Part 3: Knowledge Modeling – Graph Databases

Overview

1. Taxonomy Induction
2. Schema Mapping

1. TAXONOMY INDUCTION

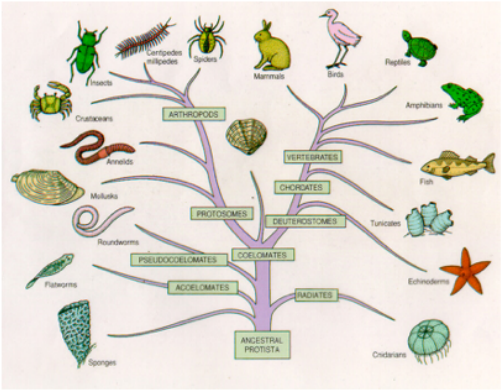
1. Taxonomy Induction

Information extraction

- Extract isolated facts from documents, e.g., *lion ISA animal*

Taxonomy induction

- Extract related facts from documents, e.g., classification of animals



Information extraction concerns the extraction of isolated facts, such as ISA relationships. Taxonomy induction aims at extracting related facts and organizing them in a structured knowledge basis, e.g. a hierarchical taxonomy. It is a special case of the more general ontology induction, which organizes knowledge using arbitrary relationships.

Use of Taxonomies

Hyponyms (subordinate terms) can inherit properties from hypernyms (more general terms)

- Due to transitivity of ISA, no need to learn inferred facts

No unique taxonomies

- Depending on the perspective and application different taxonomies may be useful:

A tiger and a puppy are both Mammals and hence belong close together in a typical taxonomy, but tiger is aWildAnimal (in the perspective of Animal-Function) and a JungleDweller (in the perspective of Habitat), while a puppy is a Pet (as function) and a HouseAnimal (as habitat), which would place them relatively far from one another

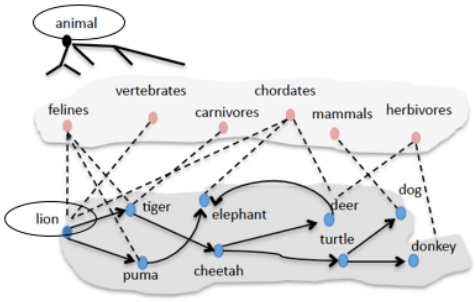
One of the advantages of taxonomy induction (and more generally ontology induction) is the possibility to perform inferences on the extracted knowledge. For example, in a ISA hierarchy, lower nodes in the hierarchy can inherit properties from higher nodes, thus these extra facts need not to be learnt separately.

One of the challenges in taxonomy induction is the fact that there is no notion of “correct” taxonomy. A taxonomy strongly depends in its intended use and on the specific perspective of the user on the domain. Integrating all possible perspectives into one single global taxonomy would not be feasible and useful, as the resulting taxonomy would potentially be too complex to be used.

Taxonomy Induction Task

Starting from a root concept and a basic concept

1. Learn relevant terms and their hypernym / hyponym relationships
2. Filter out erroneous terms and relations
3. Induce a taxonomy structure



We present in the following one specific approach to taxonomy induction, which was one of the first approaches proposed in the field. It starts from the assumption that one general concept (e.g. animals) and at least on basic concept of the taxonomy (e.g. lion) are provided as input. From this starting point the task is to identify more relevant concepts (represented as terms), establish the hypernym and hyponym relationships, filter out erronous terms and relations and finally induce an overall taxonomy structure. The figure illustrates the process: at the bottom level first more related terms are identified. Then intermediate concepts, more abstract than the basic concepts are identified. From this data finally the taxonomy will be induced.

root: top most of graph (animal)
basic: one example (lion)

Learning Terms

Template approach

- Given a **root concept c** (e.g. animal) and a **seed s** (e.g. lion)
- Hyponym pattern: $P_i(c, s, X)$ = c such as s and X
- Hypernym pattern: $P_c(t_1, t_2, X)$ = X such as t_1 and t_2

X is the new found hyponym

X is the new found hypernym

given ANIMAL and LION
- find hyponym: animal such as LION and TIGER
- find hypernym: carnivore such as LION and TIGER

The basic idea is to learn terms and relationships by querying a Web search engine. As patterns language template capturing hypernym relationships are used. This is analogous to the approach with Hearst patterns for extracting ISA relationships. However, as in a first phase on has to detect more relevant terms, so-called double anchored patterns are used, that relate one (known) term to another (unknown) term of the same class of concepts. One example of such a pattern would be “c such as s and X”, where c and s would be known and from the result the term at position X would be a new term from the same class of concepts as c. Using such patterns new terms can be harvested by recursively applying the pattern to the terms known so far.

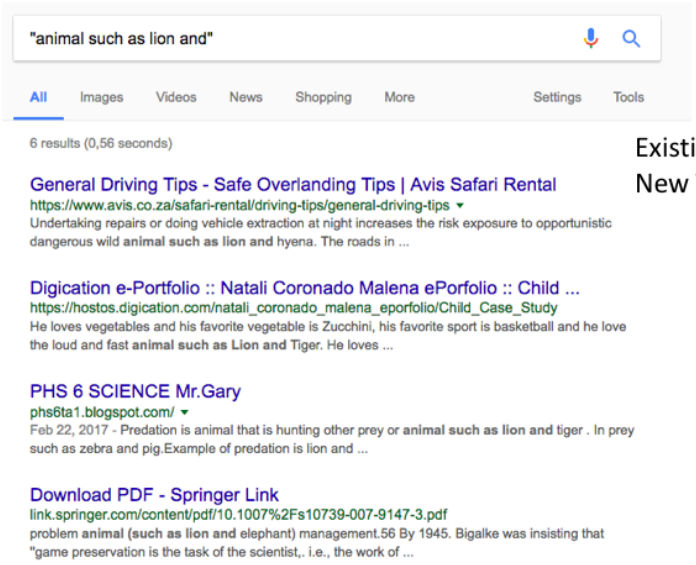
1. This step searches for more relevant terms (the graph is not directed yet)

Finding Hyponyms

Recursively harvest new terms using a Web search engine

```
T = {s}; w(t) = 0
while T changes
   for all t in T: submit P_i(c, t, X) to search engine
         add to T all new terms t_new found in position X in a result
         w(t)++
```

Example



Existing Term: lion
New Terms: hyena, tiger, elephant

This is an example of how terms related to an initial term “lion” would be found using double-anchored patterns.

Finding Hypernyms

C = {c}

for all t_1, t_2 in T with $w(t_i) > 0$:

submit $P_c(t_1, t_2, X)$ to search engine

add new term h found in position X to C

add t_1 *ISA* h and t_2 *ISA* h to the hypernym relations H

$w(t_1,t_2,h)$++

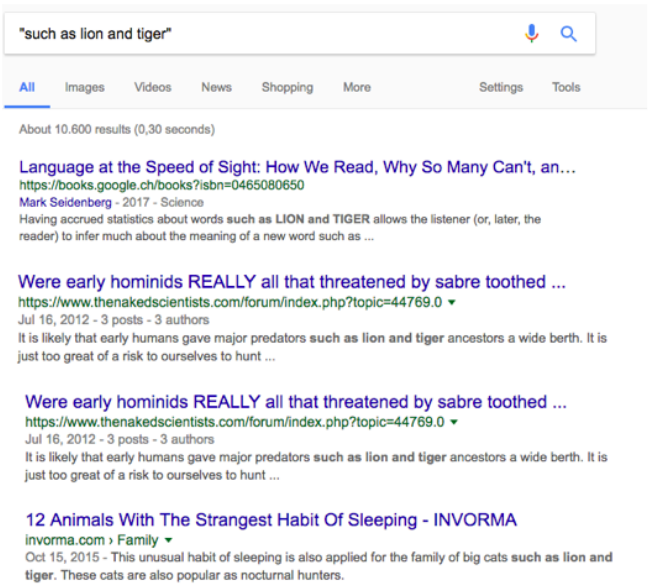
Filtering

- rank concepts h by $\sum_{t_1 t_2} w(t_1, t_2, h)$
- keep top concepts

Once the search for terms is completed, hypernyms can be searched by using the same patterns, but now be putting a variable X in the place of the concept, and using the basic terms found so far. In this search only terms that have been leading to the discovery of other basic concepts are being used, which is expressed by the condition w(t) > 0.

Once the search for higher level concepts is completed, a filtering step is performed. The concepts found are ranked by the number of times they have been discovered starting from different term pairs and only the highest ranked concepts are retained.

Example

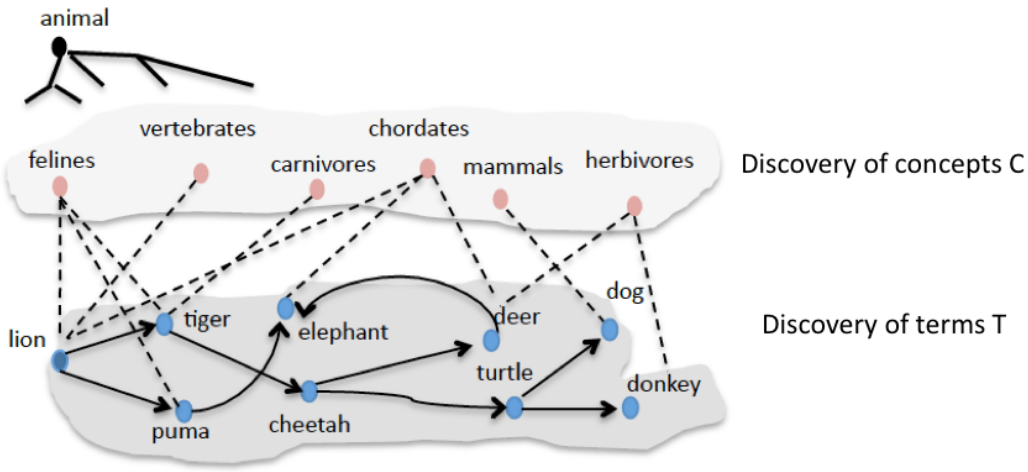


New Classes: predators, bug cats,

But also: words, …

Here a few examples of how higher level concepts related to the basic concepts “lion” and “tiger” can be found.

Example Result



As a result of the two previous step we obtain basic data on terms for basic concepts T and higher level concepts C, together with hypernym relationships indicated by dotted arrows.

Inducing Hypernym Graph

Many possible relationships among concepts and terms have likely not been discovered

For each pair t_1, t_2 in TUC

Construct query $q_1 = h(t_1, t_2)$ and $q_2 = h(t_2, t_1)$
with Hearst pattern h(X, Y), e.g., h(X, Y) = "X such as Y"

Submit query to search engine and count number of results

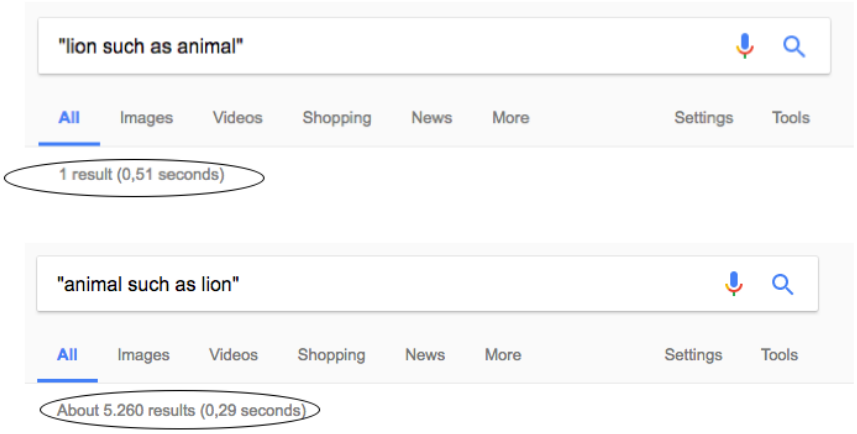
If #results(q_1) > #results(q_2)
then add t_1 *ISA* t_2 to H
else add t_2 *ISA* t_1 to H

Result: A directed hypernym graph H

In the steps performed so far, many possible relationships among higher level concepts and basic concepts and higher level concepts may not have been discovered. For discovering those again queries against a search engine are performed, testing the two possible directions of a hypernym relationship for each pair of terms. The alternative that produces more results in the search is considered as being the correct one. As templates we can use any of the Hearst patterns, such as "X such as Y", "X are Y that", "X including Y", "X like Y", "such X as Y" etc

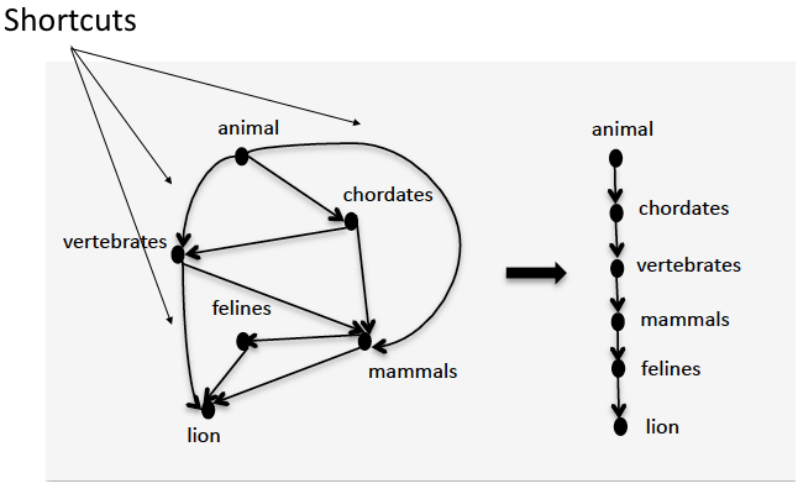
1. Previous page searches for more relevant terms (the graph is not directed yet)
2. This step establishes the hypernym / hyponym

Example



This example shows how powerful this method to test a direction of a relationship is.

Example



Graph H: may contain redundant paths

After adding all possible relationships to the set of concepts, we obtain a a graph that contains many redundant paths, including many of the induced transitive relationships. The method would, for example, find that animal ISA chordate, and chordate is a vertebrate, but also that animal is a vertebrate. In the last step such redundant relationships are removed.

induced transitive relationship
- animal -> chordate -> vertebrate
- animal -> vertebrate (redundant)

Cleaning the Hypernym Graph

1. Determine all **basic concepts**
   - Not hypernym of another concept
2. Determine all **root concepts**
   - Have no hypernyms
3. For each basic concepts - root concept pair:
   - Select all hypernym paths that connect them
4. Choose the longest hypernym paths for the final taxonomy

For cleaning up the taxonomy graph, first the basic and root concepts are identified, which are the ones that are not hypernym of another concept, resp. have no hypernym. Then for every possible pair of a root concept and a basic concept all paths between the two are extracted, and only the longest one (or the longest ones) is retained.

- for each basic concept, root concept pair
- find all paths
- select longest path

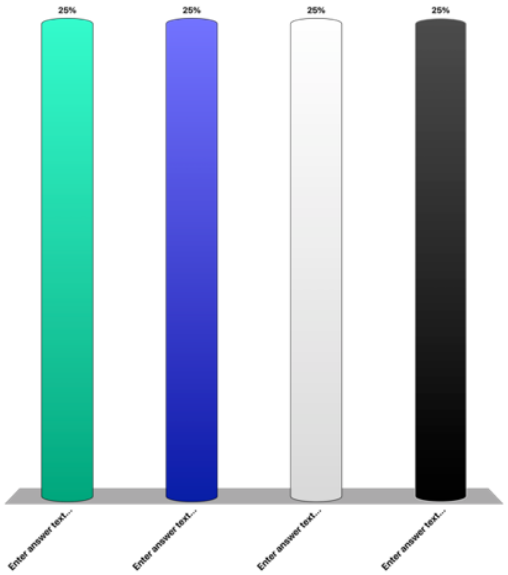
If t has no Hypernym ..

A. It is a root concept

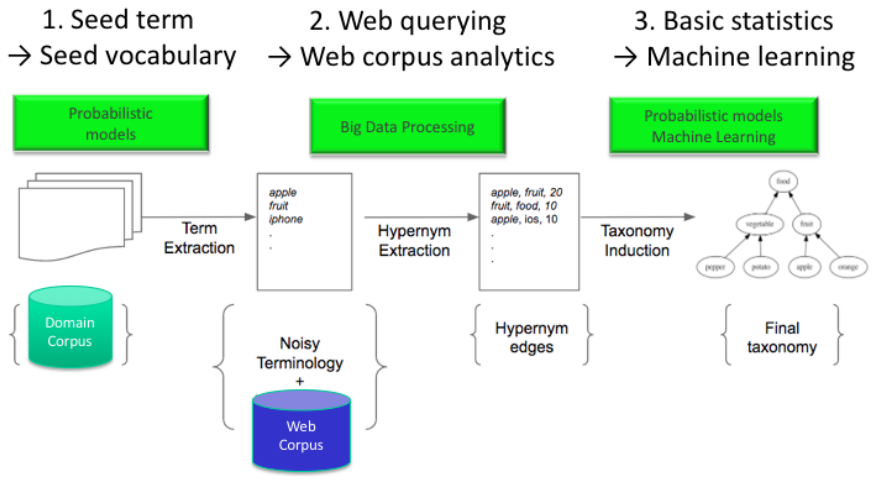
B. It cannot match c such as t and X

C. It is identical to the initial root concept

D. It is a basic concept



Taxonomy Induction: 7 years later



Gupta, Amit, et al. "Taxonomy Induction using Hypernym Subsequences.", CIKM 2017

We will present now a more advanced technique for automated taxonomy induction that we recently developed. It takes advantage of different advances in the available tools, but also incorporates some very fundamental ideas that have not been taken into account in the literature and largely help to improve the quality of results. The main ideas are the following:

1. Instead of starting from a clean predefined set of seed terms, the method starts from a document corpus and uses keyphrase extraction to generate an initial vocabulary. This vocabulary may be noisy, but the method will help to clean it up while generating the taxonomy.
2. Instead of querying the Web, the method uses a Hypernym database that has been generating from analyzing a Web corpus.
3. Instead of performing simple statistics, the method uses various machine learning techniques for inducing the taxonomy.

Hypernym Extraction: WebIsADB

http://webdatacommons.org/isadb/

Seitner, J., Bizer, C., Eckert, K., Faralli, S., Meusel, R., Paulheim, H. and Ponzetto, S.P., A Large DataBase of Hypernymy Relations Extracted from the Web. In *LREC* 2016

We have already seen WebIsADB that is used in this method for determining potential hypernym relationships.

Example: Hypernyms for “apple”

Candidate hypernym	Occurrence frequency
company	5536
fruit	3898
apple	2119
vegetable	928
orange	797
tech company	619
brand	463
hardware company	460
technology company	427
food	370

Note that hypernyms occur at difference level of abstraction
e.g. apple – fruit - food

WebISDB provides statistics of how often a term is in a possible hypernym relationship with another term. The example shows that the same term can be related to different terms at different levels of abstraction.

Key Observations

Current approaches in the literature make two main assumptions

1. The vocabulary is free of noise
   - Requires manual cleaning step before taxonomy induction
2. The quality of the taxonomy by estimating the probability of correctness of individual hypernym relationships
   - There is evidence that this works not well for more general terms

In particular the second assumption is important. Considering only individual hypernym relationships for assessing correctness means that contextual knowledge, i.e., the other relationships the terms have, is not considered and thus important information is lost.

Semantic Drift in Generalization

TopEdge	2	blintz→goody
	3	blintz→goody→thing
	4	blintz→goody→ulead→editor
	5	blintz→goody→ulead→social networking →networking→part
	6	blintz→goody→ulead→editor→storyliner→role
	2	oat→food
	3	oat→crop→thing
	4	oat→crop→total loss→partial loss→loss
	5	oat→cereal grain→grain→balanced diet→diet→factor
	6	oat→cereal→industry→field of life→other carrier→carrier

Considering only individual hypernym relationships frequently results in semantic drift, in particular at the higher levels of the taxonomy.

semantic drift occurs towards the right of the relation

Key Ideas

1. Allow noisy vocabulary, but clean the resulting taxonomy after induction
   - After taxonomy induction more information is available to identify noisy terms
2. Estimate the probability of correctness of a complete path of hypernym relationships!
   - More contextual information is exploited in assessing the correctness of hypernym relationships

Based on the two key observations, two key ideas are applied in order to achieve better performance in taxonomy induction. Both ideas are based on the approach of exploiting more contextual information when performing the taxonomy induction task.

Approach

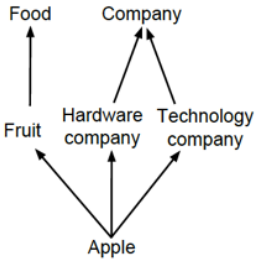
Find a DAG of generalizations

- Starting from one seed term in the vocabulary, e.g. “apple"
- Only one path for each hypernym of the seed term

freq(apple,company) = 5536 ?

Candidate hypernym	Freq.
company	5536
fruit	3898
apple	2119
vegetable	928
orange	797
tech company	619
brand	463
hardware company	460
technology company	427
food	370

Evidence E



Graph G

The basic approach of the method is to induce for each term in the vocabulary a DAG that consists of hypernym paths from the term to a root concept. In the approach it is assumed that for every hypernym of the term only a single path is generated. Each of the paths can correspond to a different sense of the term.

Probabilistic Model

Ideally: given the evidence E, find the most probable Graph G

$argmax_G\ P(G|E)$ (not feasible)

Aproximation: $argmax_G\ P(G|E) = argmax_G \prod_i P(E|S_i) \times P(S_i)$

where S_i subsequence from seed term to a root

An independence assumption

- But weaker than assuming all hypernym edges are independent of each other!

Need to estimate $P(E|S_i)$ and $P(S_i)$

Ideally, we would find the most likely graph structure, given the evidence E provided in the hypernym database. Since it is not feasible to explore the complete space of all possible graphs, we will apply an independence assumption. We assume that the different paths from a leaf to the root of a hypernym graph are independent of each other (which is of course not correct, since the paths are in general overlapping). But in this way we need no more to estimate the probability of a graph, but only those of path. Note that this is still a weaker independence assumption than the one being made when assuming that al hypernym edges are independent of each other.

Solution Strategy

1. Estimation of probabilities of subsequences
2. Search strategy for subsequences
3. Optimizing the resulting DAG

Probability Estimation: P(S)

For $S = t \rightarrow h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow \ldots \rightarrow h_n$ Subsequence S

$P(S) = P(t, h_1) \times P(h_1, h_2) \times \ldots \times P(h_{n-1}, h_n)$

P(a, b) - edge probability

Edge Probability

$P(a, b) \propto \exp(\mathbf{w} \cdot f(a, b))$

Edge features f(a, b)

- Normalized count, $n(a, b) = freq(a, b) / \max_c (freq(a, c))$
- Normalized difference
- String-based features (prefix, suffix, substring, length)
- Generality based features

Weights **w** obtained from a classifier trained on a manually annotated set of edges

Probability Estimation: P(E|S)

For $S = t \rightarrow h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow \ldots \rightarrow h_n$

$P(E|S) = \sum_j P(E_j|S)$

$P(E_j|S) \propto \max(\text{sim}(E_j, h_i))$ for all h_i

$\text{sim}(a, b) = \max(P(a, b), P(b, a))$

- why do we not want unrelated terms to hurt ?
- probably so we can get both graphs out

Why maximum (and not, e.g., sum)?

- Consider S = *apple→fruit→food→substance→matter→entity*
- For the term E_j = *fruit,* the hypernyms *matter* and *entity* occurring in the sequence should not hurt, even when unrelated

Intuitive Interpretation

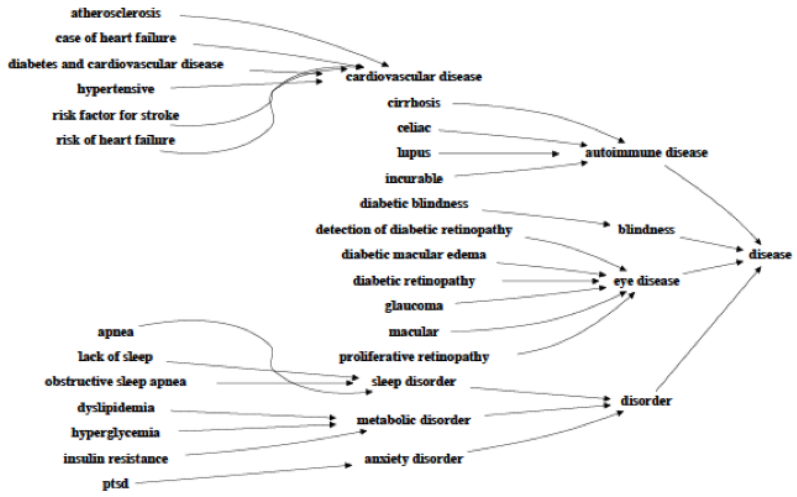
Pr(S) promotes subsequences, which consist of individual edges with a larger probability of hypernymy

Pr(E|S) promotes subsequences, which contain a larger number of candidate hypernyms from E

Results

TopEdge	2	blintz→goody
	3	blintz→goody→thing
	4	blintz→goody→ulead→editor
	5	blintz→goody→ulead→social networking →networking→part
	6	blintz→goody→ulead→editor→storyliner→role
	2	oat→food
	3	oat→crop→thing
	4	oat→crop→total loss→partial loss→loss
	5	oat→cereal grain→grain→balanced diet→diet→factor
	6	oat→cereal→industry→field of life→other carrier→carrier
SubSeq	2	blintz→homemade jewish food→food
	3	blintz→homemade jewish food→food→supply
	4	blintz→thin pancake→pastry→snack food→food
	5	blintz→homemade jewish food→food→supply→necessity→thing
	6	blintz→homemade jewish food→food→supply→keyword→beta test→test
	2	oat→cereal grain→grain
	3	oat→cereal grain→grain→supply
	4	oat→cereal grain→grain→complex carbohydrate→carbohydrate
	5	oat→cereal grain→grain→complex carbohydrate→carbohydrate→essential nutrient→nutrient
	6	oat→cereal grain→grain→supply→keyword→beta test→test

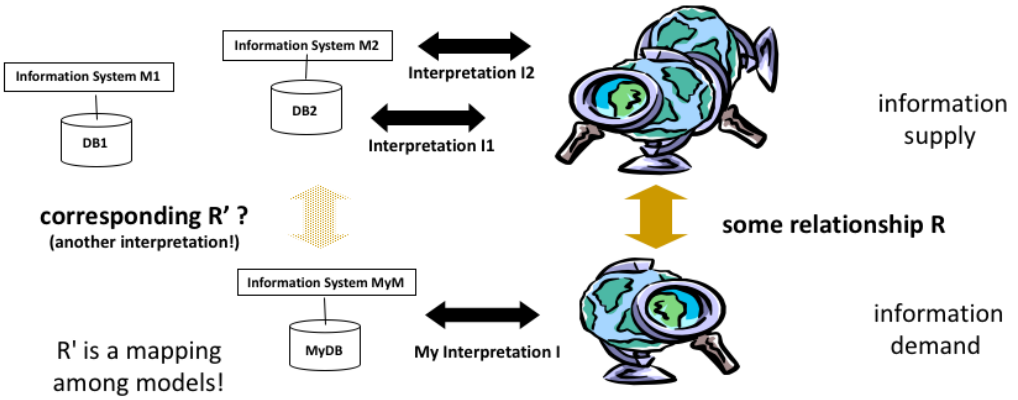
Example: Resulting Taxonomies



2. SCHEMA INTEGRATION

Key Tasks in Distributed Information Management

More data! ... More models!? ... More useful information?



This figure illustrates the problem of information starvation: there exist many information systems supplying data, but each having their own view on the world, which does not necessarily match the needs or understanding of a specific consumer. Every information system is interpreting its model differently with respect to the real world and relating to different views on the real world. Though there exists some relationships among all these views on the real world (let's denote it as R), and it surely implies some relationship R' among the different models used in the different information systems, the consumers of the information cannot easily understand the relationship R, and thus can also not easily relate their models to the models of others via the relationship R'. From the viewpoint of the data providers, introducing R' introduces a new interpretation of their data with respect to the model used by the data consumer.

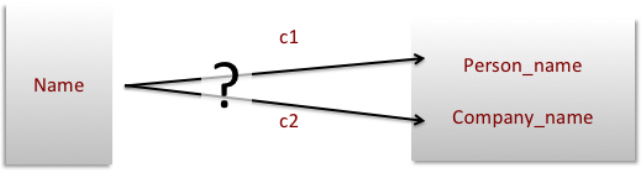
Schema Matching

Integration of heterogeneous data sources

- Every project on Big Data analysis first has to integrate data from different, heterogeneous data sources
- Different schemas, different taxonomies
- One of the long-standing open problems in data management

How to find good “matches”?

How to choose the “best matches”?



(both industry and research)

Schema Matching Approaches

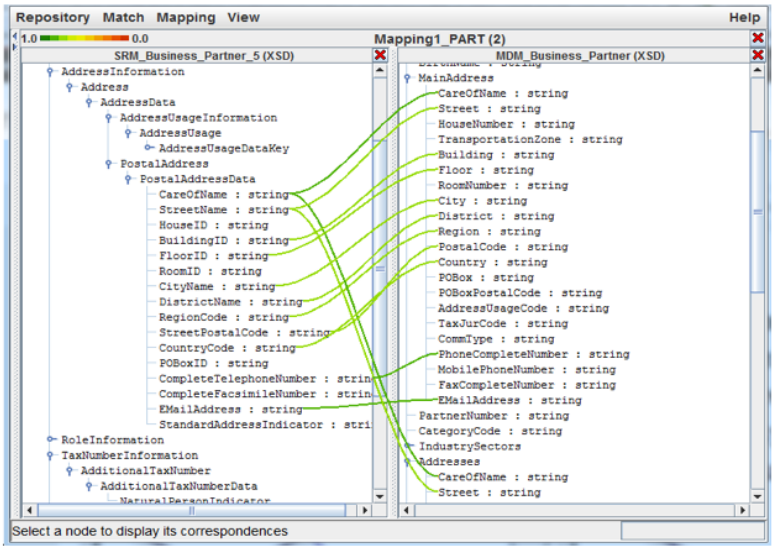
Manual matching

- still common practice today

Schema matching tools

- Based on structural and content features
  - names, domains, structure, values, …
- Establish correspondences and rank according to quality
  - Errors are frequent and unavoidable
  - Works well for small schemas

Schema Matching Tools



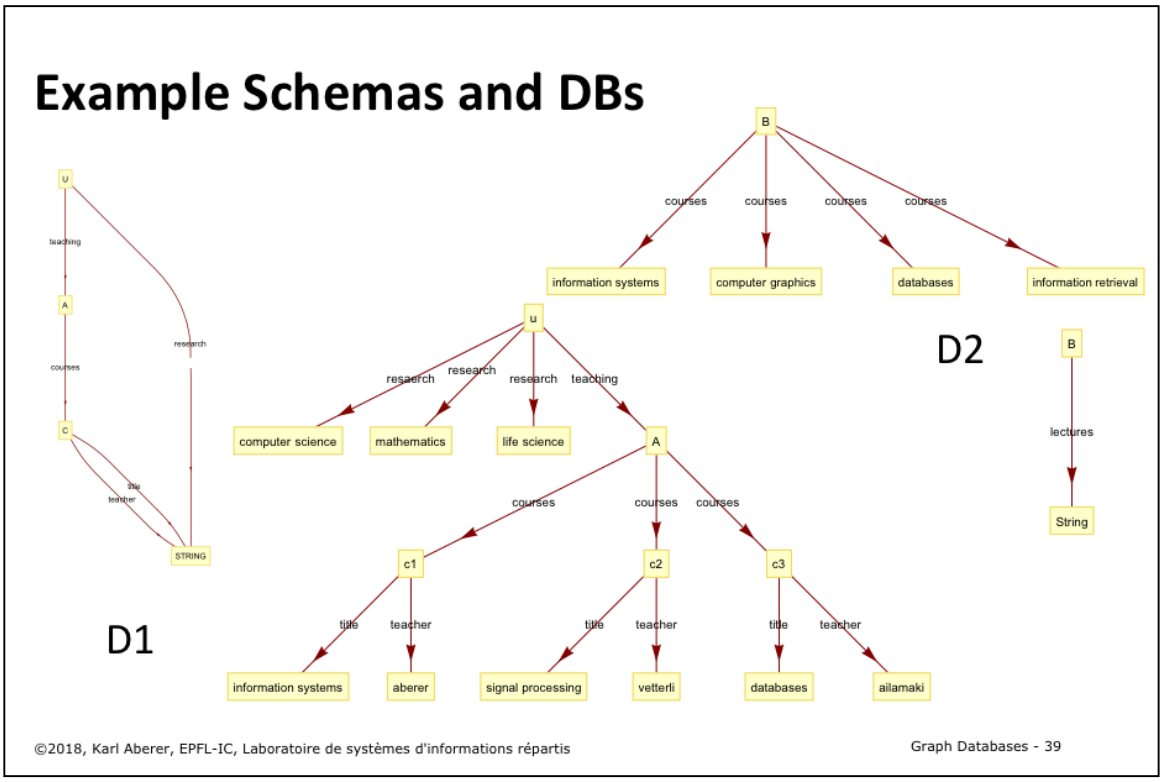
The Problem

Given two (RDF) schemas S1, S2 with two (RDF) databases D1, D2

Goal: find a 1:1 matching of schema nodes that have the same or similar "meaning"

Assumption: two schema nodes (classes) are considered similar, if they have the same or similar instances

In the following we will study the problem of integrating heterogeneous databases for the example of graph databases. We assume that we want to integrate data that are available in two graph databases and that are semantically related to each other. As explained earlier database schema integration proceeds in three steps: 1. matching of corresponding schema elements 2. resolving conflicts, and 3. establishing a mapping. In the case we will study we focus mainly on step 1.



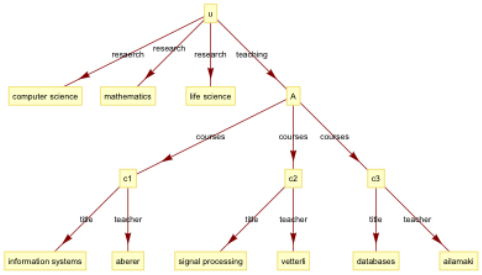
Let us assume that we have two example graph databases together with a schema. By inspecting the databases, a human can easily recognize that class A in database D1 corresponds to class B in database D2. The problem is how to perform the task of identifying this correspondence automatically.

Capital U - Universe
small u - instance

Universe of a Database

Universe U is a finite set of possible instances

Example:
U = {u, computer science, mathematics, life sciences, A, c1, c2, c3, information systems, aberer, signal processing, vetterli, databases, ailamaki, …}



One basic approach to assess whether two classes are similar, is to measure their level of similarity at the content level, i.e., to test to which extent the elements (=instances) of the two classes overlap. The Jaccard measure is a standard approach to measure such a set similarity.

Similarity of Classes

A, B are classes, classes are subsets of U

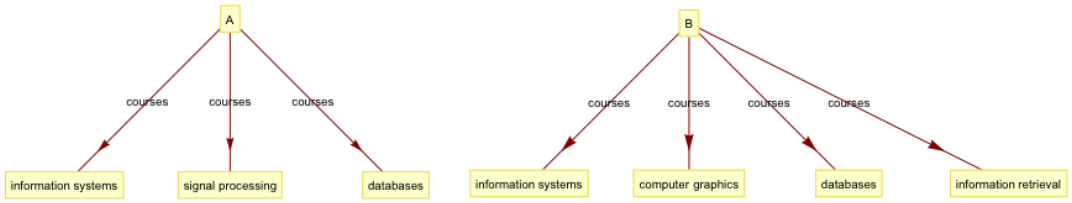
Similarity measure (Jaccard similarity)

$$sim(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{P(x \in A \text{ and } x \in B)}{P(x \in A \text{ or } x \in B)} = \frac{P(A,B)}{(P(A,B)+P(\bar{A},B)+P(A,\bar{B}))}$$

where $P(A,B) = P(x \in A \text{ and } x \in B)$ and
$P(A,\bar{B}) = P(x \in A \text{ and } x \notin B)$ etc

Example

$$sim(A, B) = \frac{2}{5}$$



Note: instances of A are
"information systems", "signal processing", "databases"

In this example the two classes have 2 common elements, and the number of all elements in their union is 5, thus the Jaccard similarity is 2/5.

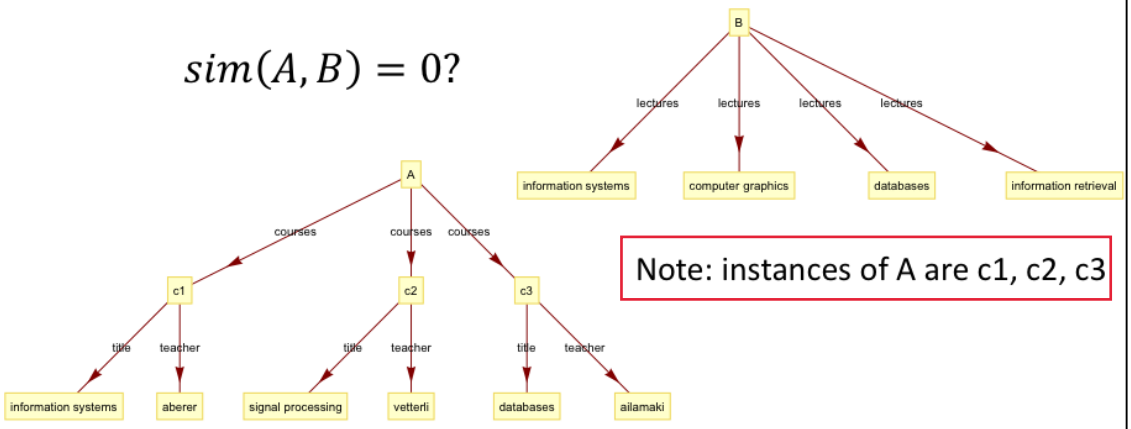
Note: we are exploiting the flexibility of the graph database model in the following examples to simplify the presentation in the following way: we represent classes directly within the graph database (in this example nodes A and B). We could also study the integration problem at the level of the database schema by identifying correspondences at the schema level, but this would unnecessarily complicate the presentation.

Problem 1

A similar to B?

Same information has **structurally different representation**

$sim(A, B) = 0?$



Note: instances of A are c1, c2, c3

In the previous example, the instances of the two classes have been leaf nodes in the graph databases. Therefore it was straightforward to compare their instances directly. In a general graph database, the correspondences are not always that simple. In this example, we see that classes A and B correspond to each other, even if A has as instances complex data (nodes ci) and B atomic data (Strings). More complex features of classes are required. For the example shown, this is easy to resolve, by considering the atomic data found at leaf level for an inner node, such as A.

- Instances of A are complex data because they each contain multiple instances

Problem 2

Two databases might have

- **Classes** with similar meaning (e.g. Course)
- Different **Instances** for those classes (e.g. different courses at different universities)

Intension is similar, but **extension** is different

Due to different database extensions and different naming conventions even classes that have a strong semantic similarity often have no common instances in two different databases

Complex Features of Classes

Considering the instances (direct children) of classes is only an example of a simple class feature

Many complex features can be exploited (and have been exploited in schema integration)

- Names of attributes and relationships
- Structural relationships (data types)
- Distributional features (data values)
- Content features (e.g., text)

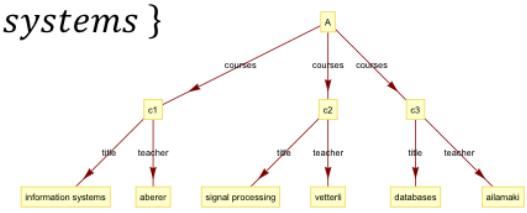
In database integration, many different features that can be associated with structural elements of a database or database schemas have been considered for establishing semantic correspondences. For illustration, we will use in the following a simple feature, the set of atomic data values (strings) that can be reached from a node in the graph database.

Content Feature of Classes

We consider as content feature, the set of terms associated with the paths of an node i to the leaves

$T_i = \{t_1, \ldots, t_n\}$ with repeated occurrences (bag of words)

- $T_A = \{\, aberer,$ $information\ systems, signal\ processing, vetterli, \ldots\}$
- $T_{c1} = \{\, aberer,\ information\ systems\,\}$



- all leaves of the subtree

Another feature that can be exploited for this case would be the labels used along the paths.

Finding Corresponding Classes

If U1 and U2 are the universes of DB1 and DB2 we may assume

$$U1 \cap U2 = \emptyset$$

Thus no way to compute $P(A,B)$ directly!

Given $i \in A$, the question is whether it would be likely that also $i \in B$, even if i is not part of U2, considering its features

Even if we have identified features that can help to spot correspondences between structurally different, but semantically equivalent elements of two databases, it might be the case that the coverage of a real-world aspect in two databases is different (e.g. courses in two different universities). Thus, directly comparing the features (e.g. the instances of a class) does not help to detect the correspondences.

Probabilistic Approach

We want to construct a function the gives the probability for a given instance i with feature T_i to belong to a class A

$$P(A|T_i) = P(T_i|A)P(A)P(d) \propto P(T_i|A)P(A) \text{ (Bayes law)}$$

$P(A|T_i)$ is a Naïve Bayes classifier to determine whether i belongs to A

Objective: determine $P(T_i|A)$ and $P(A)$

To tackle the problem, we will construct a model that determines (probabilistically) whether a given data instance with certain features is semantically related to a class.

Naïve Bayes Classifier

We know $P(A) = \frac{|A|}{|U_1|}$

Independence assumption: $P(T_i|A) = P(t_1|A) \dots P(t_n|A)$

With T_A being the bag of all terms occurring in all instances of A we have

$$P(t|A) = \frac{|t \in T_A|}{\sum_{t'}|t' \in T_A|}$$

Computing P(A) is straightforward. We have just to determine the relative frequency of instances of A (how big A is) in the set of all possible instances.

For computing P(Ti | A) we first make an independence assumption: we assume that different terms occurring in the instance i of a class A are independent of each other. In practice this is not the case, but is a generally accepted assumption for simplicity. Then we have to compute the probability of a single term t in class A to occur.

By computing P(A) and P(Ti | A) we obtain (up to a constant) the probability of for a (new) instance I to belong to class A.

Example

Classifier for class $A = \{c1, c2, c3\}$

$T_A = \{information\ systems, aberer, signal\ processing, \dots\}$

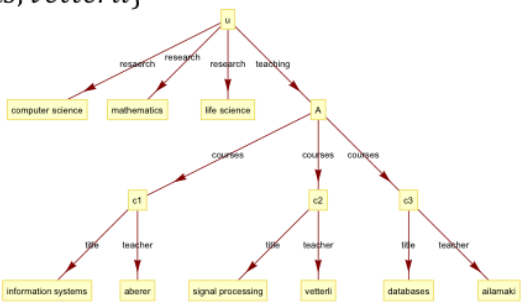
New instance cn: $T_{cn} = \{information\ systems, vetterli\}$

$P(vetterli|A) = \frac{1}{6}, P(vetterli|U) = \frac{1}{9}$

1/6 * 1/6 → $P(T_{cn}|A) = \frac{1}{36}, P(T_{cn}|U) = \frac{1}{81}$

$P(A) = \frac{3}{13}, P(U) = \frac{4}{13}$ (13 instances total)

$P(A|T_{cn}) \propto \frac{1}{36}\frac{3}{13}, P(U|T_{cn}) \propto \frac{1}{81}\frac{4}{13}$



Thus instance *cn* is considered to correspond more likely to *A* than to *U*

We show here a complete example of how for a new instance cn, the most likely corresponding class can be determined in the database. The Naïve Bayes classifier assigns a higher probability for cn to belong to A than to U (which coincides with our intuition).

- an instance can be a class
- a class can be an instance
- a class carry multiple instances

P(vetterli |A) = number of vetterli as leaves of A
---
total number of leaves of A

P(A) = number of direct children of A
---
total number of instances in the universe

Computing Similarity between Classes A and B

1. Take all instances of $U1$ and train a classifier to decide whether an instance belongs to A or not
2. Select all instances in $U2$ belonging to B: $U2^B$
3. Apply the classifier trained with $U1$ to all instances in $U2^B$ to produce set $U2^{AB}$
4. Do the same with roles of A and B exchanged
5. Compute $P(A,B) = \frac{|U1^{AB}|+|U2^{AB}|}{|U1|+|U2|}$
6. Compute similarly $P(A,\bar{B})$ etc
7. Then compute $sim(A,B) = \frac{P(A,B)}{(P(A,B)+P(\bar{A},B)+P(A,\bar{B}))}$

For computing Jaccard similarity between two classes A and B we need to know how many elements are contained in the intersection of A and B and the union of A and B. To that end we can first separate in each of the two databases the elements that belong to the class present in the database (e.g. A in database D1 with universe U1), and then apply the classifier learnt from the other database, whether an element in each of those two sets belongs also to the other class, or not. We consider $U2^{AB}$ as the set of elements of B that are likely to belong to A, if they where part of database D1. In this way we determine (or better estimate) the sizes of the potential intersection of A and B and the union of A and B and can based on this compute an estimate for Jaccard similarity.

Node Mapping

With the similarity values, alternative class mappings are possible

Naïve approach

- Order matchings by probability
- Choose the most probable matching and produce a mapping among the classes
- Remove the mapped classes
- Choose the next most probable matching and repeat

Drawback

- Obvious consistency constraints may be violated, e.g., if all children of a class are mapped, then also their parent class should be mapped
- Solution: more sophisticated mapping algorithms that incorporate such general and domain-specific constraints (research problem)

Once the similarity values are computed the process of matching for establishing correspondences is completed. Now the problem of mapping remains, i.e. determine which classes should be considered as equivalent. This is in general not uniquely determined, since a class can be similar to several others. On the other hand we can assume that one class should only be mapped to exactly another class in the other database. This imposes a constraint on the mapping.

A simple approach to establish a mapping that observes this constraint is by proceeding in a greedy fashion. First the best matches are used to produce mappings, then the the mapped classes are removed (to assure that each class is mapped to a unique other class) and then the process continues.

References

Course material based on

- Doan, AnHai, et al. "Learning to map between ontologies on the semantic web." *Proceedings of the 11th international conference on World Wide Web*. ACM, 2002.
- Kozareva, Zornitsa, and Eduard Hovy. "A semi-supervised method to learn and construct taxonomies using the web." *Proceedings of the 2010 conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2010.
- Gupta, A., Lebret, R., Harkous, H., & Aberer, K. (2017, November). Taxonomy Induction using Hypernym Subsequences. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (pp. 1329-1338). ACM.