Knowledge Inference

1. Entity Disambiguation
2. Label Propagation
3. Link Prediction

Basic Questions in Knowledge Inference

Who are the entities?

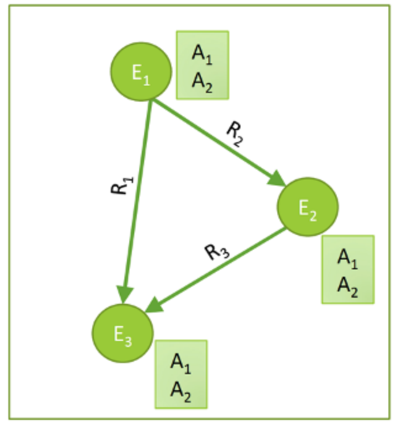
- Entity Linking / Disambiguation
- Data integration

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 many 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 bases 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.

1. ENTITY DISAMBIGUATION

Entity Disambiguation

Task: Link a text mention in a document to an entry in a knowledge base (e.g. WikiPedia or WikiData)

- Also called entity resolution and linking

Example: "Schindler is a Swiss industrial company. One of its main competitors is the American producer Otis."

Once entities in a text have been recognized, one would link them to their corresponding counter-parts in a knowledgebase. So entitiy disambiguation is a step that usually follows named entity recognition.

Challenge

Two problems

- Homonyms: entities with the same name
- Synonyms: different names for the same entity

Entity disambiguation can however be quite challenging to the homonymy and synonymy problem. Handling these problems is essential for every text analytics tasks. Not being able to handle homonymy usually results in the introduction of noise into the results (poor precision) whereas not properly handling synonymy risks to miss relevant documents (poor recall).

Sources of Information

Local information: similarity of a text mention of the entity and the data in the knowledge base

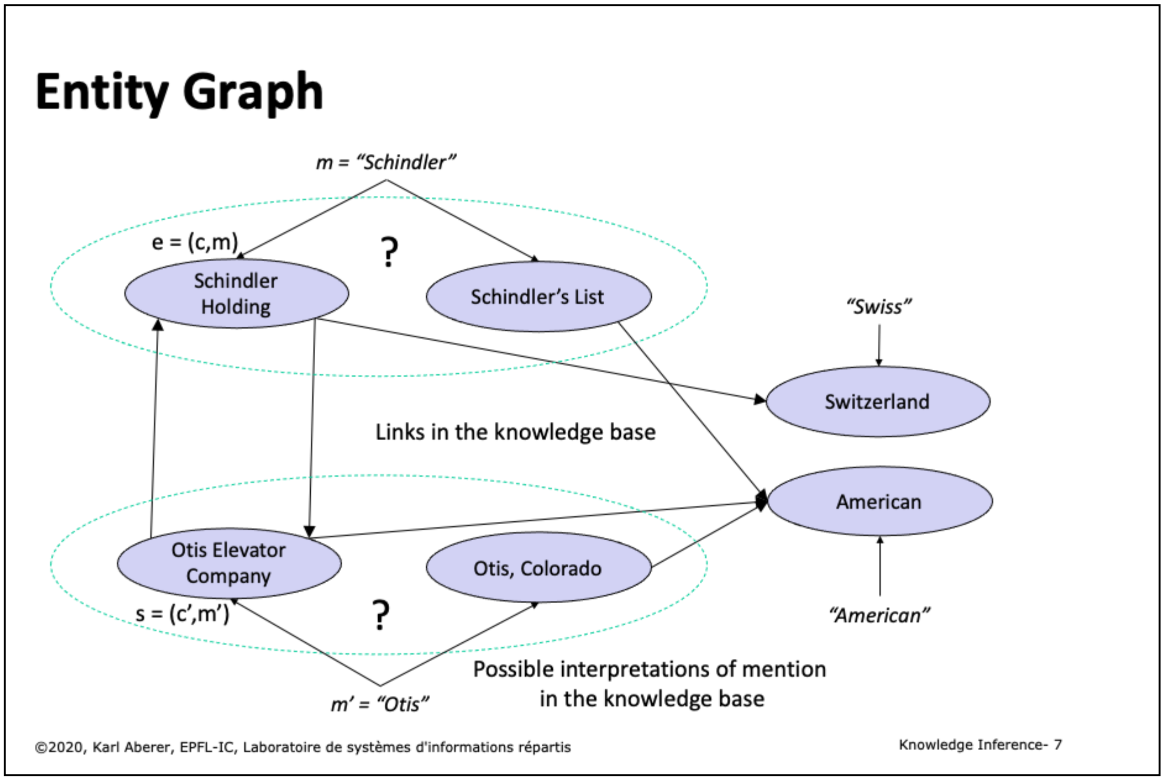
Example: "Schindler" ≈ "Schindler's list"

"Schindler" ≈ "Schindler Group"

Global information: coherence of the different text mentions of potential entities within a document with respect to a knowledge base

→ entity graph

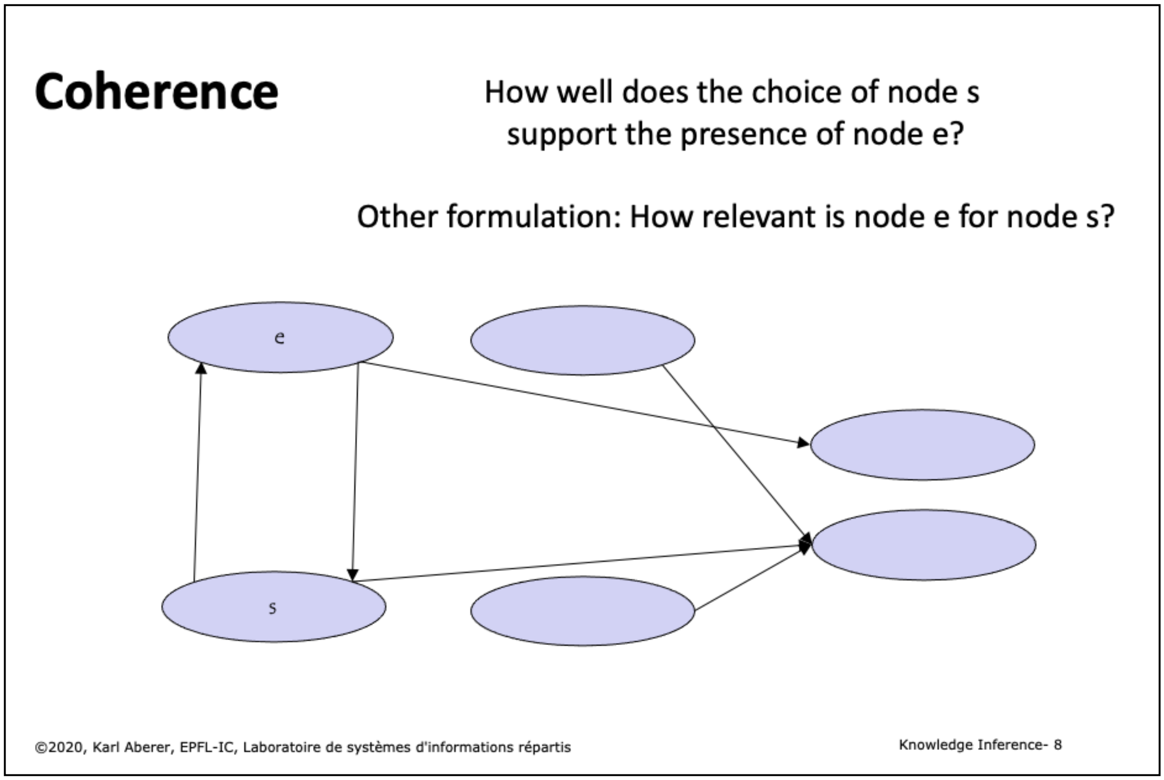
For performing entity disambiguation one can exploit two different sources of information. On the one hand local information extracted from the text mention, or its vicinity, can be exploited. This can be used to compare the text mention and its features with the representation in the knowledge base, in order to obtain evidence which entitities in the knowledge base are potentially matched. A second source of information is the knowledge base itself, when multiple entities are mentioned in the same text. Since these entities from the same text are likely to be in some form of relationship among each other, it is likely that such relationships are also discovered in the knowledge base. This can help in entity disambiguation.



Here we illustrate the basic model of how a knowledge base can be employed for entity disambiguation:

- m are text mentions of entities (extracted using NER)
- For each mention there are candidates in KB; these can be identified using local information from the text mention
- Each is a graph node; we can also associate a similarity measure to each node
- Edges between mention-candidate pairs are included if a link exists in the KB among the respective candidates

Since the same text mention can relate to several candidates, the problem of entity disambiguation is to determine which among the multiple candidates is the most likely to be correct.



To perform entity disambiguation, we pose the following question: given a node s in the entity graph, how well does it support the presence of another node e in the graph?

This question is actually related to a question that has been investigated in the context of personalized Web search: given a URL from the personal bookmark list of a user, how relevant is a page in the Web for this user. To answer this question a variation of the PageRank algorithm, which determines general relevance, has been proposed. It is called Personalized PageRank, which determines relevance with respect to a specific node in the graph.

Personalized PageRank

Same as PageRank, except that random jumps are always back to the same node (or same set of nodes)

- Original motivation: use personal bookmark list as source of rank

$$\vec{p_s} = c(qR \cdot \vec{p_s} + (1-q)\vec{e_s})$$
$$\vec{e_s} = (1,0,0, \ldots, 0), \text{ 1 at entry } s$$

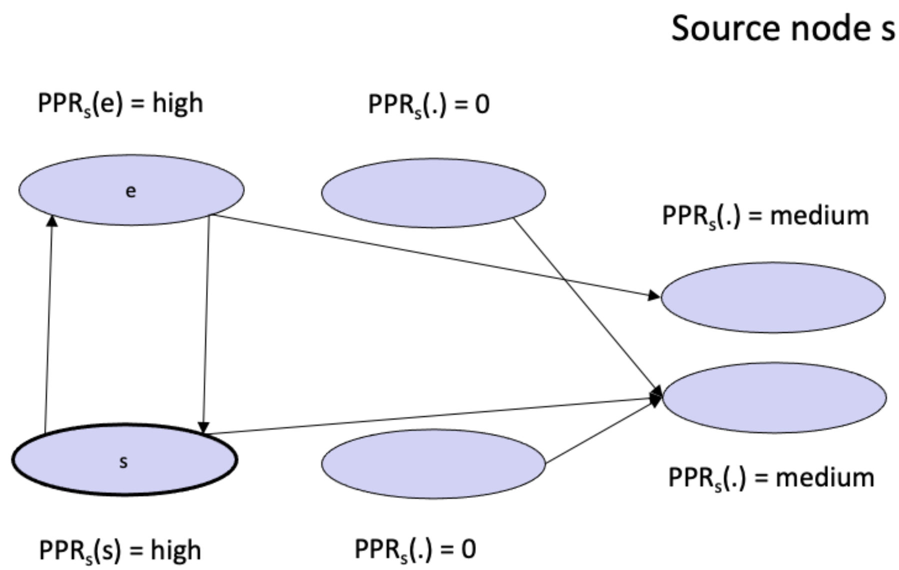
Can be computed using Monte Carlo method

- Perform multiple independent random walks
- Compute distribution of end points of random walks

Personalized PageRank works almost the same as the original PageRank. The difference is that random jumps are not performed uniformly random to any other node of the graph, but to a selected subset of nodes. In the case of Web search this subset would be the personal bookmark list, and by jumping back to this subset nodes from that list will have a large influence on the ranking of a page, such that nodes that are well connected to the bookmarks will receive more ranking value.

PPR can be computed either like standard PageRank, or using a Monte Carlo method, by starting random walks independently and aggregate the distribution of the end points of those walks.

PPR on Entity Graph



Knowledge Inference- 10

Now applying PPR, by considering a **source node s** as the selected subset of preferred nodes, will generate a distribution of ranking values for all other nodes. Nodes that are well connected to s will naturally receive higher ranking values. Intuitively it is clear, that in our example node e will be receive higher ranking when starting from s, and this be probably the preferred interpretation for the entity.

Approach

Finding the concept candidate linked to a mention m that is most likely to be valid

1. For all concept candidates c compute total support received from other nodes

$$e = (c, m), s = (c', m')$$
$$score(e) = \sum_{s \in Contributors_e} PPR_s(e)$$

2. Select the candidate with highest score

The procedure for selecting the best concept candidate for a text mention is straightforward. One computes the total support that the candidate receives from other nodes s, by computing the PPR of e with the source node s a selected node. Which source nodes s are contributing to this computation we need still to determine. At the end the candidate with the highest score is selected.

Contributing Nodes

Only one interpretation c for a mention m is valid

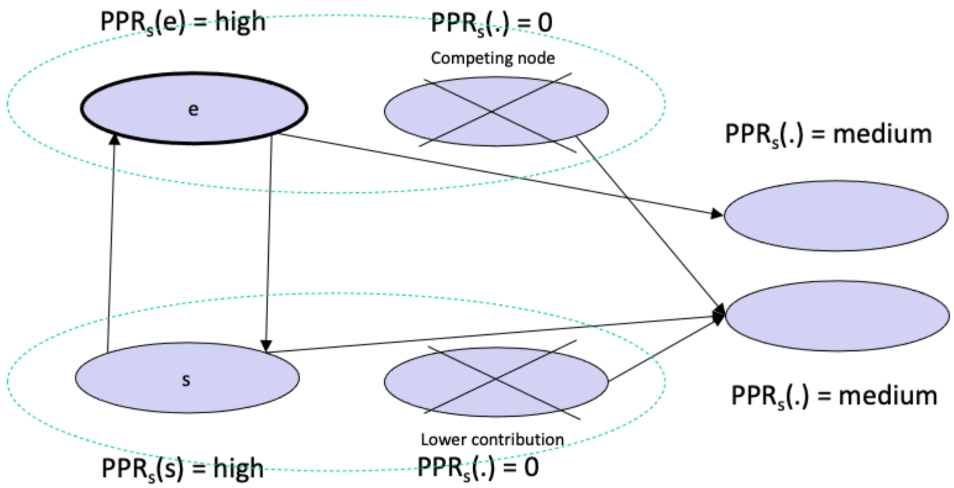
- Competing nodes $s = (c', m)$ that have the same entity mention as $e = (c, m)$ cannot support e
- For multiple nodes s that have the same entity mention m, only the one with highest contribution is considered

Thus

$$Contributors_e = \{(m', \operatorname*{argmax}_{c} PPR_{(c,m')}(e), m' \neq m)\}$$

Not every node should contribute as source node for a mention m, if we assume that only one interpretation can be correct. First, we exclude nodes that are “competing” for the interpretation of the text mention. Second for multiple nodes related to the some other entity mention, we select only the contribution of the one that has the highest value. In this way we favor a unique interpretation for mentions.

Example: Contributors_e



Considering Popularity

The method can furthermore consider popularity measures for nodes, e.g., it's degree

- If information is insufficient, favor popular nodes

$$score(e) = \underbrace{\sum_{s \in Contributors_e} PPR_s(e)pop(s)}_{\text{Promotes contributions from popular nodes}} + \underbrace{PPR_{avg}pop(e)}_{\text{Promotes popular nodes}}$$

To further improve the method, one can add a general popularity measure to weight the contributions of source nodes.This will favor popular nodes, which is beneficial if little information is available for disambiguation. Then it is better to choose a popular candidate since chances that this is correct are higher. One of possible choice of a popularity score could be the number of links a node has in the knowledge base.

Some Results

			Other methods		Uses pageRank	Without popularity		With popularity
Models	Cucerzan	Kulkarni	Hoffart	Shirakawa	Alhelbawy	iSim	PPR	PPRSim
Micro	51.03	72.87	81.82	82.29	87.59	62.61	85.56	91.77
Macro	43.74	76.74	81.91	83.02	84.19	72.21	85.86	89.89

Micro = fraction of correctly disambiguated entities
Macro = proportion of textual mentions, correctly disambiguated per entity, averaged over all entities

Experimental results show that the method works relatively will, with around 90% of entities that are correctly disambiguated. One can observe that the use of popularity helps to slightly improve the results.

Which is false?

A. Entity disambiguation addresses the problem of synonyms

B. Named entity recognition addresses the problem of synonyms

C. Entity disambiguation addresses the problem of entity classification

D. Named entity recognition addresses the problem of entity classification

Which nodes cannot contribute to the score of a mention linked to a concept?

A. Other concepts linked to the same mention
B. Concepts that have in the knowledge graph no outgoing links
C. Concepts that have in the knowledge graph no incoming links
D. Concepts with low popularity

2. LABEL PROPAGATION

Inferring Attribute Values

Example problem: 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

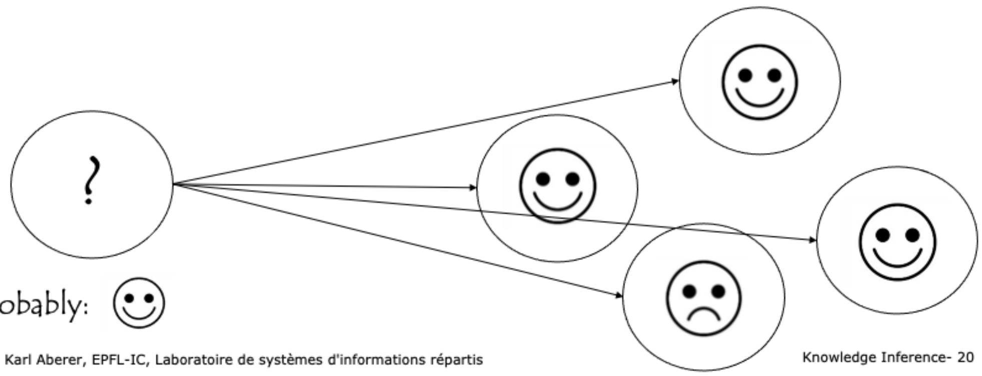
A second inference task in knowledge bases, after entities have been disambiguated, is to assign to the entities correct attribute values. For discrete attributes this problem an be understood as a classification problem. This question has, for example, been studied for classifying users in a social 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

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

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.

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.

Adding 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 $\boldsymbol{l}_{unknown}$

```
while not converged
```

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

$$\boldsymbol{l}_{inferred}(v) = \quad p_v^{inj}\ \boldsymbol{l}_{apriori}(v) + \qquad \text{// inject apriori knowledge}$$
$$p_v^{con}\ \boldsymbol{l}_{inferred}(v) + \qquad \text{// infer from neighbors}$$
$$p_v^{aba}\ \boldsymbol{l}_{unknown} \qquad \text{// stop}$$

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.

- For vertices with apriori labels, at every step the apriori distribution is injected with a certain probability p_v^{inj} that depends on the vertice
- For all vertices the propogation process can also be abandoned with a certain probability p_v^{aba}
- 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.

Determining the Probabilities

Entropy of transition probabilities $H(v) = -\log \frac{1}{\deg(v)}$

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

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

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

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

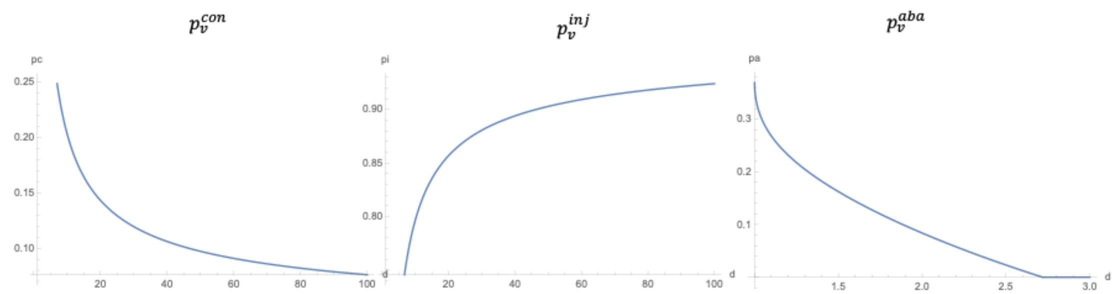
$$p_v^{aba} = 1 - p_v^{cont} - p_v^{inj}$$

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)

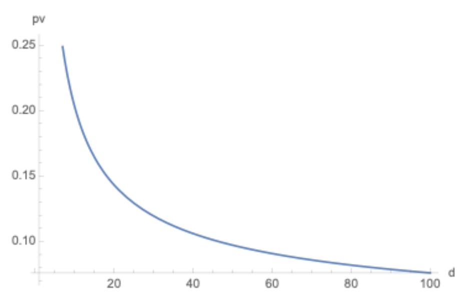


For labelled nodes the injection probability increases with the degree. Thus, well connected pre-labelled nodes have a lot of influence.

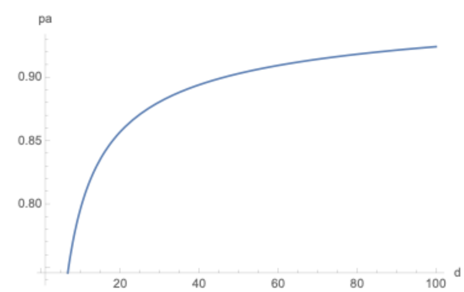
Behavior of Probabilities: Unlabelled Nodes

- 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

3. LINK PREDICTION

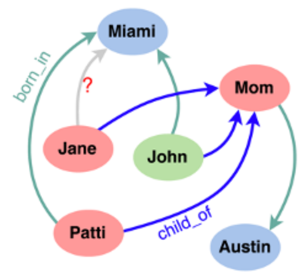
Knowledge Base Completion

Large knowledge bases are usually incomplete

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

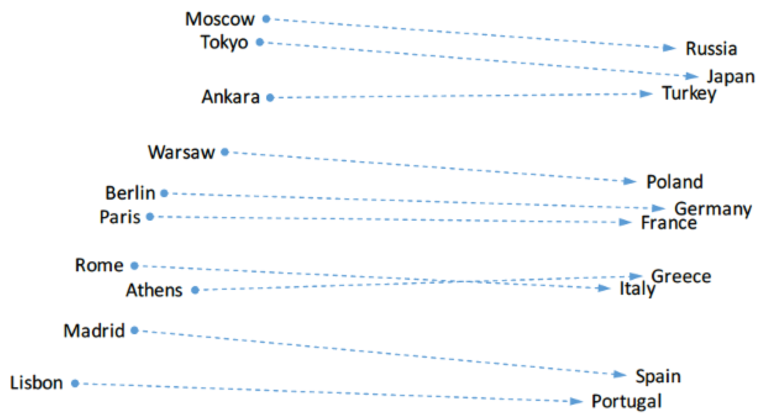
Try to predict missing links from existing data

Jane born in Miami?



In general, knowledge bases are incomplete. Thus there is a significant interest in completing in particular 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 this problem, we come back to an observation that we had made for word embeddings. We found that (in some cases) relationships seem to be encoded 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!

This observation can be formally described in two different ways. We could say, that the differences of entity vectors that are in a relationship should result in similar values. Alternatively, we sould also say, that there must be some vector that represents the relationship and that the sum of this vector with the difference vector of the entities should be approximately zero. This second formulation of the properties gives rise to an idea of how link prediction could be performed using and embedding technique.

Model

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

Define a plausibility score $\mathrm{f}(h, r, t)$ such that

$$\mathrm{f}(h, r, t) > \mathrm{f}(h', r', t')$$

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

To formulate the model we assume that we have a knowledge graph consisting of triples (h, r, t). h indicates the term "head" and t the term "tail".

Then we introduce a function that measure how plausible a triple is (we can think of it as a probability). Clearly, the function should return higher values for plausible tuples than for implausible ones.

Learning the Model

Minimize the loss function

$$J(\theta) = \sum_{\substack{(h,r,t)\epsilon G \\ (h',r',t')\epsilon G'(h,r,t)}} \max(0, \gamma + \mathrm{f}(h,r,t) - \mathrm{f}(h',r',t'))$$

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

γ is a hyperparameter

Using the plausibility function, we can formulate a loss function that should be minimized. The minimization will be performed as usual using SGD.

TransE Model

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

- Based on the intuition from text WE

$$\mathrm{f}(h, r, t) = \|\boldsymbol{v}_h + \boldsymbol{v}_r - \boldsymbol{v}_t\|_{1/2}$$

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

With the generic model we can instantiate difference concrete approaches for link prediction, depending on the choice of f. One of the initial approaches that was strongly inspired by our initial observation on word embeddings from the beginning. It directly introduces a vector $\boldsymbol{v}_r$ that represents the linear transformation corresponding to a relationship, and mapping the head and tail into the same (low-dimensional) vector space.

The mappings from the entities and relationships to the latent space are performed (as in word embeddings) using embedding matrices, which constitute the model parameters that have to be learnt using SGD.

Performing SGD

Initialize vectors with random values

- From interval $[-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}}]$ 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). 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

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 predections. However, given that the knowledge bases used for evaluation are incomplete, it is not out of question that also other predictions are meaningful.

Alternative Models

Model	Score function $f(h,r,t)$	Opt.
Unstructured	$\lVert \boldsymbol{v}_h - \boldsymbol{v}_t \rVert_{\ell_{1/2}}$	SGD
SE	$\lVert \mathbf{W}_{r,1}\boldsymbol{v}_h - \mathbf{W}_{r,2}\boldsymbol{v}_t \rVert_{\ell_{1/2}}$; $\mathbf{W}_{r,1}, \mathbf{W}_{r,2} \in \mathbb{R}^{k\times k}$	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	$\lVert \boldsymbol{v}_h + \boldsymbol{v}_r - \boldsymbol{v}_t \rVert_{\ell_{1/2}}$; $\boldsymbol{v}_r \in \mathbb{R}^k$	SGD
TransH	$\lVert (\mathbf{I} - \boldsymbol{r}_p\boldsymbol{r}_p^\top)\boldsymbol{v}_h + \boldsymbol{v}_r - (\mathbf{I} - \boldsymbol{r}_p\boldsymbol{r}_p^\top)\boldsymbol{v}_t \rVert_{\ell_{1/2}}$ $\boldsymbol{r}_p, \boldsymbol{v}_r \in \mathbb{R}^k$; $\mathbf{I}$: Identity matrix size $k \times k$	SGD
TransR	$\lVert \mathbf{W}_r\boldsymbol{v}_h + \boldsymbol{v}_r - \mathbf{W}_r\boldsymbol{v}_t \rVert_{\ell_{1/2}}$; $\mathbf{W}_r \in \mathbb{R}^{n\times k}$; $\boldsymbol{v}_r \in \mathbb{R}^n$	SGD
TransD	$\lVert (\mathbf{I} + \boldsymbol{r}_p\boldsymbol{h}_p^\top)\boldsymbol{v}_h + \boldsymbol{v}_r - (\mathbf{I} + \boldsymbol{r}_p\boldsymbol{t}_p^\top)\boldsymbol{v}_t \rVert_{\ell_{1/2}}$ $\boldsymbol{r}_p, \boldsymbol{v}_r \in \mathbb{R}^n$; $\boldsymbol{h}_p, \boldsymbol{t}_p \in \mathbb{R}^k$; $\mathbf{I}$: Identity matrix size $n \times k$	AdaDelta
lppTransD	$\lVert (\mathbf{I} + \boldsymbol{r}_{p,1}\boldsymbol{h}_p^\top)\boldsymbol{v}_h + \boldsymbol{v}_r - (\mathbf{I} + \boldsymbol{r}_{p,2}\boldsymbol{t}_p^\top)\boldsymbol{v}_t \rVert_{\ell_{1/2}}$ $\boldsymbol{r}_{p,1}, \boldsymbol{r}_{p,2}, \boldsymbol{v}_r \in \mathbb{R}^n$; $\boldsymbol{h}_p, \boldsymbol{t}_p \in \mathbb{R}^k$; $\mathbf{I}$: Identity matrix size $n \times k$	SGD
STransE	$\lVert \mathbf{W}_{r,1}\boldsymbol{v}_h + \boldsymbol{v}_r - \mathbf{W}_{r,2}\boldsymbol{v}_t \rVert_{\ell_{1/2}}$; $\mathbf{W}_{r,1}, \mathbf{W}_{r,2} \in \mathbb{R}^{k\times k}$; $\boldsymbol{v}_r \in \mathbb{R}^k$	SGD
TranSparse	$\lVert \mathbf{W}_r^h(\theta_r^h)\boldsymbol{v}_h + \boldsymbol{v}_r - \mathbf{W}_r^t(\theta_r^t)\boldsymbol{v}_t \rVert_{\ell_{1/2}}$; $\mathbf{W}_r^h, \mathbf{W}_r^t \in \mathbb{R}^{n\times k}$; $\theta_r^h, \theta_r^t \in \mathbb{R}$; $\boldsymbol{v}_r \in \mathbb{R}^n$	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	$\lVert \boldsymbol{v}_h + \boldsymbol{v}_{r_1} + \boldsymbol{v}_{r_2} + ... + \boldsymbol{v}_{r_m} - \boldsymbol{v}_t \rVert_{\ell_{1/2}}$; $\boldsymbol{v}_{r_1}, \boldsymbol{v}_{r_2}, ..., \boldsymbol{v}_{r_m} \in \mathbb{R}^k$	AdaGrad
ConvE	$\boldsymbol{v}_t^\top g\left(\text{vec}\left(g\left(\text{concat}(\overline{\boldsymbol{v}}_h, \overline{\boldsymbol{v}}_r) * \boldsymbol{\Omega}\right)\right)\boldsymbol{W}\right)$; 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 very many methods that have been in the meanwhile proposed to tackle the link prediction problem.

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

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.