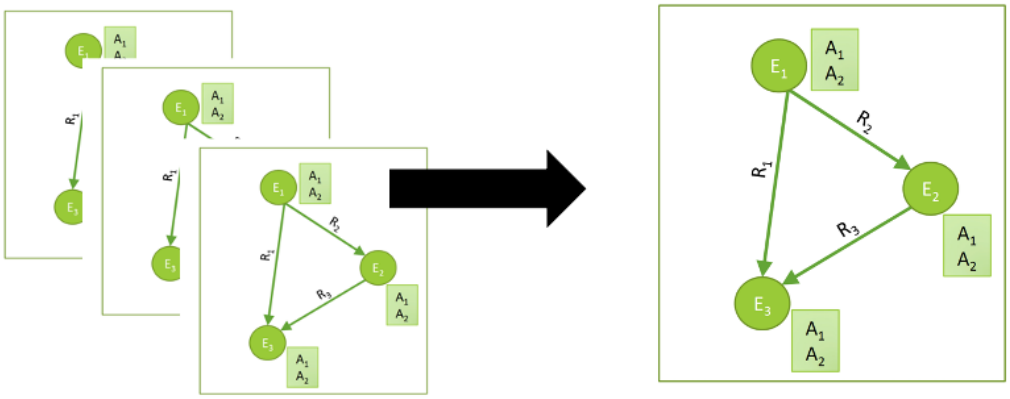
4.4 KNOWLEDGE INFERENCE

Knowledge Inference

From available knowledge to more complete and precise knowledge



Once knowledge graphs have been extracted from text, they can be further processed. This enables the inference of new knowledge from the existing knowledge, but as well the correction, completion and integration of existing knowledge bases.

Basic Questions in Knowledge Inference

Who are the entities?

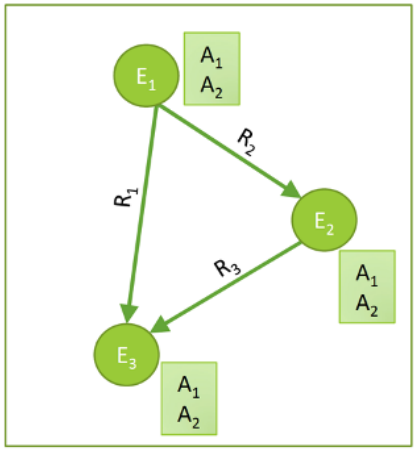
- ➢ Entity Linking / Disambiguation
- ➢ Data integration

What are their attributes?

- ➢ Collective Classification

How are they related?

- ➢ Link prediction



Knowledge inference concerns a wide number of problems that have been studied in different contexts. Some of the basic examples are:

- Entity linking and disambiguation, which concerns the problem of identifying which entity names represent the same real-world entity, respective which entity is referred to in case of ambiguous entity names.
- Schema integration, which concerns the problem which classes, attributes and relationships in one knowledge base correspond to which in another one.
- Collective classification, which concerns the problem of learning unknown attribute values from the available knowledge in a knowledge base.
- Link prediction, which concerns the problem of learning unknown relationships from the available knowledge in a knowledge base.

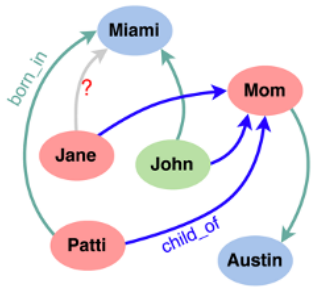
4.4.1 Link Prediction

Large knowledge bases are usually incomplete

- DBPedia: 60% of persons miss place of birth
- FreeBase: 71% of persons miss place of birth etc.

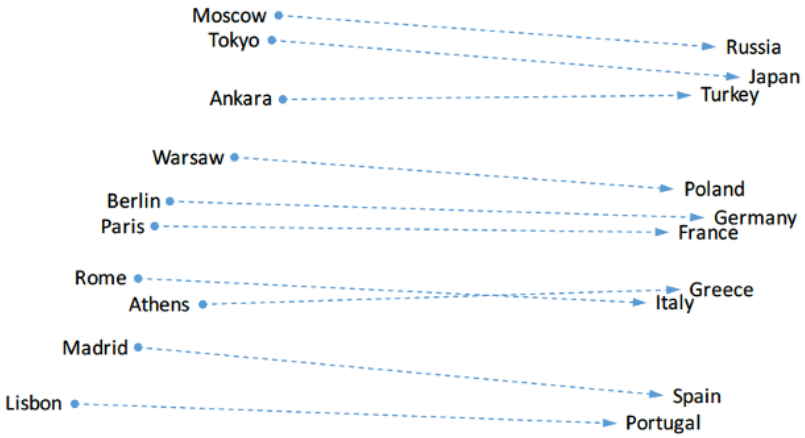
Try to predict missing links from existing data

Jane born in Miami?



In general, knowledge bases are incomplete. Thus, there is a significant interest in completing the relationships among entities. To do so one might exploit “patterns” that entities and relationships follow and generalize them.

Observation on Word Embeddings



In order to tackle the problem of link prediction, we come back to an observation that we had made earlier for word embeddings. We have seen that relationships seem to be represented as linear transformations.

Relations in Word Embeddings

$$\begin{aligned} \boldsymbol{v}_{Japan} - \boldsymbol{v}_{Tokyo} &\approx \boldsymbol{v}_{Germany} - \boldsymbol{v}_{Berlin} \\ \boldsymbol{v}_{Germany} - \boldsymbol{v}_{Berlin} &\approx \boldsymbol{v}_{Italy} - \boldsymbol{v}_{Rome} \\ \boldsymbol{v}_{Italy} - \boldsymbol{v}_{Rome} &\approx \boldsymbol{v}_{Portugal} - \boldsymbol{v}_{Lisbon} \end{aligned}$$

Idea: Find a vector $v_{is\_capital\_of}$ such that

$$\begin{aligned} \boldsymbol{v}_{Tokyo} + \boldsymbol{v}_{is\_capital\_of} - \boldsymbol{v}_{Japan} &\approx \mathbf{0} \\ \boldsymbol{v}_{Berlin} + \boldsymbol{v}_{is\_capital\_of} - \boldsymbol{v}_{Germany} &\approx \mathbf{0} \\ \boldsymbol{v}_{Rome} + \boldsymbol{v}_{is\_capital\_of} - \boldsymbol{v}_{Italy} &\approx \mathbf{0} \\ \boldsymbol{v}_{Lisbon} + \boldsymbol{v}_{is\_capital\_of} - \boldsymbol{v}_{Portugal} &\approx \mathbf{0} \end{aligned}$$

Relations are also represented as embedding vectors!

We can express the linear relationship among the embeddings of two types of entities, e.g., countries and their capitals, by stating that the differences of the embedding vectors are similar. Since the differences of the vectors are similar, we can also introduce a vector for this difference, which we call $v_{is\_capital\_of}$. This vector can be considered as a representation of the relationship. This formulation of the problem is the starting point for methods for identifying new relationships in knowledge graphs using embedding techniques.

Model

Knowledge graph G consists of (correct) triples (h, r, t) where $h, t \in E$ and $r \in R$

Each entity and relationship is mapped to a low-dimensional vector, resulting in $\boldsymbol{v}_h, \boldsymbol{v}_r, \boldsymbol{v}_t$

To formulate the model, assume that we have a knowledge graph consisting of triples (h, r, t). h indicates the term “head” and t the term “tail”, and both are entities. The objective is to find low-dimensional vectors representing both the head and tail entities, and the relationships.

Plausibility Score

Define a (im)plausibility score $\mathrm{f}(\boldsymbol{v}_h, \boldsymbol{v}_r, \boldsymbol{v}_t)$ such that

$$\mathrm{f}(\boldsymbol{v}_h, \boldsymbol{v}_r, \boldsymbol{v}_t) < \mathrm{f}(\boldsymbol{v}_{h'}, \boldsymbol{v}_{r'}, \boldsymbol{v}_{t'})$$

if (h, r, t) is a plausible triple and (h', r, t') is an implausible triple for relation r

$(h', r, t') \in G'(h, r, t)$ is a set of incorrect triples, generated by corrupting the correct triple (h, r, t)

For learning the model, we need to introduce a loss function. To define this loss function, we first introduce a plausibility score $\mathrm{f}(\boldsymbol{v}_h, \boldsymbol{v}_r, \boldsymbol{v}_t)$ for triples of embedding vectors. This score can be thought of as the inverse of the probability a triple triple to represent a correct fact. That means that more plausible the fact is the lower the plausibility score, approaching zero.

The property that we require the score to satisfy is that correct triples have a higher score than incorrect triples. This requires examples of both correct and incorrect triples. For correct triples we can use facts from a given knowledge graph. For obtaining incorrect triples we can take a correct fact and corrupt it by modifying parts of the fact with random replacements.

Learning the Model

Minimize the margin loss function

$$J(\theta) = \sum_{\substack{(h,r,t)\in G \\ (h',r,t')\in G'(h,r,t)}} \max(0, \gamma + \mathrm{f}(\boldsymbol{v}_h, \boldsymbol{v}_r, \boldsymbol{v}_t) - \mathrm{f}(\boldsymbol{v}_{h'}, \boldsymbol{v}_r, \boldsymbol{v}_{t'}))$$

$\gamma > 0$ is a hyperparameter

The loss function attempts to separate scores from correct and incorrect triples by the margin γ

Using the plausibility function, we can formulate a loss function that should be minimized. The form of the function is a margin loss function. It tries to separate the positive from the negative samples, using the plausibility score. The hyperparameter γ determines by which margin the optimization will try to separate the plausibility scores of correct vs. incorrect triples.

The optimization is performed as usual with SGD.

Example

if $\gamma = 2$ and $f(\boldsymbol{v}_h, \boldsymbol{v}_r, \boldsymbol{v}_t) = 0$, then

if $f(\boldsymbol{v}_{h'}, \boldsymbol{v}_r, \boldsymbol{v}_{t'}) = 2$ or 3, then

$$\max(0, \gamma + f(\boldsymbol{v}_h, \boldsymbol{v}_r, \boldsymbol{v}_t) - f(\boldsymbol{v}_{h'}, \boldsymbol{v}_r, \boldsymbol{v}_{t'})) = 0$$

if $f(\boldsymbol{v}_{h'}, \boldsymbol{v}_r, \boldsymbol{v}_{t'}) = 1$, then

$$\max(0, \gamma + f(\boldsymbol{v}_h, \boldsymbol{v}_r, \boldsymbol{v}_t) - f(\boldsymbol{v}_{h'}, \boldsymbol{v}_r, \boldsymbol{v}_{t'})) = 1$$

☞ in order to minimize $J(\theta)$ $f(\boldsymbol{v}_{h'}, \boldsymbol{v}_r, \boldsymbol{v}_{t'}))$ needs to be increased

This example illustrates of how the loss function forces a separation of the scores of plausible and implausible tuples.

TransE Model

One of the first embedding-based models for knowledge base completion

- Based on the intuition from text WE

$$f(\boldsymbol{v}_h, \boldsymbol{v}_r, \boldsymbol{v}_t) = \|\boldsymbol{v}_h + \boldsymbol{v}_r - \boldsymbol{v}_t\|_2$$

Each entity and relationship is mapped to a low-dimensional vector, resulting in $\boldsymbol{v}_h, \boldsymbol{v}_r, \boldsymbol{v}_t$

The generic model described so far can be instantiated using different choices for the plausibility score f. In an early version the plausibility score was computed following the inspiration from the initial observation made for relationships in word embeddings. It introduces a vector $\boldsymbol{v}_r$ to represent the relationship and compares it to the difference of head and tail entity embedding vectors.

Performing SGD

Initialize vectors with random values

- From interval $[-\frac{c}{\sqrt{k}}, \frac{c}{\sqrt{k}}]$ where k is the embedding dimension
- In each iteration
  - Sample a correct triple or batch
  - Derive a corrupt triple from the correct one: replace h or t by a random entity
  - Update embeddings by minimizing loss function
  - Normalize all entity vectors to 1 (not relationship vectors!)

The SGD algorithm proceeds as follows. First the vectors are initialized with random values (the choice is motivated by empirical findings from neural network training. In the published paper c = 6). Then in every iteration a triple (or several triples are randomly chosen). Negative samples are generated by randomly replacing head or tail (not both). The update of the embeddings is performed as usual by computing the differential of the loss function. Entity vectors are normalized to 1 in every iteration (this avoids the model to find a trivial solution).

Qualitative Results

Not found in KB, but correct Found in KB

INPUT (HEAD AND LABEL)	PREDICTED TAILS
J. K. Rowling `influenced by`	*G. K. Chesterton*, J. R. R. Tolkien, *C. S. Lewis*, **Lloyd Alexander**, Terry Pratchett, Roald Dahl, Jorge Luis Borges, *Stephen King*, Ian Fleming
Anthony LaPaglia `performed in`	*Lantana*, *Summer of Sam*, *Happy Feet*, *The House of Mirth*, Unfaithful, **Legend of the Guardians**, Naked Lunch, X-Men, The Namesake
Camden County `adjoins`	**Burlington County**, *Atlantic County*, *Gloucester County*, Union County, Essex County, New Jersey, Passaic County, Ocean County, Bucks County
The 40-Year-Old Virgin `nominated for`	*MTV Movie Award for Best Comedic Performance*, *BFCA Critics' Choice Award for Best Comedy*, *MTV Movie Award for Best On-Screen Duo*, MTV Movie Award for Best Breakthrough Performance, **MTV Movie Award for Best Movie**, MTV Movie Award for Best Kiss, D. F. Zanuck Producer of the Year Award in Theatrical Motion Pictures, Screen Actors Guild Award for Best Actor - Motion Picture
Costa Rica football team `has position`	*Forward*, *Defender*, *Midfielder*, **Goalkeepers**, Pitchers, Infielder, Outfielder, Center, Defenseman
Lil Wayne `born in`	**New Orleans**, Atlanta, Austin, St. Louis, Toronto, New York City, Wellington, Dallas, Puerto Rico
WALL-E `has the genre`	Animations, Computer Animation, *Comedy film*, *Adventure film*, *Science Fiction*, **Fantasy**, Stop motion, *Satire*, Drama

The qualitative results show that the method works reasonably well. The bold phrases are the correct predictions. However, given that the knowledge base used for evaluation is incomplete, it is possible that also other predictions are meaningful, like the ones highlighted in italic.

Alternative Models

Model	Score function $f(h, r, t)$	Opt.
Unstructured	$\|\boldsymbol{v}_h - \boldsymbol{v}_t\|_{\ell_{1/2}}$	SGD
SE	$\|\mathbf{W}_{r,1}\boldsymbol{v}_h - \mathbf{W}_{r,2}\boldsymbol{v}_t\|_{\ell_{1/2}}$; $\mathbf{W}_{r,1}, \mathbf{W}_{r,2} \in \mathbb{R}^{k\times k}$	SGD
SME	$(\mathbf{W}_{1,1}\boldsymbol{v}_h + \mathbf{W}_{1,2}\boldsymbol{v}_r + \mathbf{b}_1)^\top(\mathbf{W}_{2,1}\boldsymbol{v}_t + \mathbf{W}_{2,2}\boldsymbol{v}_r + \mathbf{b}_2)$ $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^n$; $\mathbf{W}_{1,1}, \mathbf{W}_{1,2}, \mathbf{W}_{2,1}, \mathbf{W}_{2,2} \in \mathbb{R}^{n\times k}$	SGD
TransE	$\|\boldsymbol{v}_h + \boldsymbol{v}_r - \boldsymbol{v}_t\|_{\ell_{1/2}}$; $\boldsymbol{v}_r \in \mathbb{R}^k$	SGD
TransH	$\|(\mathbf{I} - \boldsymbol{r}_p\boldsymbol{r}_p^\top)\boldsymbol{v}_h + \boldsymbol{v}_r - (\mathbf{I} - \boldsymbol{r}_p\boldsymbol{r}_p^\top)\boldsymbol{v}_t\|_{\ell_{1/2}}$ $\boldsymbol{r}_p, \boldsymbol{v}_r \in \mathbb{R}^k$; $\mathbf{I}$: Identity matrix size $k \times k$	SGD
TransR	$\|\mathbf{W}_r\boldsymbol{v}_h + \boldsymbol{v}_r - \mathbf{W}_r\boldsymbol{v}_t\|_{\ell_{1/2}}$; $\mathbf{W}_r \in \mathbb{R}^{n\times k}$; $\boldsymbol{v}_r \in \mathbb{R}^n$	SGD
TransD	$\|(\mathbf{I} + \boldsymbol{r}_p\boldsymbol{h}_p^\top)\boldsymbol{v}_h + \boldsymbol{v}_r - (\mathbf{I} + \boldsymbol{r}_p\boldsymbol{t}_p^\top)\boldsymbol{v}_t\|_{\ell_{1/2}}$ $\boldsymbol{r}_p, \boldsymbol{v}_r \in \mathbb{R}^n$; $\boldsymbol{h}_p, \boldsymbol{t}_p \in \mathbb{R}^k$; $\mathbf{I}$: Identity matrix size $n \times k$	AdaDelta
lppTransD	$\|(\mathbf{I} + r_{p,1}\boldsymbol{h}_p^\top)\boldsymbol{v}_h + \boldsymbol{v}_r - (\mathbf{I} + r_{p,2}\boldsymbol{t}_p^\top)\boldsymbol{v}_t\|_{\ell_{1/2}}$ $\boldsymbol{r}_{p,1}, \boldsymbol{r}_{p,2}, \boldsymbol{v}_r \in \mathbb{R}^n$; $\boldsymbol{h}_p, \boldsymbol{t}_p \in \mathbb{R}^k$; $\mathbf{I}$: Identity matrix size $n \times k$	SGD
STransE	$\|\mathbf{W}_{r,1}\boldsymbol{v}_h + \boldsymbol{v}_r - \mathbf{W}_{r,2}\boldsymbol{v}_t\|_{\ell_{1/2}}$; $\mathbf{W}_{r,1}, \mathbf{W}_{r,2} \in \mathbb{R}^{k\times k}$; $\boldsymbol{v}_r \in \mathbb{R}^k$	SGD
TranSparse	$\|\mathbf{W}_r^h(\theta_r^h)\boldsymbol{v}_h + \boldsymbol{v}_r - \mathbf{W}_r^t(\theta_r^t)\boldsymbol{v}_t\|_{\ell_{1/2}}$; $\mathbf{W}_r^h, \mathbf{W}_r^t \in \mathbb{R}^{n\times k}$; $\theta_r^h, \theta_r^t \in \mathbb{R}$; $\boldsymbol{v}_r \in \mathbb{R}^n$	SGD
DISTMULT	$\boldsymbol{v}_h^\top\mathbf{W}_r\boldsymbol{v}_t$; $\mathbf{W}_r$ is a diagonal matrix $\in \mathbb{R}^{k\times k}$	AdaGrad
NTN	$\boldsymbol{v}_r^\top tanh(\boldsymbol{v}_h^\top\mathbf{M}_r\boldsymbol{v}_t + \mathbf{W}_{r,1}\boldsymbol{v}_h + \mathbf{W}_{r,2}\boldsymbol{v}_t + \mathbf{b}_r)$ $\boldsymbol{v}_r, \mathbf{b}_r \in \mathbb{R}^n$; $\mathbf{M}_r \in \mathbb{R}^{k\times k\times n}$; $\mathbf{W}_{r,1}, \mathbf{W}_{r,2} \in \mathbb{R}^{n\times k}$	L-BFGS
HolE	$sigmoid(\boldsymbol{v}_r^\top(\boldsymbol{v}_h \circ \boldsymbol{v}_t))$; $\boldsymbol{v}_r \in \mathbb{R}^k$, $\circ$ denotes *circular correlation*	AdaGrad
Bilinear-COMP	$\boldsymbol{v}_h^\top\mathbf{W}_{r_1}\mathbf{W}_{r_2}...\mathbf{W}_{r_m}\boldsymbol{v}_t$; $\mathbf{W}_{r_1}, \mathbf{W}_{r_2}, ..., \mathbf{W}_{r_m} \in \mathbb{R}^{k\times k}$	AdaGrad
TransE-COMP	$\|\boldsymbol{v}_h + \boldsymbol{v}_{r_1} + \boldsymbol{v}_{r_2} + ... + \boldsymbol{v}_{r_m} - \boldsymbol{v}_t\|_{\ell_{1/2}}$; $\boldsymbol{v}_{r_1}, \boldsymbol{v}_{r_2}, ..., \boldsymbol{v}_{r_m} \in \mathbb{R}^k$	AdaGrad
ConvE	$\boldsymbol{v}_t^\top g\left(\text{vec}\left(g\left(\text{concat}(\overline{v}_h, \overline{v}_r) * \boldsymbol{\Omega}\right)\right)\boldsymbol{W}\right)$; g denotes a non-linear function	Adam
ConvKB	$\mathbf{w}^\top \text{concat}\left(g\left([\boldsymbol{v}_h, \boldsymbol{v}_r, \boldsymbol{v}_t] * \boldsymbol{\Omega}\right)\right)$; $*$ denotes a convolution operator	Adam

The TransE method is only one example of numerous methods that have been in the meanwhile proposed to tackle the link prediction problem. In the meantime, also transformer models are used for this task.

Which is true? The score function f(h,r,t) …

A. has always larger values for triples (h,r,t) that are part of the known knowledge graph than for other triples
B. maps triples to vectors in the embedding space
C. is always positive
D. is optimized by stochastic gradient descent

Question

If $f(\boldsymbol{v}_h, \boldsymbol{v}_r, \boldsymbol{v}_t) = 0.1$ and γ is increased from 2 to 3, optimizing the loss function

A. is primarily achieved by decreasing the values of $f(\boldsymbol{v}_h, \boldsymbol{v}_r, \boldsymbol{v}_t)$
B. is primarily achieved by increasing the values of $f(\boldsymbol{v}_h, \boldsymbol{v}_r, \boldsymbol{v}_t)$
C. is primarily achieved by decreasing the values of $f(\boldsymbol{v}_{h'}, \boldsymbol{v}_r, \boldsymbol{v}_{t'})$
D. is primarily achieved by increasing the values of $f(\boldsymbol{v}_{h'}, \boldsymbol{v}_r, \boldsymbol{v}_{t'})$

4.4.2 Label Propagation

Inferring Attribute Values

Example: Which users on Twitter have positive or negative emotion towards a topic?

- Users are nodes in a graph (follower network)
- Emotion is an attribute of the node

Potential source of information in the case of Twitter

- Emoticons in tweets: indicate stance of user towards the topic
- Only a (small) fraction of the users is using emoticons

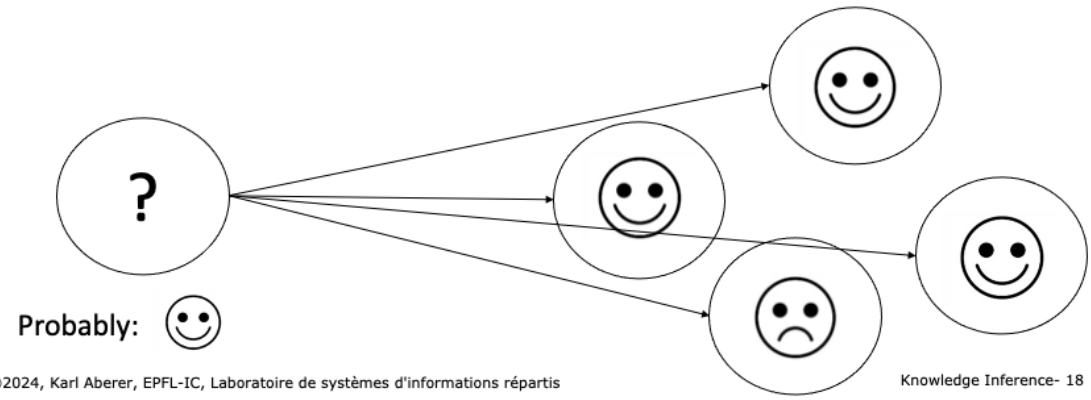
A second inference task in knowledge bases, after entities have been disambiguated, is to assign to the entities correct attribute values. For discrete attributes this problem can be understood as a classification problem. This question has, for example, been studied for classifying users in a social network with respect to their stance or emotion towards a specific topic.

In the case of emotion analysis there exists typically indications of emotions, e.g., in the form of the use of emoticons or specific hashtags. However, only few users are using those.

Propagating Attribute Values

Assumption: nodes that are connected by an edge, have a higher propensity of sharing the attribute of interest

- Twitter users following each other, are more likely to share the same emotion towards a topic

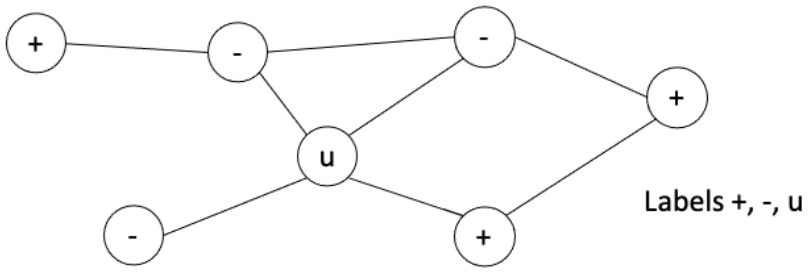


In many cases nodes that are connected by edges in a knowledge graph or as well in a social network, share properties. In social networks this is quite apparent. People that are connected through social links (e..g follower, friend, retweet, reply etc) are in general more likely to share opinions than those that are not. Is it possible to exploit this property to predict the attributes (respectively class lables) for those users that have none?

Model

Graph $G = (V, E)$ with vertices V and edges E

- Label set L of size n
- Vertices V have a label from a set $L \cup \{unknown\}$
- Edges are undirected and unweighted



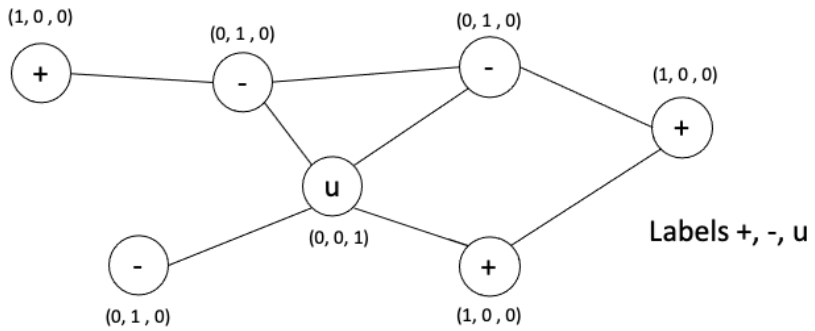
We consider the following model. A graph with vertices that can have either a label from a set L, or a label unknown. The edges are all undirected and unweighted. The model and approach can be extended for directed graphs with weighted edges.

We associate with vertices probability distributions that represent our knowledge about the assignment of a label. For all vertices we will compute an inferred probability distribution.

Optimization Objective

Objective

- Determine for a vertex v a label vector $\boldsymbol{l}_{inferred}(v)$ of size $n + 1$
- $\boldsymbol{l}_{inferred}(v)$ assigns a label probability



This example shows of how the probabilities of the labels, or the value unknown are initially distributed.

Label Inference

We assume that all neighbors excert the same influence on a node

Thus, we would require that

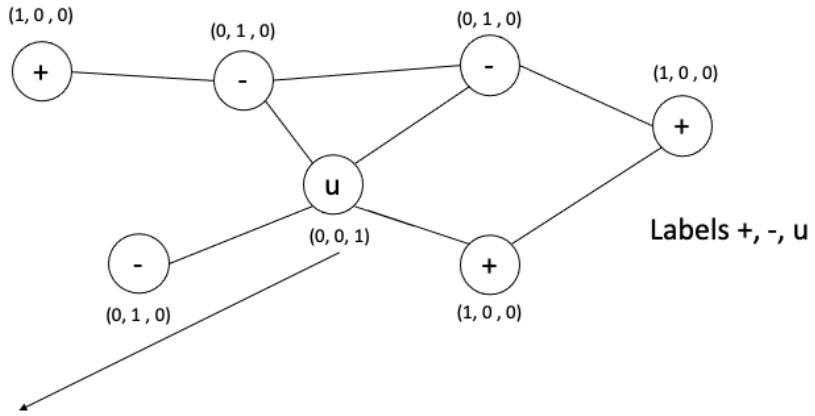
$$\boldsymbol{l}_{inferred}(v) = \frac{1}{\deg(v)} \sum_{(v,w)\in E} \boldsymbol{l}_{inferred}(w)$$

Recursive equation resp. random walk model (like PageRank)

In this model we assume that all neighbors of a node in the graph are equally influencing it. Thus, we can compute its expected label distribution by averaging the distributions of the neighbors. The resulting equation is analogous to the formulation of the PageRank model. Thus, the Label Inference can be interpreted as a random walk model.

The model can be extended to weighted graphs in which case the edge weight would be considered in the aggregation of the distributions of the neighboring vertices.

Example



$l_{inferred}$ = 1/4 ((0, 1 , 0) + (0, 1 , 0) + (0, 1 , 0) + (1, 0 , 0)) = (1/4, ¾, 0)

Injecting Pre-existing Knowledge

Initial knowledge on labels

Known labels

- $\boldsymbol{l}_{apriori}(v)$ is a vector of size $n+1$
- assigns weight 1 for label if known for $v \in V$

Unknown labels

- $\boldsymbol{l}_{unknown}$ is a vector of size $n+1$
- assigns weight 1 for label $unknown$

For some vertices we have an apriori assignment of labels (these are the vertices for which the label is known). For vertices that have no apriori label assigned we have a vector to represent the unknown state. Note that the apriori distribution can also be a true probability distribution, if we are initially not sure about the label, but have some partial knowledge.

Label Propagation Algorithm

$\boldsymbol{l}_{inferred}(v) := \boldsymbol{l}_{apriori}(v)$ for nodes with known labels,
otherwise $\boldsymbol{l}_{inferred}(v) := \boldsymbol{l}_{unknown}$

while not converged

$$\boldsymbol{l}_{inferred}(v) := \frac{1}{\deg(v)} \sum_{(v,w)\in E} \boldsymbol{l}_{inferred}(w)$$

$$\begin{aligned} \boldsymbol{l}_{inferred}(v) := \quad & p_v^{inj}\ \boldsymbol{l}_{apriori}(v) + && \text{// inject apriori knowledge} \\ & p_v^{con}\ \boldsymbol{l}_{inferred}(v) + && \text{// infer from neighbors} \\ & p_v^{aba}\ \boldsymbol{l}_{unknown} && \text{// abandon} \end{aligned}$$

The probabilities p_v^{inj}, p_v^{con} and p_v^{aba} can be interpreted as decisions in a random walk

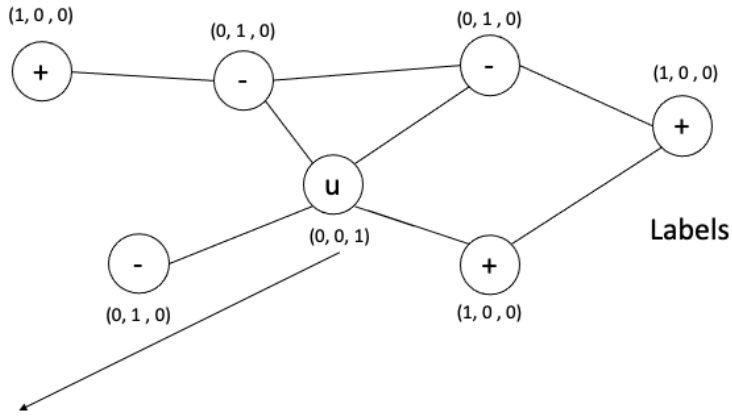
The full label propagation model adds additional aspects to the propagation of labels to neighbors. The probabilities should be understood as the decisions that a random walker can take to find a neighboring node from which to learn the label.

- For vertices with apriori labels, at every step the apriori distribution is injected with a certain probability p_v^{inj} that depends on the vertex
- For all vertices the propagation process (random walk) can also be abandoned with a certain probability p_v^{aba}. In that case the becomes unknown.
- The propagation of the label distribution to neighbors occurs then with a (remaining) probability of p_v^{con}.

As in PageRank the process is iterated till convergence occurs.

Example

Assume $p_v^{inj} = 1/2, p_v^{con} = 1/4, p_v^{aba} = 1/4$



$l_{inferred}$ = ½ (0, 0, 1) + ¼ (¼, ¾, 0) + ¼ (0, 0, 1) = (1/16, 3/16, 3/4)

Determining the Probabilities

The choice of the transition probabilities results in different variants of label propagation algorithms

Possible considerations

- pre-labelled nodes should have higher influence on neighbors than initially unlabeled nodes
- well-connected nodes should have higher influence than sparsely connected nodes

Example Model

$$c_v = \frac{\log 2}{\log(2+\deg(v))}$$

$d_v = (1 - c_v)\sqrt{H(v)}$, $H(v) = -\log \frac{1}{\deg(v)}$ if v is labelled,
$d_v = 0$ otherwise

$$z_v = \max(c_v + d_v, 1)$$

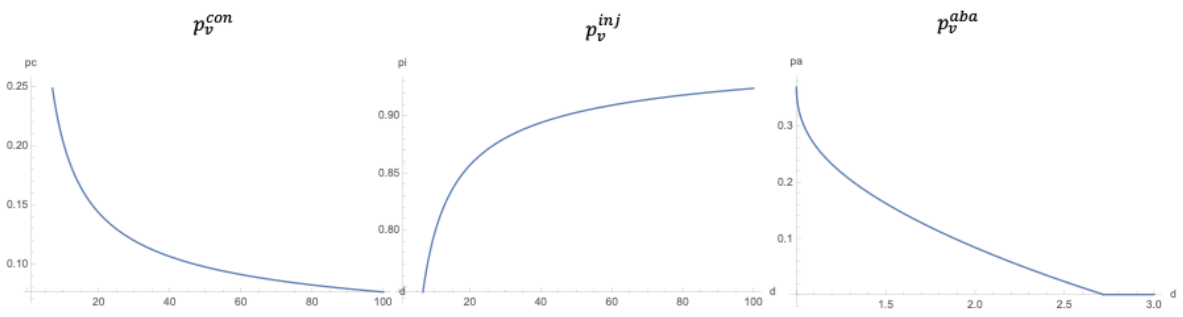
$p_v^{con} = \frac{c_v}{z_v}$, $p_v^{inj} = \frac{d_v}{z_v}$ for labelled nodes, 0 otherwise
$p_v^{aba} = 1 - p_v^{con} - p_v^{inj}$

The transition probabilities depend on the properties of the vertices. In our model, the only relevant property is its degree.

In a more general model with edge weights the probabilities would depend on the distribution of edge weights of the edges connected to the vertice (Fan-out entropy heuristics)

Behavior of Probabilities: Labelled Nodes

- Injection probability increases with the degree of the nodes, while continuation probability decreases
- Abandoning probability positive for only very low degree nodes (d = 1,2)

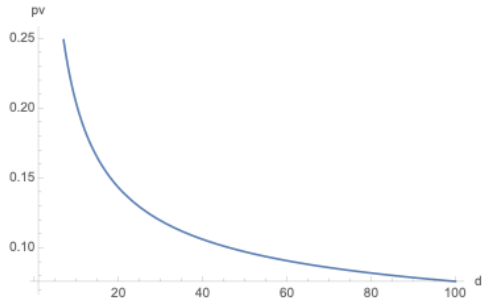


For labelled nodes the injection probability increases with the degree. Thus, well connected pre-labelled nodes have a lot of influence.

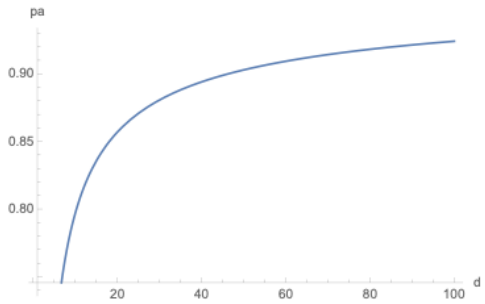
Behavior of Probabilities: Unlabelled Nodes

- Abandon probability increases with degree
- Prevents algorithm from propagating information through unlabeled, high-degree nodes

p_v^{con}



p_v^{aba}



For unlabeled nodes the behavior the abandon probability increases with the degree. Thus, high degree nodes have less influence.

Extensions

Label Propagation can be extended to

- A priori knowledge given as probability distribution
- Graphs with weighted edges
- Directed Graphs

Alternative algorithm: MAD (modified adsorption)

- Formulates an optimization problem and solves it directly
- Slightly better performance in practice

Discussion

Label propagation is an example of a **semi-supervised learning** algorithm

- Exploit partial labelling
- Useful in cases where labels are sparse or labels can produced only for special cases using heuristics or background knowledge
- Require that relationships among entities and their labels are correlated by some underlying principle

Different neighbors of a node v

A. Have a different influence depending on their degree
B. Have exactly same influence
C. Have a different influence depending on the degree of v
D. Have a different influence depending on whether they have a known label

The probabilities p_v^{inj}, p_v^{con} and p_v^{aba} depend on

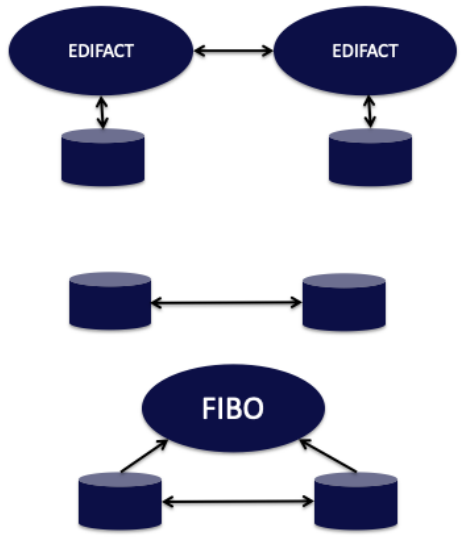
A. The node degree
B. The pre-existing knowledge on labels
C. On both node degree and pre-existing knowledge
D. On further factors

4.4.3 Schema Matching

1. Standardization
   - Mapping through standards

2. Mapping
   - Direct mapping

3. Ontologies
   - Mediated mapping



Conceptually there exist three principal approaches of how to address semantic heterogeneity. A first approach is to map all the models to one common global model. This approach is taken with standardization. For example, EDIFACT (https://www.unece.org/cefact/edifact/) is an international standard that models all concepts that are commonly used in business and trade. For exchanging information, the information systems map their data to EDIFACT and can thus share their data with other information systems.

A second approach consists of constructing directly a mapping among two information systems, without using any additional, shared knowledge in form of standards or ontologies

A third approach is to relate the model of an information system to a common model, frequently called ontology, and use this mapping to construct a direct mapping among the different models used in the information systems. . For example, in the financial domain there exists a standard reference ontology called FIBO (https://fib-dm.com/finance-ontology-transform-data-model/)

Direct Schema Mapping

Assume all data represented in canonical data model (e.g. relational)

- detect correspondences (schema matching)
- resolve conflicts
- integrate schemas (schema mapping)

Mappings are frequently expressed as queries (e.g. SQL Query)

Creating direct mappings among information systems has been widely applied for mapping data that is stored in relational, object-oriented and XML databases, where the model is represented as a database schema. The problem is known as the **schema mapping** problem. Given two schemas of databases, first schema elements are identified that are likely to correspond to each other, i.e., that are likely representing the same real-world aspect. This can be done by using many types of analysis using structural and content features of the database schema and the database. Tools for supporting these steps are called schema mapping tools. Once this step is performed some conflicting correspondences may occur, e.g., mapping some concept in one schema to two different ones in the other. These need to be resolved typically through decisions taken by humans. Once the correspondences are consistent, mappings can be automatically derived. The mappings are typically expressed as queries of the data manipulation language.

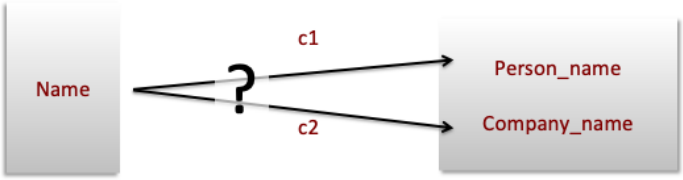
Schema Matching

Integration of heterogeneous data sources

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

How to find good “matches”?

How to choose the “best matches”?



Schema matching is a long standindg problem, both in industry and research.

Schema Matching Approaches

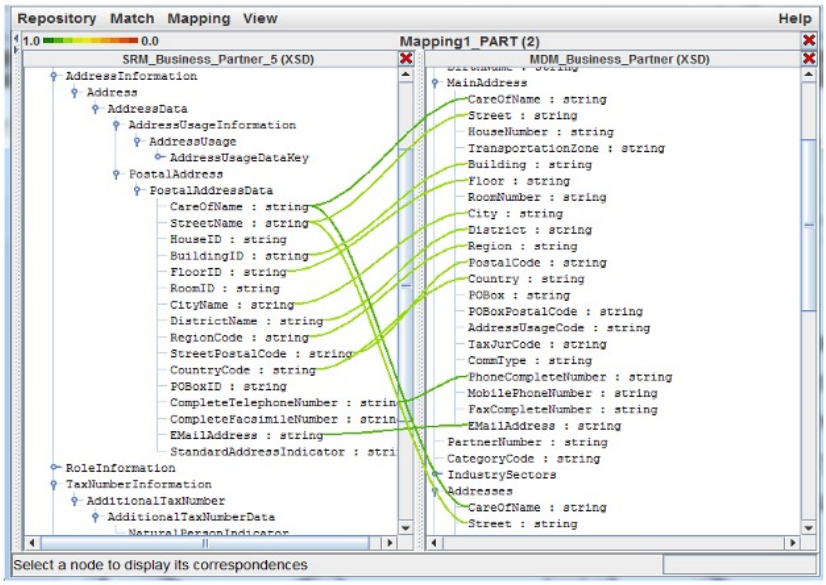
Manual matching

- 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
- Can now be implemented using LLMs

Schema Matching Tools



The Problem

Given two hierarchical knowledge graphs D1, D2

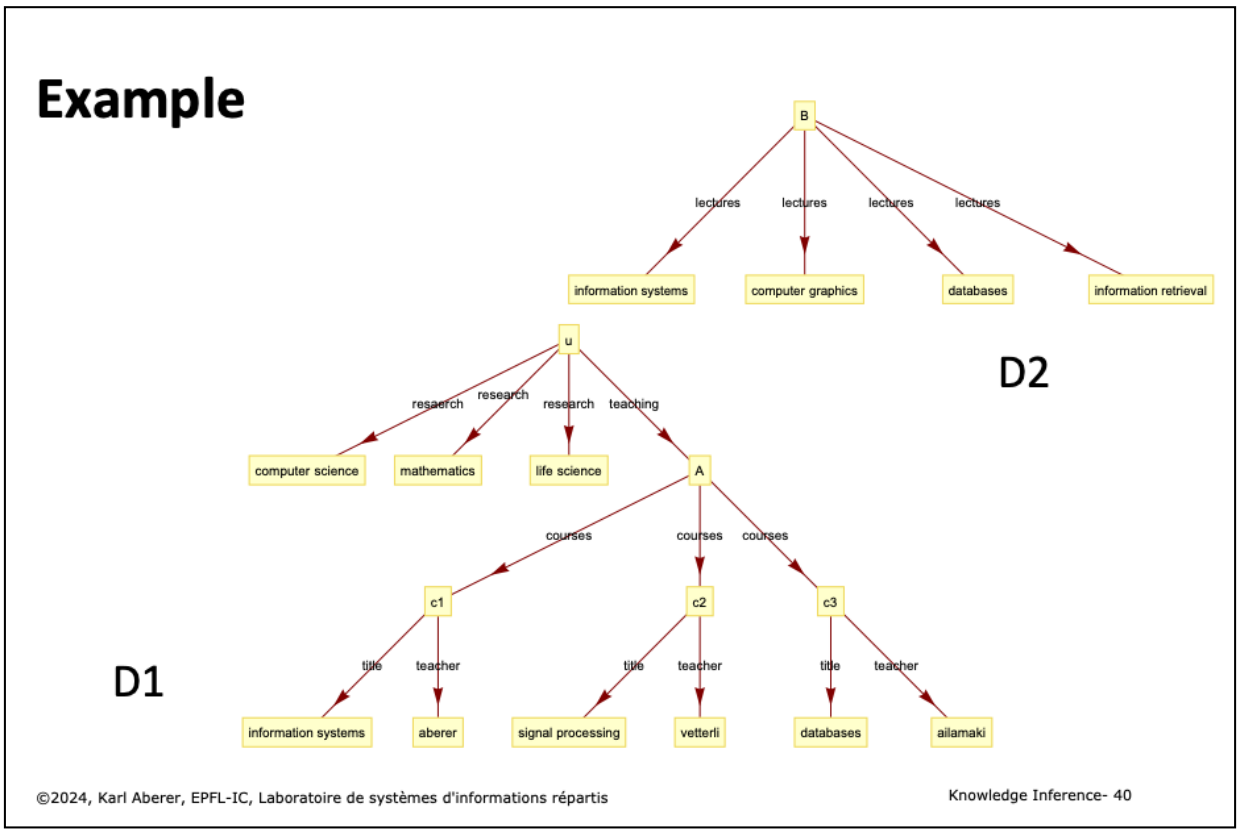
Goal: find a 1:1 matching of nodes (entities, classes) that have the same or similar "meaning"

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

In the following we will study the problem of integrating heterogeneous databases respectively knowledge graphs. We assume that we want to integrate data that are available in two knowledge graphs that are semantically related to each other.

We will make a few assumptions:

- the knowledge graph is hierarchical. It could also be a taxonomy.
- Some entities represent classes. We are in particular interested in matching those.
- Entities that represent classes share similar kinds of instances, which are entities.

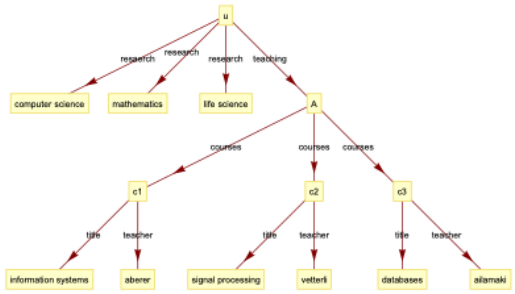


Let us assume that we have two example knowledge graphs. By inspecting the databases, a human can easily recognize that class A in graph D1 corresponds to class B in graph D2. The problem is how to perform the task of identifying this correspondence automatically.

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.

Similarity of Classes

A, B are classes, classes are subsets of U

Similarity measure (Jaccard similarity)

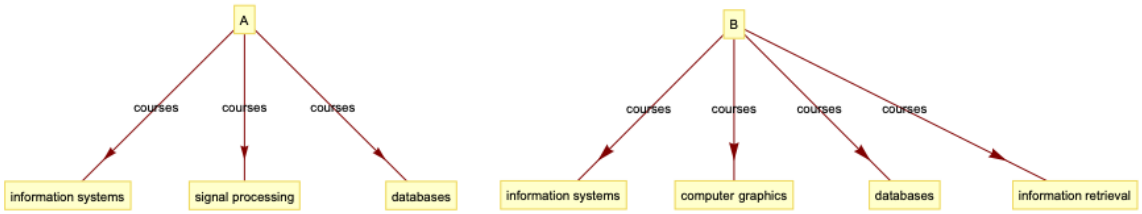
$$sim(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{P(x \in A \text{ and } x \in B)}{P(x \in A \text{ or } x \in B)} = \frac{P(A,B)}{P(A,B)+P(\bar{A},B)+P(A,\bar{B})}$$

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

The Jaccard measure is a standard approach to measure such a set similarity.

Example

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



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

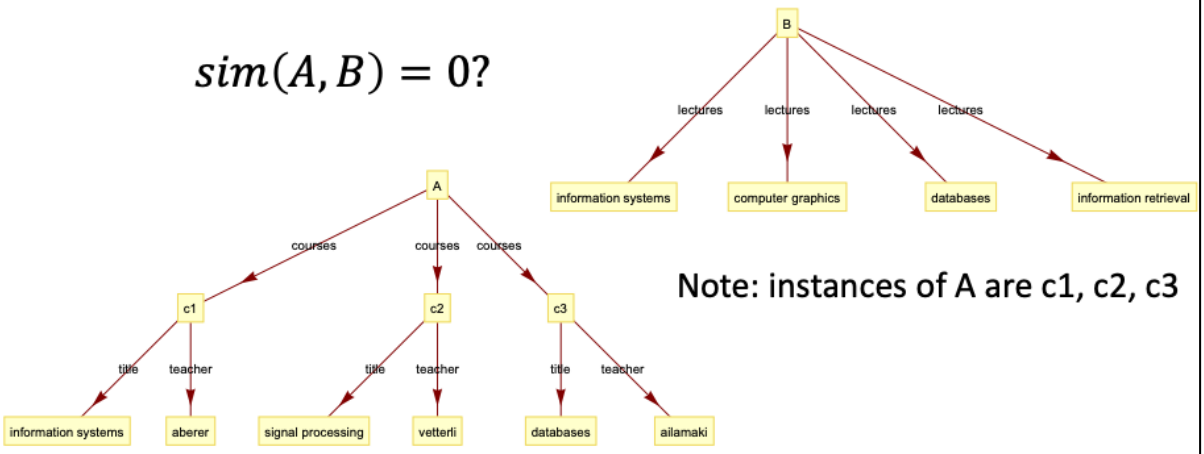
In this example the two classes have 2 common elements, and the number of all elements in their union is 5, thus the Jaccard similarity is 2/5.

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 knowledge graph. Therefore, it was straightforward to compare their instances directly. In a general knowledge graph, 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.

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 matching)

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

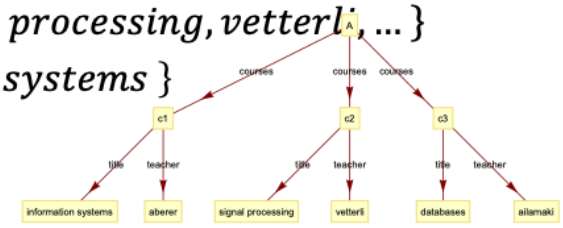
In database integration, many different features that can be associated with structural elements of a database have been considered for establishing semantic correspondences. For illustration, we will use in the following a simple feature: the set of atomic data values (strings) that can be reached from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

Content Feature of Classes

We consider as content feature, the set of terms associated with the leaf nodes that can be reached from a path starting at the instance

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

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



Another feature that can be exploited for this case would be the labels used along the paths. Note that the leaf nodes play a double role. They are instances, but at the same time their names serves as feature.

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 considering its features also $i \in B$, even if i is not part of $U2$

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

$P(A|T_i) = P(T_i|A)P(A)P(d) \propto P(T_i|A)P(A)$ (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 $A = \{c1, c2, c3\}$

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

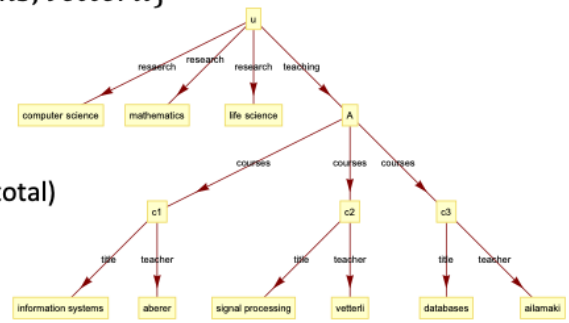
New instance cn: $T_{cn} = \{information\ systems, vetterli\}$

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

$P(T_{cn}|A) = \frac{1}{36}$, $P(T_{cn}|u) = \frac{1}{81}$

$P(A) = \frac{3}{13}$, $P(u) = \frac{4}{13}$ (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

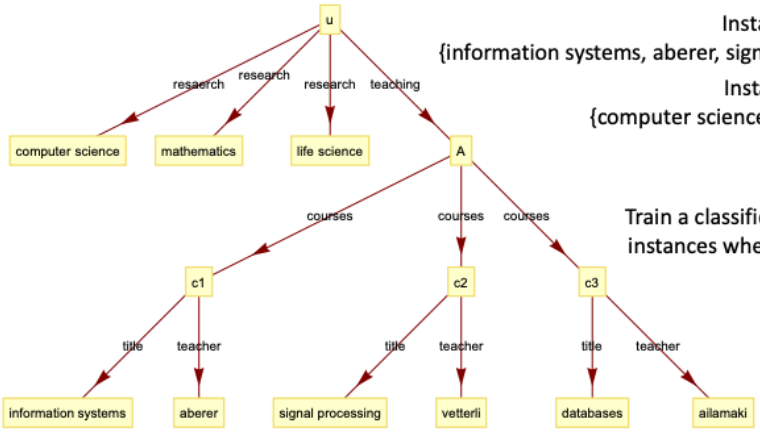
We show here a complete example of how for a new instance cn, the most likely corresponding class can be determined in the database. The Naïve Bayes classifier assigns a higher probability for cn to belong to A than to U (which coincides with our intuition).

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
   - Training set are instances in A and $\bar{A}$
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 $U2^{AB}$, the set of instances in $U2$ that could correspond to A
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.

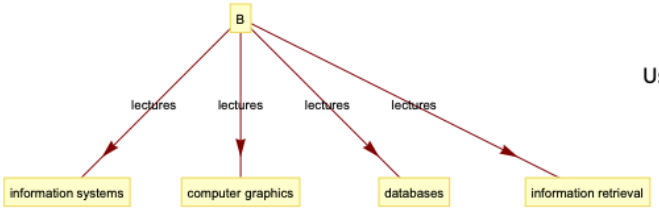
Example



Instances of $U1\mathrm{A}$
{information systems, aberer, signal processing, vetterli, databases, ailamaki}
Instances of $U1\bar{A}$
{computer science, mathematics, life science}

Train a classifier C_A to decide for a set of instances whether it belongs to A or not

Example



Instances of U2B
{information systems,computer graphics, databases, information retrieval}

Instances of $U1\bar{B}$
{}

Using classifier C_A : Instances of $U1AB$
{information systems, databases}

F Instances of $UA\bar{B}$
{}

Question

In the proposed schema matching approach, classifiers are constructed

1. for one schema using as training data the universe of the database
2. for each class of one schema using as training data the features of the instances of the class
3. for each class of both schemas using as training data the universe of the database
4. for each class of both schemas using as training data the features of all instances of the universe of the database

References

Course material based on

- Pershina, Maria, Yifan He, and Ralph Grishman. "Personalized page rank for named entity disambiguation." *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2015.
- Talukdar, Partha Pratim, and Koby Crammer. "New regularized algorithms for transductive learning." *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, Berlin, Heidelberg, 2009.
- Bordes, Antoine, et al. "Translating embeddings for modeling multi-relational data." Advances in neural information processing systems. 2013.
- Nguyen, Dat Quoc. "An overview of embedding models of entities and relationships for knowledge base completion." arXiv preprint arXiv:1703.08098 (2017).
- 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.