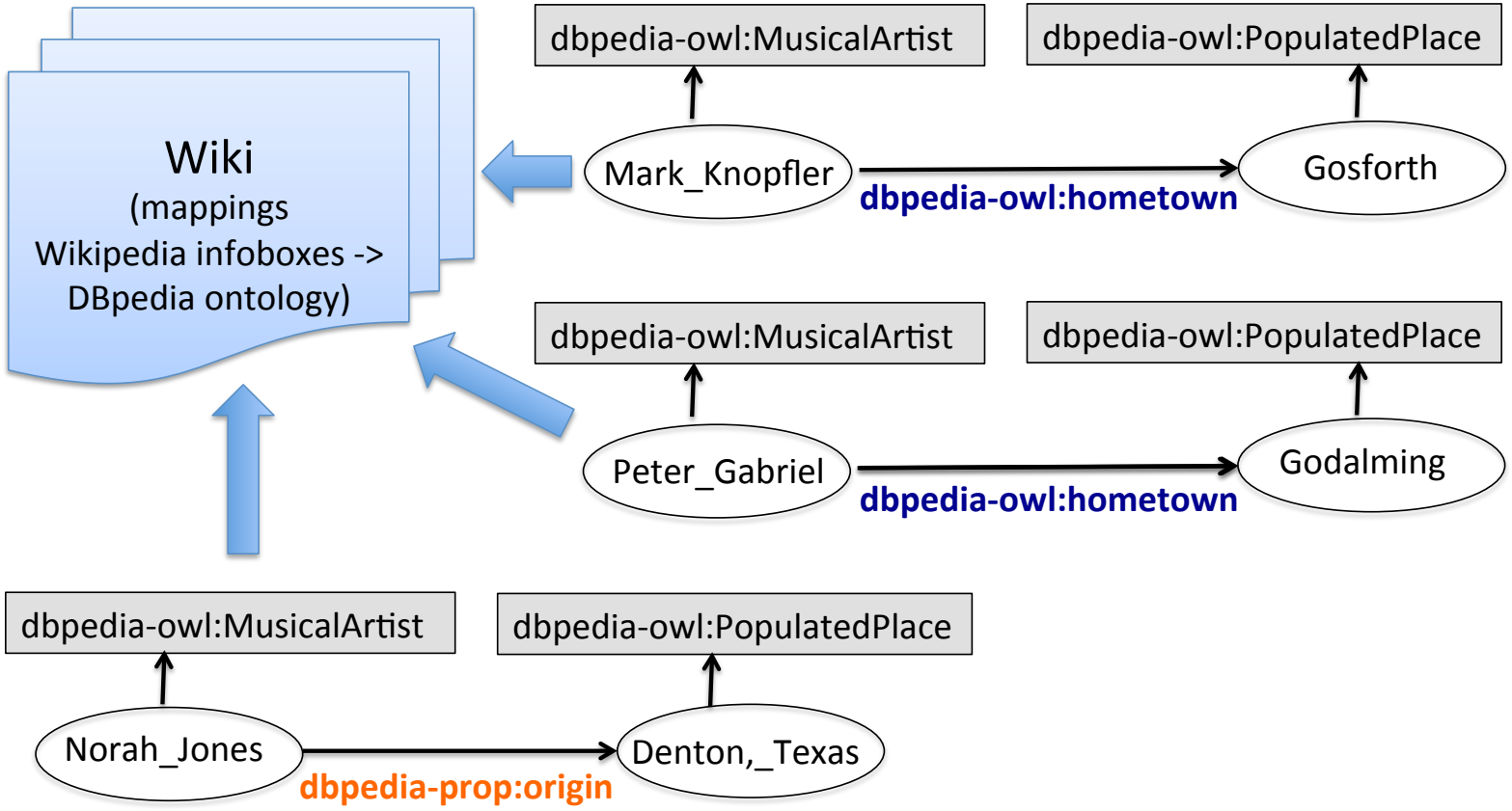
Mining Substitutive Properties: Motivating scenario 1/3



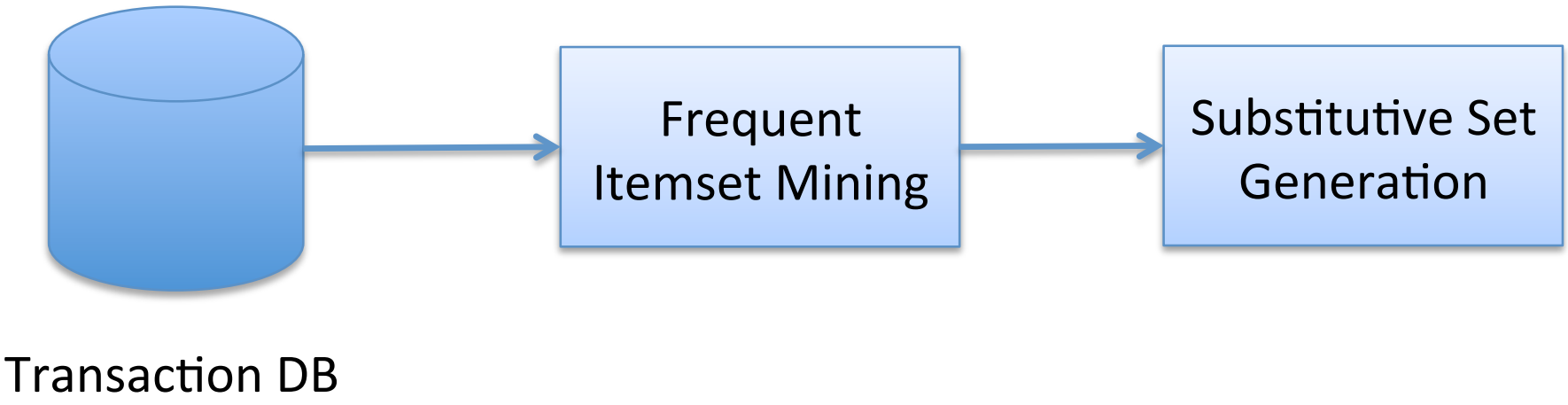
Mining Substitutive Properties: Motivating scenario 2/3

- DBpedia 2014 ontology has 1310 object and 1725 data properties
- Many large Linked Data use relatively lightweight schemas with a high number of object properties

Mining Substitutive Properties: Motivating scenario 3/3



Mining Substitutive Properties: Substitutive Sets Mining Framework



Mining Substitutive Properties: Frequent Itemsets

- $I = \{i_1, i_2, \ldots, i_m\}$ - a set of items
- $D_T = \{t_1, t_2, \ldots, t_n\}$, where $\forall i \; t_i \subseteq I$ -a database of transactions

- $support(X) = \frac{|\{t \in D_T : X \subseteq t\}|}{|D_T|}$

ID	Items
1	Nachos, Pepsi, Salsa
2	Nachos, Coca-Cola, Salsa
3	Nachos, Coca-Cola
4	Nachos, Pepsi, Salsa
5	Milk, Bread

Frequent Itemset	Support
{Nachos}	80%
{Salsa}	60%
{Coca-Cola}	40%
{Pepsi}	40%
{Nachos, Salsa}	60%
{Nachos, Coca-Cola}	40%
{Nachos, Pepsi}	40%
{Salsa, Pepsi}	40%

Mining Substitutive Properties: Covering Set

- $CS(i|L) = \{X \in L : \{i\} \cup X \in L\}$
- $coverage(i|L) = |CS(i|L)|$

Frequent Itemset
{Nachos}
{Salsa}
{Coca-Cola}
{Pepsi}
{Nachos, Salsa}
{Nachos, Coca-Cola}
{Nachos, Pepsi}
{Salsa, Pepsi}

i	CS(i)	coverage
{Nachos}	{{Salsa}, {Coca-Cola}, {Pepsi}}	3
{Salsa}	{{Nachos}}	1
{Coca-Cola}	{{Nachos}}	1
{Pepsi}	{{Nachos}, {Salsa}}	2

Mining Substitutive Properties: Substitutive Sets

A two-element itemset $\{x, y\}$ is a *substitutive itemset*, if:

- $x \in L_1$,
- $y \in L_1$,
- $support(\{x\} \cup \{y\}) < \varepsilon$, where ε is a user-defined threshold representing the highest amount of noise in the data allowed,
- $\frac{|CS(x|L) \cap CS(y|L)|}{\max\{|CS|L(x)|, |CS(y|L)|\}} \geqslant mincommon$.

i	CS(i)	coverage
{Nachos}	{{Salsa}, {Coca-Cola}, {Pepsi}}	3
{Salsa}	{{Nachos}}	1
{Coca-Cola}	{{Nachos}}	1
{Pepsi}	{{Nachos}, {Salsa}}	2

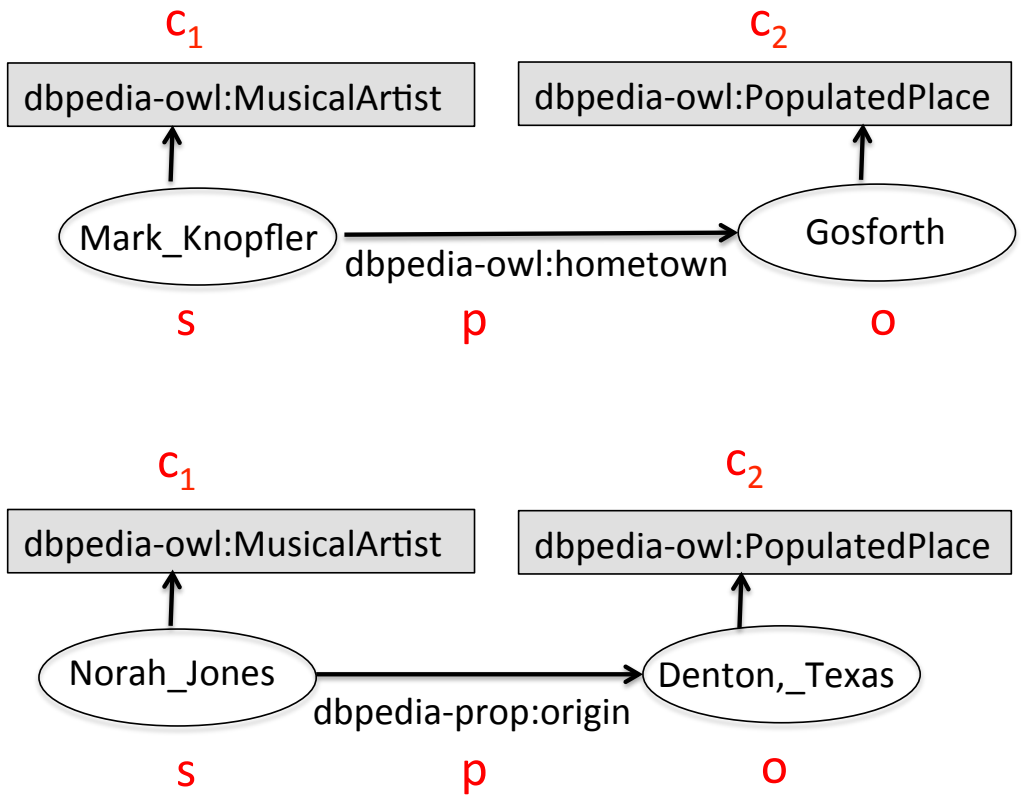
$$\frac{|CS(Pepsi) \cap CS(Coca-Cola)|}{\max\{|CS(Pepsi)|, |CS(Coca-Cola)|\}} = 0.5$$

Mining Substitutive Properties: Use Case - DBpedia

- DBpedia knowledge base version 2014
- sets of 3–item transactions $\{c_1, p, c_2\}$, where c_1 and c_2 classes of subject and object of RDF triple, and p property connecting s and o

```
SELECT ?c1 ?p ?c2
WHERE {
?s rdf:type dbpedia-owl:Organization .
?s ?p ?o .
?s rdf:type ?c1 .
?o rdf:type ?c2 .
FILTER(?p != dbpedia-owl:wikiPageWikiLink) .
FILTER(?p != rdf:type) .
FILTER(?p != dbpedia-owl:wikiPageExternalLink) .
FILTER(?p != dbpedia-owl:wikiPageID) .
FILTER(?p != dbpedia-owl:wikiPageInterLanguageLink) .
FILTER(?p != dbpedia-owl:wikiPageLength) .
FILTER(?p != dbpedia-owl:wikiPageOutDegree) .
FILTER(?p != dbpedia-owl:wikiPageRedirects) .
FILTER(?p != dbpedia-owl:wikiPageRevisionID)
}
```

Mining Substitutive Properties: Transaction generation



Transactions
{c1_dbpedia-owl:MusicalArtist, dbpedia-owl:hometown, c2_dbpedia-owl:PopulatedPlace }
{c1_dbpedia-owl:MusicalArtist, dbpedia-prop:origin , c2_dbpedia-owl:PopulatedPlace }
...

Mining Substitutive Properties: Sample substitutive properties for the class Organisation

Item X	Item Y	Common
dbpprop:parentOrganization	dbo:parentOrganisation	1.000
dbpprop:owner	dbo:owner	1.000
dbpprop:origin	dbo:hometown	1.000
dbpprop:headquarters	dbpprop:parentOrganization	1.000
dbpprop:formerAffiliations	dbo:formerBroadcastNetwork	1.000
dbo:product	dbpprop:products	1.000
dbpprop:keyPeople	dbo:keyPerson	0.910
dbpprop:commandStructure	dbpprop:branch	0.857
dbo:schoolPatron	dbo:foundedBy	0.835
dbpprop:notableCommanders	dbo:notableCommander	0.824
dbo:recordLabel	dbpprop:label	0.803
dbo:headquarter	dbo:locationCountry	0.803
dbpprop:country	dbo:state	0.753

Learning Types from RDF Data

SDType (Paulheim ISWC2013)

- Basic idea: incoming/outgoing properties as indicators for a resource type, e.g. starring$\rightarrow$ Movie
- Basic compiled statistics:
  - P(C|p) = probability of class C in the presence of property p
  - e.g.: P(dbpedia:film|starring)=0.79

Learning Types from RDF Data

SDType (Paulheim ISWC2013)

- Using compiled statistics:
  - find instance types
  - use voting
- score (c): avg(all properties p) P(C|p)
- refine with weights for properties

Embeddings

- Embeddings are modeling and feature learning techniques where words, phrases, entities or concepts from some vocabulary are mapped to dense vectors of real numbers.
- It consists of mathematical embedding from a space with one dimension per word/phrase/entity/concept to a low-dimensional and continuous vector space where co-occurring words/phrases/entities/concepts are located close to each other.
- Methods to generate such mapping: neural networks, dimensionality reduction on the word/phrase/entity/concept co-occurrence matrix, probabilistic models, others.

Vector space models

Represent an entity as a vector of numbers

banana	1	0	0	0	2	0	1	0	1	0	0	0

World analogy

man is to *woman* as *king* is to ___?
USA is to *Washington* as *Poland* is to ___?

[king]-[man]+[woman]≈[queen]

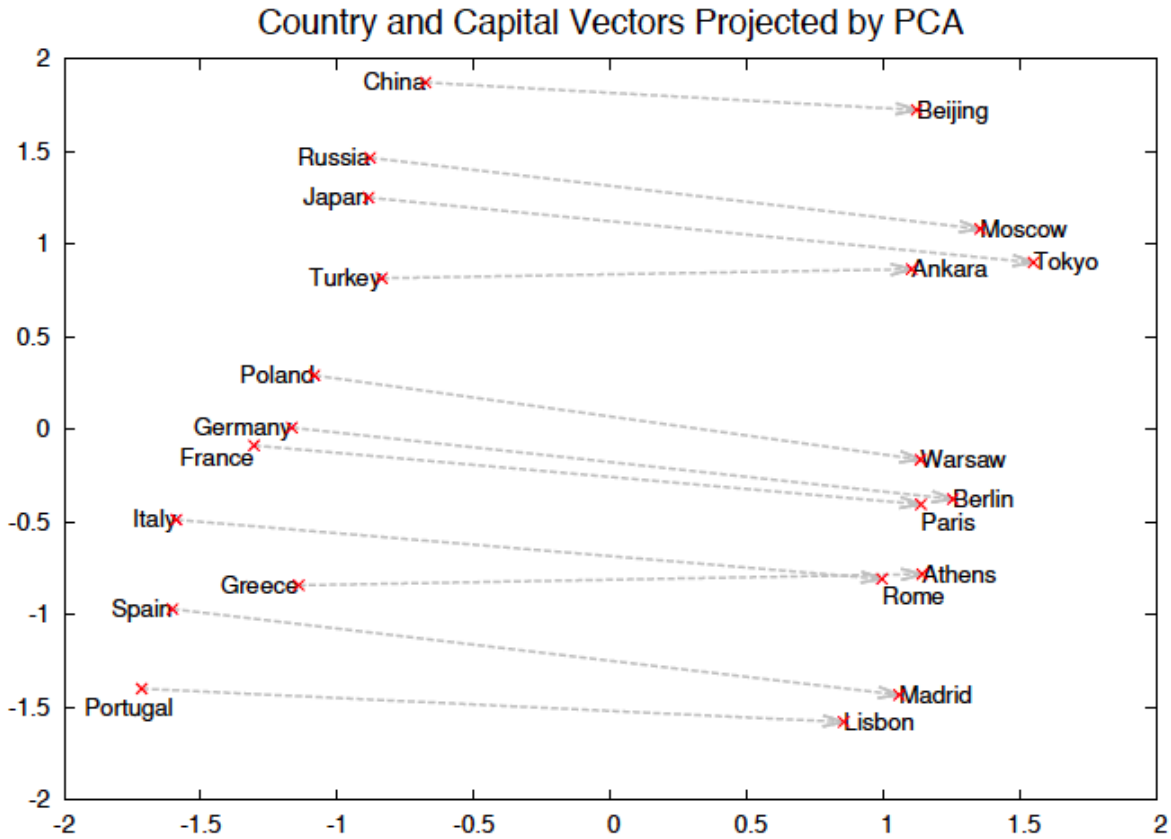
Learning dense embeddings: Matrix factorization

Factorizing word-context matrix, e.g. GloVe (Pennington et al., EMNLP 2014)

	Context 1	Context 2	…	Context k
Word 1				
Word 2				
…				
Word n				

Learning dense embeddings: Neural networks

Shallow, two-layer neural networks, trained to reconstruct contexts of words with word and context as both input and output, e.g. Word2vec (Mikolov et al. NIPS 2013)



Knowledge graph embeddings

- represent entities or concepts of the graphs as vectors
- relations among entities or concepts are represented with various forms of vector calculation bound with specific relational semantics
- two major families: translation-based and non-translation-based embeddings

Translation-based Models

- Triple represented as: (h, r, t):
  - h: head entity
  - r: relation
  - t: tail entity
- h and t represented as two k-dimensional vectors **h** and **t**
- score function f_r(**h**, **t**) used to measure the plausibility of (h, r, t), involves the transformation **r** characterizing r

Construct embeddings by treating relation r as translation from head entity h to tail entity t

Translation-based Models

TransE (Bordes et al., NIPS 2013)

- 1-1 relations of entities
- predicting a missing item in a triple or verifying the validity of a generated triple

Translation-based Models

TransH (Wang et al., AAAI 2014)

- projections on relation-specific hyperplanes
- knowledge graph completion and link prediction

Translation-based Models

TransR (Lin et al., AAAI 2015)

- linear transformations to heterogeneous relation spaces
- knowledge graph completion and link prediction

Non-Translation-Based Models

- NTN (Socher et al. NIPS 2013): reasoning with neural networks to learn structured data for knowledge base completion
- TADW (text-associated DeepWalk) (Yang et al. IJCAI 2015): using random walk on graphs to incorporate text features of vertices into network representation learning

RDF2Vec: RDF Graph Embeddings

Ristoski & Paulheim (ISWC 2016)

- adaptation of neural language models (Word2vec)
- converting RDF graphs in sequences of entities and relations to form sentences (graph walks, graph kernels)
- training neural language model

Graph Walks: RDF2vec

Ristoski & Paulheim (ISWC 2016)
For each entity in the graph:

- extract a subgraph with depth d
- extract walks on the subgraph
- build word2vec model

SLDM algorithm

Swift Linked Data Miner (Potoniec, Lawrynowicz et. al, JWS 46, 2017)

- discovers new partial definitions, i.e. SubClassOf axioms, for a given class
- an anytime algorithm: it delivers patterns once they are mined, then refines them
  - the longer the algorithm works, the more complex patterns are mined
- does not require an access to the whole RDF graph at the same time
  - it downloads on-demand necessary parts by querying the SPARQL endpoint with very simple queries, consisting only of a single triple pattern and a VALUES clause

SLDM - how it works

Swift Linked Data Miner (Potoniec, Lawrynowicz et. al, JWS 46, 2017)

- SLDM constructs a set of URIs belonging to the selected class, by posing a SPARQL query, e.g.: `?x rdf:type Astronaut`
- Then operates in two alternating phases:
  1. querying the endpoint, about all the triples having an URI from the set in a subject position, by using WHERE clause:
     `?s ?p ?o . VALUES ?s {«URIs»}.`
     - the triples are then organized into a three-level index, with predicates in the first level, objects in the second and subjects in the third
  2. mining the obtained triples by scanning the index to discover the axioms, e.g. if for predicate rdf:type and object Person there are many subjects in the third level, an axiom Astronaut subClassOf Person is mined.
     - if, for a given predicate, no pattern can be found, the corresponding subjects are used as an input to the first phase of SLDM, to mine more complex axioms

SLDM - three-level index

Swift Linked Data Miner (Potoniec, Lawrynowicz et. al, JWS 46, 2017)

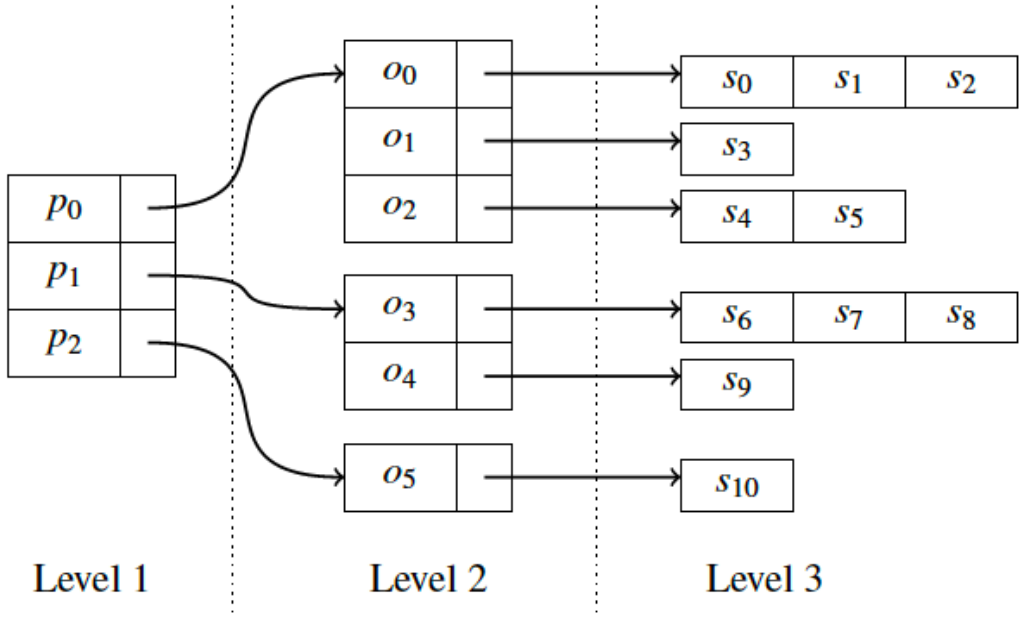
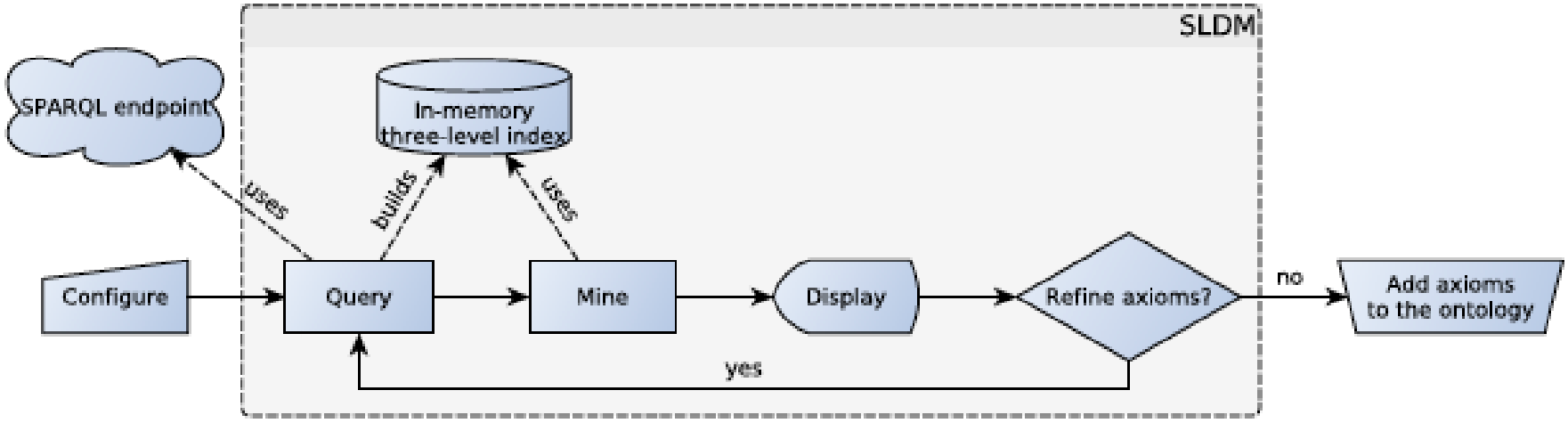


Figure: A sample three level index used in SLDM. The first level consists of predicates, the second level of objects and the third of subjects. Empty rectangles denote pointers to the next level. There are 11 triples in this index: 6 with predicate p_0, 4 with predicate p_1 and 1 with predicate p_2, namely (s_{10}, p_2, o_5).

A typical workflow with the SLDM plugin



Acknowledgements

- Some presentation ideas inspired on/borrowed from: Nada Lavrac, Claudia d'Amato, Nicola Fanizzi, Jens Lehmann, Simon Razniewski, Petar Ristoski, Heiko Paulheim, Johanna Voelker, and Jedrzej Potoniec
- Polish National Science Center (Grant No 2014/13/D/ST6/02076), grant entitled "ARISTOTELES: Methodology and algorithms for automatic revision of ontologies in task based scenarios" (2015-2018)