Ranking of Keyword-Based Search Query Results in Knowledge Graphs

Bachelor thesis

Antonio Noack

31.08.2020

DLR

FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

Overview

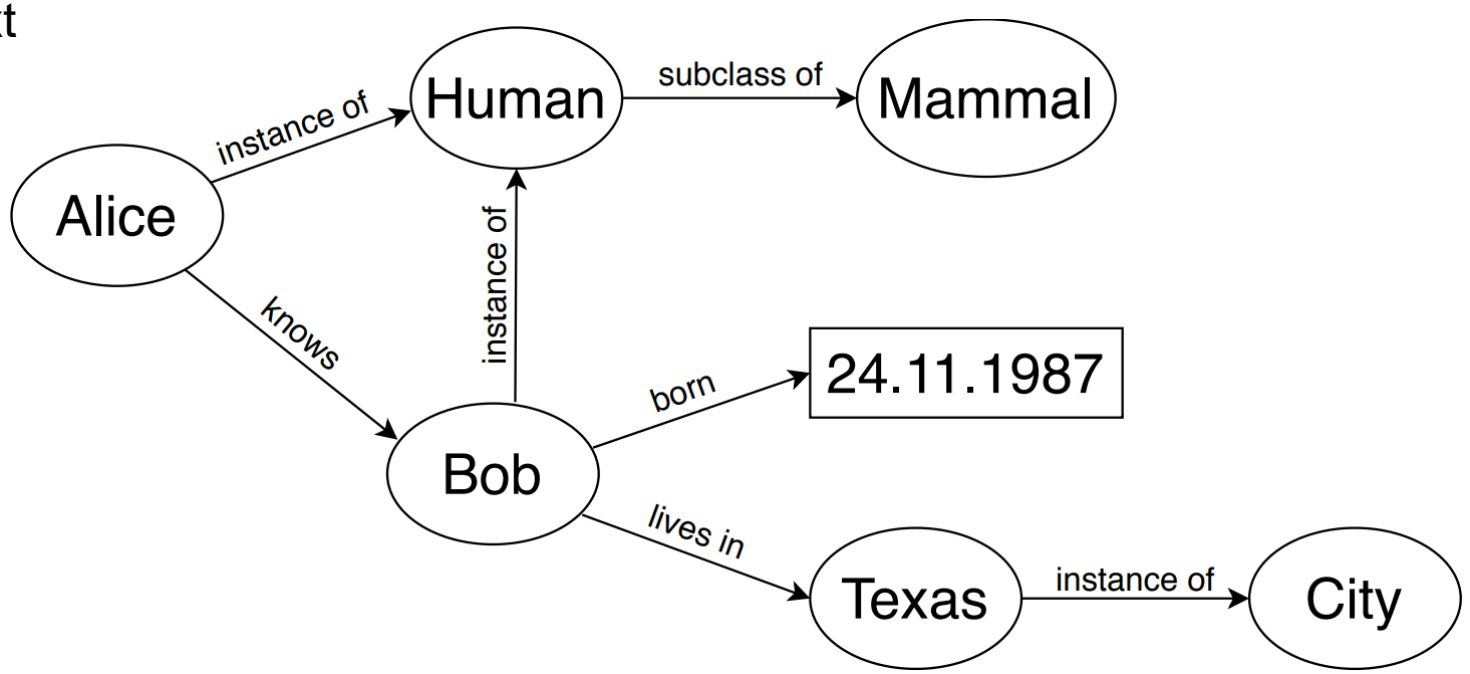
- Basics
- Related Work
- Concepts
- Conclusion
- Future Work

Knowledge Graphs

- Model knowledge as a graph instead of text documents
- Examples: WikiData, DBpedia
- Can be represented as

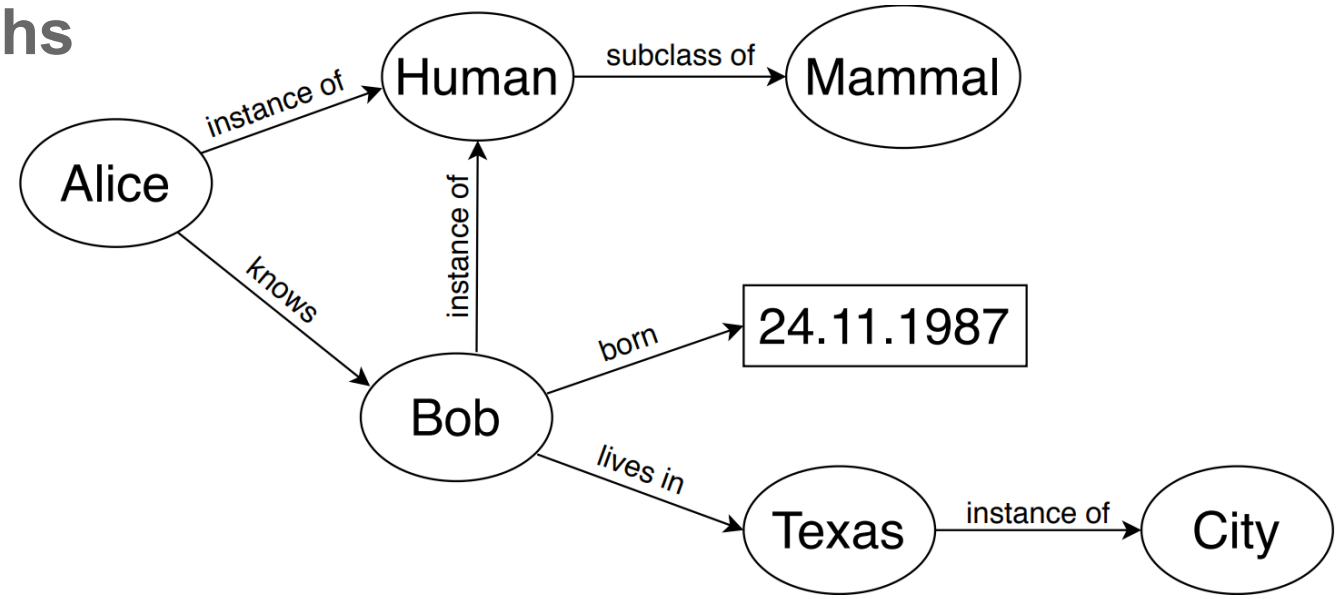
  Resource **D**escription **F**ramework or

  Labeled **P**roperty **G**raphs



SPARQL – Query Knowledge Graphs

- Query language like SQL
- For RDF graphs
- Select query:
  - list of statements on relations
  - partially variable entities inside
- Result:
  - set bindings of entities to variables




```
SELECT ?a ?b
WHERE {
  ?a ex:knows ?b .
  ?b ex:knows ?a .
  ?b ex:livesIn ex:Texas .
}
```

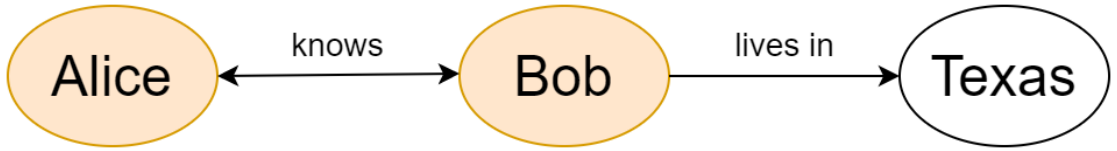
Result Subgraphs

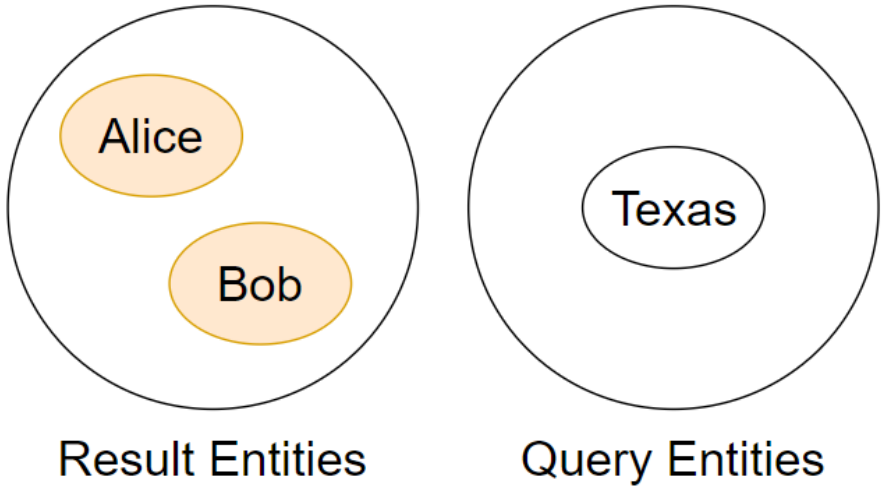
- Query + binding → subgraph
- Entities split into result entities and query entities

```
SELECT ?a ?b
WHERE {
  ?a ex:knows ?b .
  ?b ex:knows ?a .
  ?b ex:livesIn ex:Texas .
}
```

?a = Alice

?b = Bob

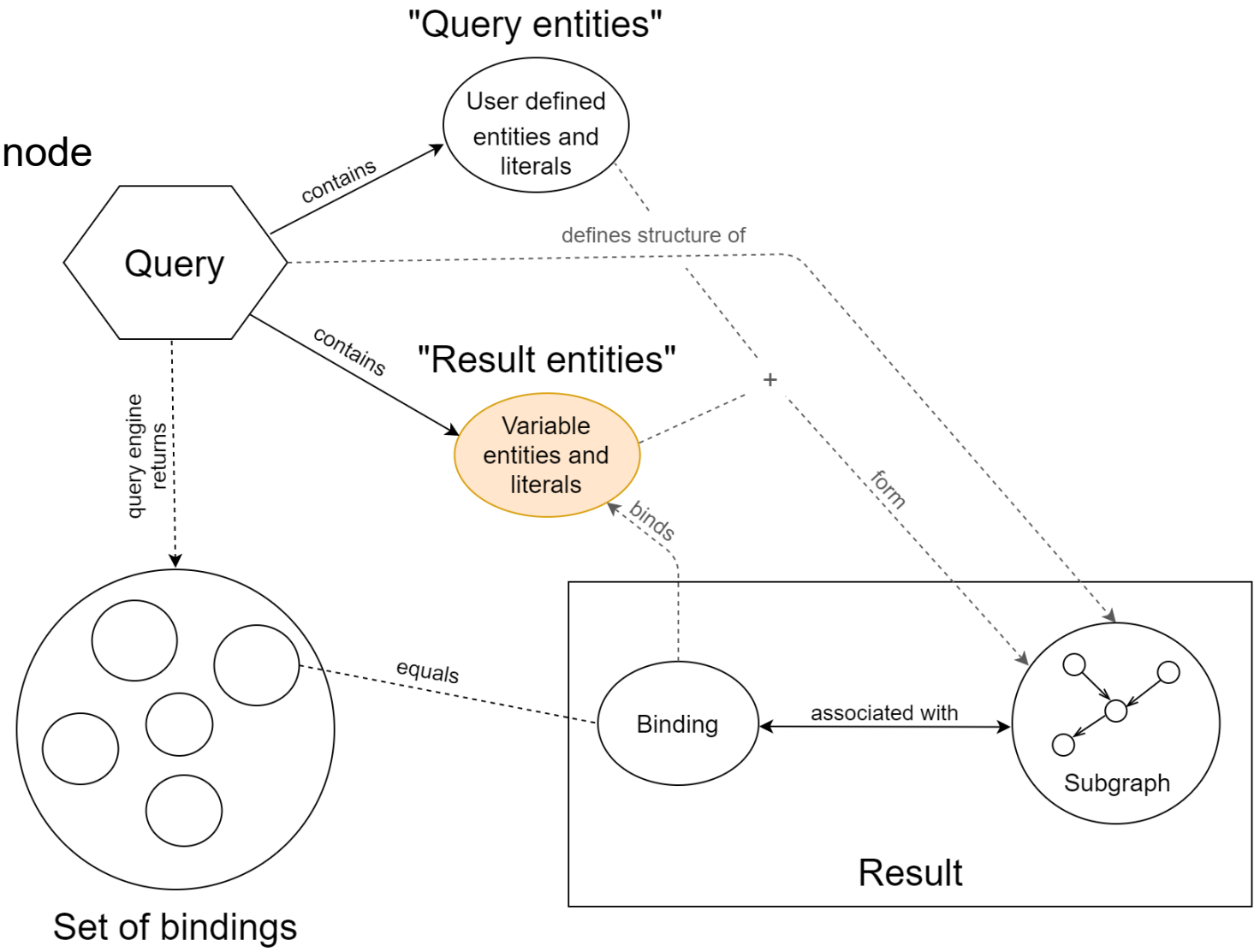




Thesis Scope and Overview

- For keyword-based search only one result node
- Query can return many results

  → rank them



Related Work

- What have others done to rank?
- How to compare rankings?

→ then we can propose our own ranking

Term Frequency – Inverse Document Frequency

- Count weighted word matches
- Prefer high occurrences of words (TF)
- Rare words more valuable than very common words (IDF)

  e.g. „the“, „a“, „knowledge“, „frequency“

- Sum TF-IDF for word groups
- Ambiguous words problematic

  e.g. break (pause/shatter)
- Synonyms are not matched

  e.g. speech and talk

Word Embeddings

- Words as vectors for machine learning
- Semantic meaning

  e.g. King + (Woman – Man) ≈ Queen
- Typically 300-1000 dimensions for a large dictionary
- Word2Vec, trained by adjacent words
- RDF2Vec for KGs
- Cosine Similarity

$$\overrightarrow{WE}(Dog) = [3, 1, 0]$$
$$\overrightarrow{WE}(Cat) = [3, 5, 0]$$
$$\overrightarrow{WE}(Wolf) = [3, 2, 1]$$
$$\overrightarrow{nWE}(Dog) = \frac{\overrightarrow{WE}(Dog)}{|\overrightarrow{WE}(Dog)|} \approx [0.95, 0.32, 0.00]$$
$$\overrightarrow{nWE}(Cat) \approx [0.51, 0.86, 0.00]$$
$$\overrightarrow{nWE}(Wolf) \approx [0.80, 0.53, 0.27]$$
$$Cosine\ Similarity(Cat, Dog) = \overrightarrow{nWE}(Cat) \cdot \overrightarrow{nWE}(Dog) \approx 0.76$$
$$Cosine\ Similarity(Wolf, Dog) \approx 0.93$$

PageRank & HITS

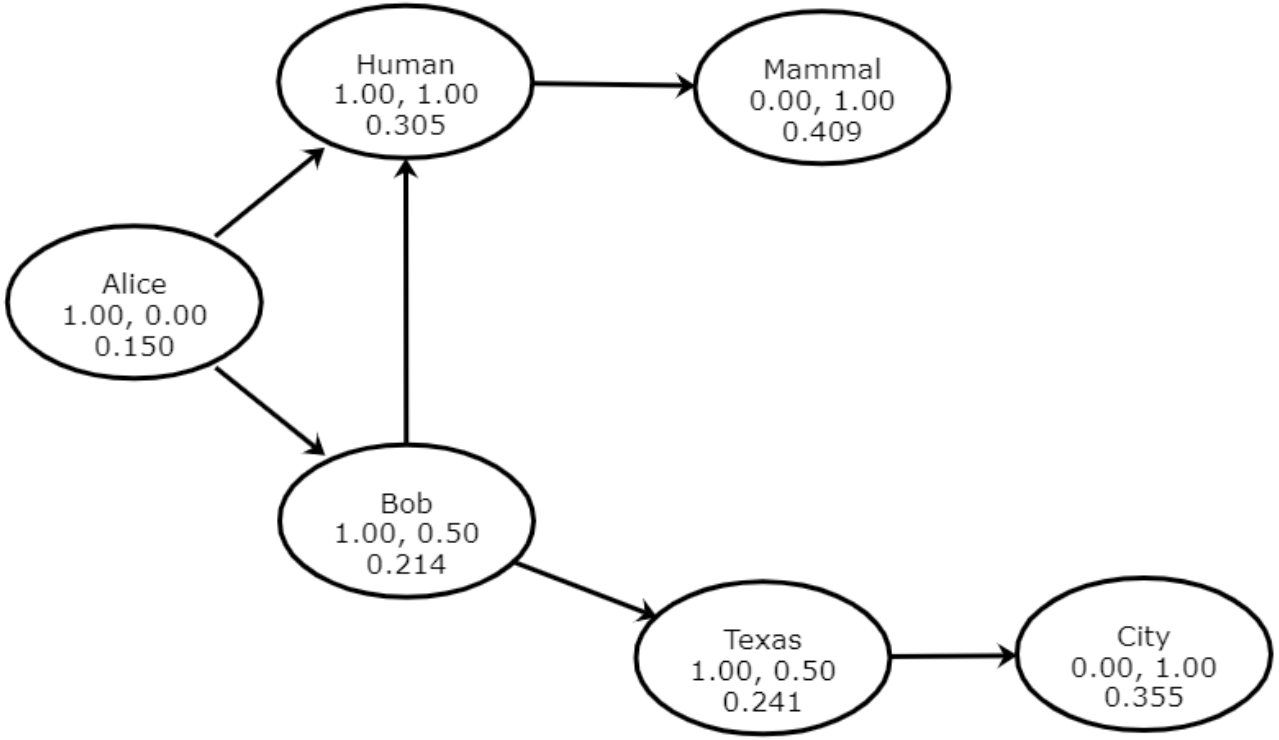
- Random surfer model:

  - Clicks pages randomly

  - Gets randomly bored and leaves

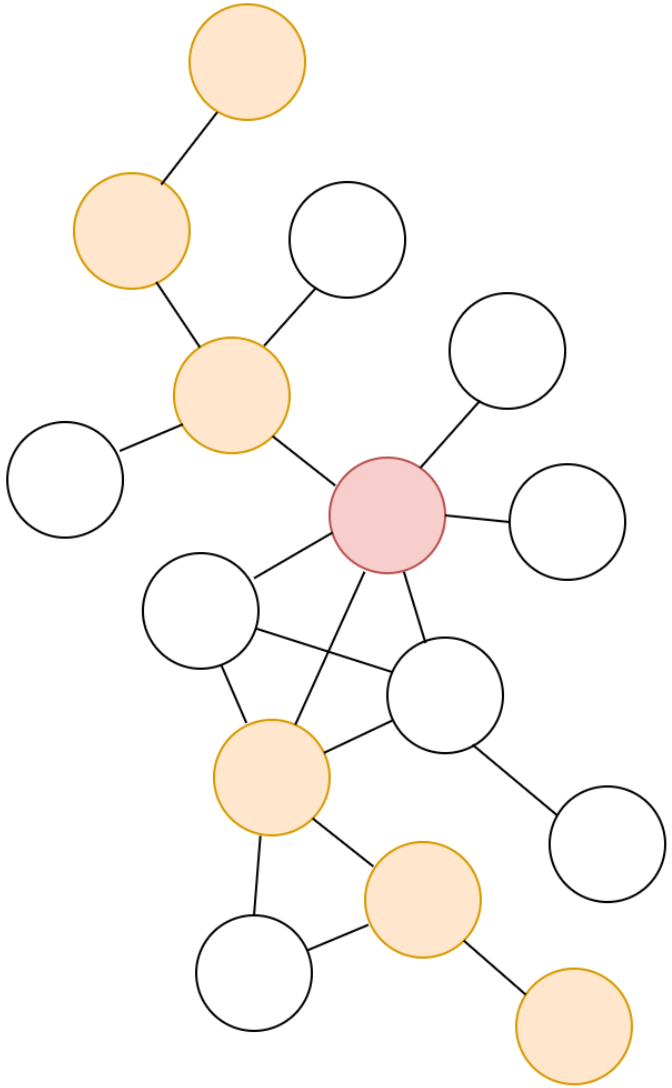
- Customizable by changing “boredom” probability
- Iterative calculation on whole graph, converges quickly
  → Pre-calculated

- HITS similar: authority transfer instead of surfers
- Two scores: hubs and authorities



Knowledge Graph-Measures

- Radius: $\min_{v_1 \in V} \max_{v_2 \in V} Distance(v_1, v_2)$
- Diameter: $\max_{v_1 \in V} \max_{v_2 \in V} Distance(v_1, v_2)$
- Centralities for importance of nodes, e.g. PageRank



Learning To Rank – Machine Learning

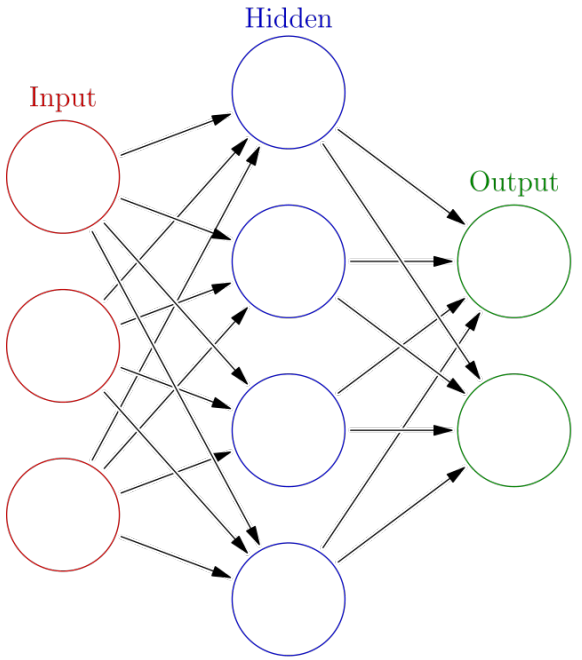
- Applies machine learning to ranking
- Used by Google-Search and IBM-Watson
- Intuition:

  - measures become features

  - features get assigned weights (linear model)

  depending on their importance for a ranking
- Needs lots of high quality relevance data by humans
- Model needs to be decided

  → overfitting / underfitting issue
- Needs optimization function



Evaluation Metrics

- To compare rankings or models with parameters
- Needs relevance data
- Hits@N

  - percentage of relevant results
- NDCG@N:

  - includes rank-depending importance

  - normalized for results with few relevant nodes

Concepts

- Proposing our own method
- LTR-based to get the best from all worlds

String Classes

- For text document based approaches
- Three classes on nodes for different closeness:
  - String Class 1: labels + descriptions (rdf:label, rdf:comment)
  - String Class 2: all literals as strings
  - String Class 3: all literals + adjacent node literals

TFScore

- Like text search: find word/term matches between query and result (nodes)
- Sum over all matching terms
- Function for matching? Combine(TF_Query, TF_Result)
- For each String Class individually

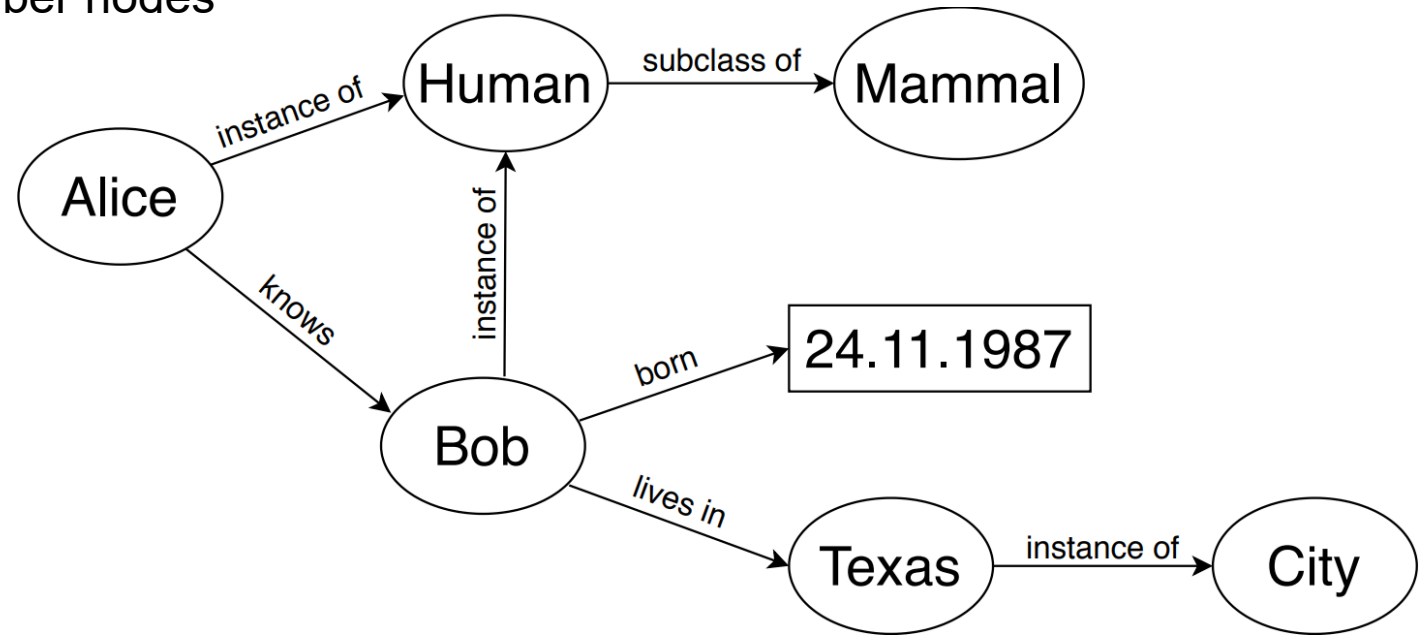
$$TFScore(QN, RN, SC) = \frac{1}{|SC(SN)|} \sum_{w \in SC(SN)} IDF(w) \cdot combine\Big(TF\Big(SC(QN), w\Big), TF\Big(SC(RN), w\Big)\Big)$$

$$IDF(w, KG) = ln\left(\frac{|nodes(KG)| + 1}{|\{n \in nodes(KG), w \in SC2(n)\}| + 1}\right)$$

KG-based features

- Models popularity or importance
- PageRank, HITS, in-degree and out-degree
- Result nodes and query nodes could be relevant

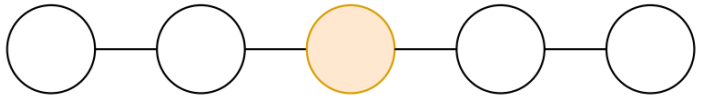
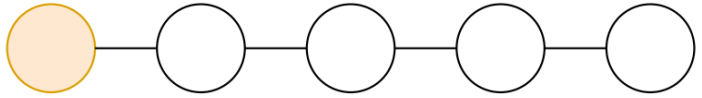
  → use a feature for both, averaged over member nodes



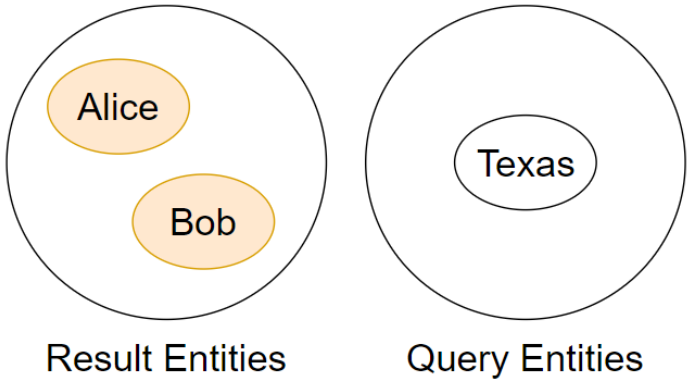
Subgraph-Measures

- Concentrated Graphs may be better results
- Measure this using radius, diameter, DistanceScore

$$DistanceScore(G) = \sum_{q \in QN} \sum_{r \in RN} Distance(q, r)$$

Subgraph	DistanceScore
A 	2 + 1 + 1 + 2 = 6
B 	1 + 2 + 3 + 4 = 10

Embeddings



- Generate entity embeddings
- Average them into entity group embeddings
- Features: similarity between result entities and query entities
- Embeddings can be created from words (Word2Vec) or from entities (RDF2Vec)
- Description-length-issue: use on all string classes
- Name-issue: use RDF2Vec as well

Domain-Specific Properties

- Use-case specific
- If a value is generally preferred
- Numerical: e.g. price, area coverage, age, …
- Categorical: e.g. type of product, class
- Categories can be represented to ML-model by vector

Conclusion

- Signals are mainly based on subgraph-structure, embeddings, TF-IDF and KG
- LTR model to find best combination of relevance signals

Group	Feature
KG	Average In-Degree QN
KG	Average In-Degree RN
KG	Average Out-Degree QN
KG	Average Out-Degree RN
KG	Average PageRank QN
KG	Average PageRank RN
KG	Average HITS-Hub-Score QN
KG	Average HITS-Hub-Score RN
KG	Average HITS-Authority-Score QN
KG	Average HITS-Authority-Score RN
Subgraph	Subgraph Radius
Subgraph	Subgraph Diameter
Subgraph	DistanceScore
TF-IDF	$TFScore(QN, RN, SC1)$
TF-IDF	$TFScore(QN, RN, SC2)$
TF-IDF	$TFScore(QN, RN, SC3)$
Embeddings	$CosineSimilarity\left(\overrightarrow{NGE}(SC1, QN), \overrightarrow{NGE}(SC1, RN)\right)$
Embeddings	$CosineSimilarity\left(\overrightarrow{NGE}(SC2, QN), \overrightarrow{NGE}(SC2, RN)\right)$
Embeddings	$CosineSimilarity\left(\overrightarrow{NGE}(SC3, QN), \overrightarrow{NGE}(SC3, RN)\right)$
Embeddings	$CosineSimilarity\left(\frac{1}{\lvert QN\rvert}\sum_{q\in QN} RDF2Vec(q), \frac{1}{\lvert RN\rvert}\sum_{r\in RN} RDF2Vec(r)\right)$
Domain-Specific	*Domain-specific*

Future Work

- Collect training data for use-case
- Decide about additional, useful numerical/categorical features
- Implement the system / feed the data into LTRHits@10

  - could be attempted on selected features and expanded later
- Evaluate results using NDCG / Hits@10

Thank you for your attention

Sources:

- LTR image: https://en.wikipedia.org/wiki/File:Colored_neural_network.svg,
  Creative Commons Attribution-Share Alike 3.0 Unported