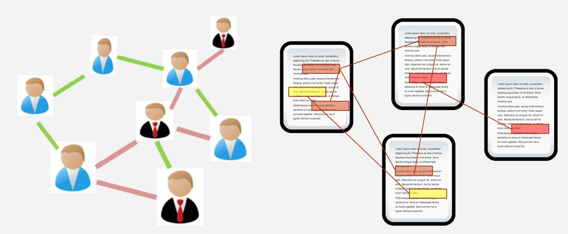
SCHEMALESS GRAPH QUERYING

Shengqi Yang, Yinghui Wu, Huan Sun and Xifeng Yan
{sqyang, yinghui, huansun, xyan}@cs.ucsb.edu

OVERVIEW

Motivation



The big graph challenge

Real graph is large.

Real graph is heterogeneous.

- The nodes and relations are from various domains and have rich content.

The query challenge

Queries are often schemaless

- End users possess little or no prior knowledge of the underlying data.
- There is no unified data specification and vocabulary followed by the data contributors and end users.

Contributions

A novel transformation-based matching strategy.

- Name the query and the search engine will do the rest.

An efficient graph search algorithm to fast find the results.

A principled ranking method based on machine learning algorithm.

Impact

◊ **I have no idea about schema/data specification/query language; yet I still want to query graph data.**

◊ **I want to query not only the knowledge graphs but also the document corpus or even the relational tables.**

Related Work: BANKS, YAGO-NAGA, BLINKS, SAGA, NeMa, ...

MATCHING

Transformation-based matching

◊The users without prior knowledge of the graph can freely post queries.

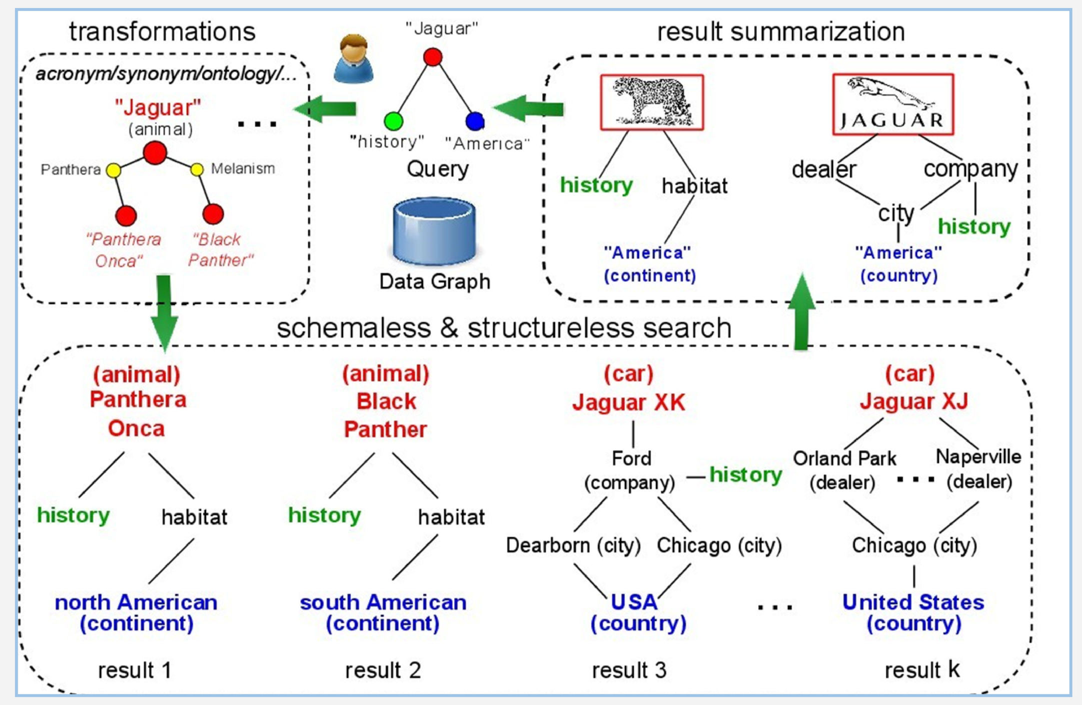
◊The system automatically finds the matches by a set of **transformations**.

⇒ **Acronym** "UCB" to "University of California, Berkeley"
⇒ **First token** "Chris" to "Chris Pine"
⇒ **Abbreviation** "M : I" to "Mission : Impossible"
⇒ **Topology** "Chris—M:I" to "Chris—Abrams—M:I"

Transformation	Category	Example (Data)		Example (Query)
First/Last token	String	"Barack Obama"	>	"Obama"
Abbreviation	String	"Jeffrey Jacob Abrams"	>	"J. J. Abrams"
Prefix	String	"Doctor"	>	"Dr"
Acronym	String	"International Business Machines"	>	"IBM"
Synonym	Semantic	"tumor"	>	"neoplasm"
Ontology	Semantic	"teacher"	>	"educator"
Range	Numeric	"~30"	>	"1980"
Distance	Topology	"Pine"-"M:I"	>	"Pine"-"J.J. Abrams"-"M:I"

* A list of example transformations. More transformations can be easily plugged into the framework.

HIGHLIGHTS



Technique Highlights

- **Support various query forms.**
  Current: Keyword query, graph query, results visualization and summarization.
  Future: Query-by-example, natural language query, user feedback
- **No knowledge on the query language and the underlying data schema is required.**

Publications

- **Schemaless graph querying** - SIGMOD14 demo, VLDB14
- **Result summarization** - VLDB14
- **Ontology-based indexing technique** - ICDE13

RANKING

The Ranking Model

With a set of matching/transformations, given a query Q and its result R, the ranking model considers

- **Node matching**: query node **v** to its match $\phi(v)$

$$F_V(v,\phi(v)) = \sum_i \alpha_i f_i(v,\phi(v))$$

- **Edge matching**: query edge **e** to its match $\phi(e)$

$$F_E(e,\phi(e)) = \sum_i \beta_i f_i(e,\phi(e))$$

The overall model: a probabilistic model based on **Conditional Random Fields (CRFs)**.

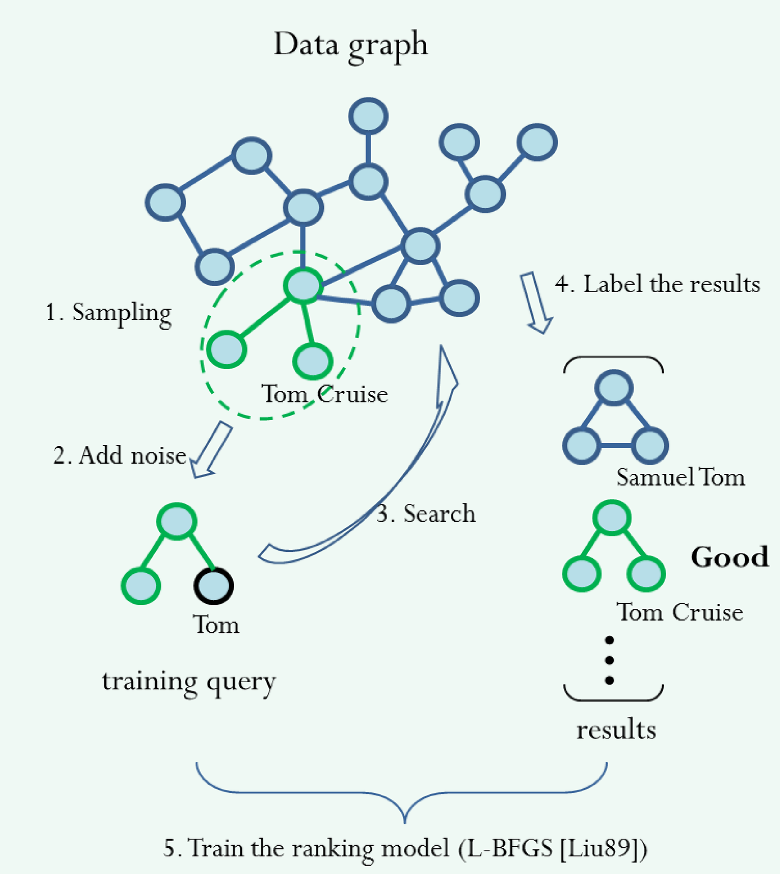
$$P(R\,|\,Q) \propto \exp\Big(\sum_{v\in V_Q} F_V(v,\phi(v)) + \sum_{e\in E_Q} F_E(e,\phi(e))\Big)$$

Parameter Learning

The parameters $\{\alpha_i;\beta_j\}$ have to be determined properly.

- **Warm-start**
  User query logs
  Manual labels
- **Cold-start**
  Automatic training data generation

Automatic Training Data Generation

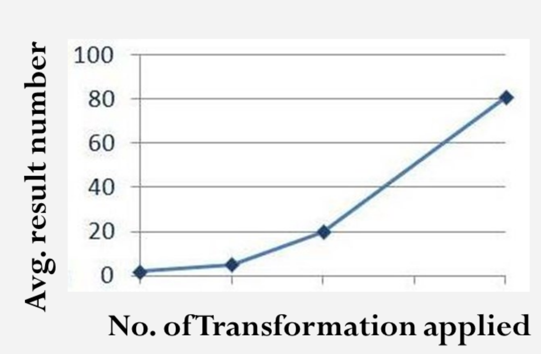


1. **Sampling**: a set of subgraphs are randomly extracted from the data graph.
2. **Query generation**: randomly add transformations to the extracted subgraphs.
3. **Searching**: search the generated queries on the data graph.
4. **Labeling**: the results are labeled based on the original subgraph.
5. **Model training**

SEARCHING

Exact search

The transformations incur many match candidates. Exact search is quite expensive.



Inference in the graphical model

◊ A CRFs model is constructed based on the query and the match candidates.

◊ Top-1 result: the most likely assignment (MAE).

1. Approximate inference: **Loopy Belief Propagation**.
2. Two-level search: **sketch graph**.

◊ Top-K result: best max-marginal first algorithm [Yanover04nips].

ARCHITECTURE

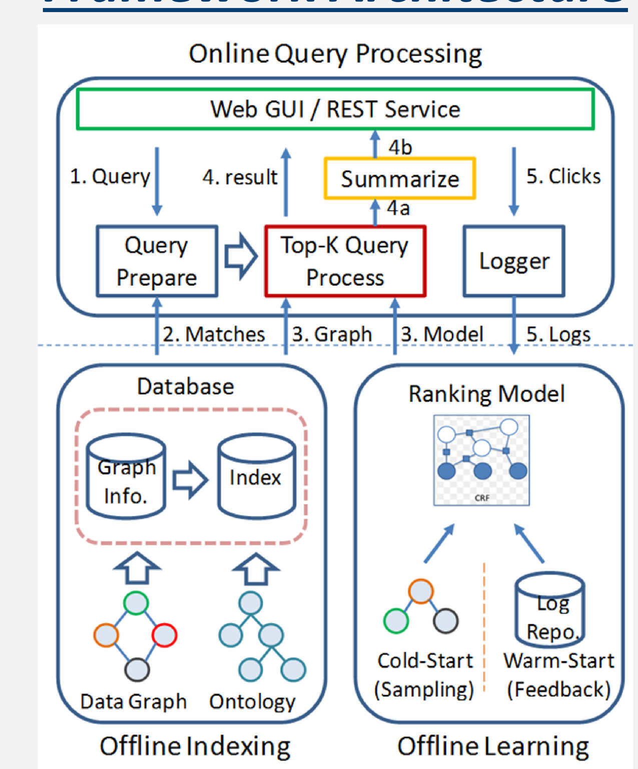
The front-end modules

- **Query Prepare**: interpret the input query and find the matches from the index.
- **Top-K search**: apply the ranking model to find the top results.
- **Logger, Summarizer**, etc.

The back-end modules

- **Indexing:** support the transformation based matching.
- **Leaner**: train/refine the ranking model with the labeled logs.
- **Distributed scheduler** (Akka), etc.

Framework Architecture



RESULTS

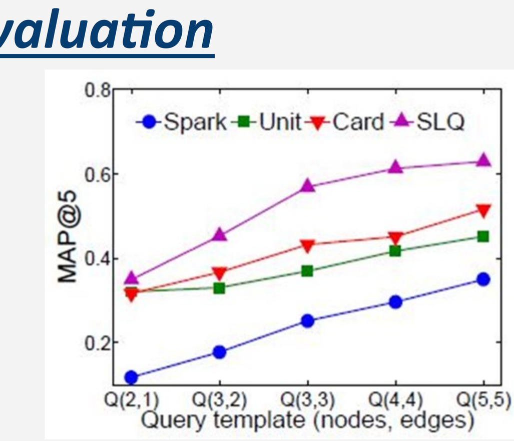
Dataset

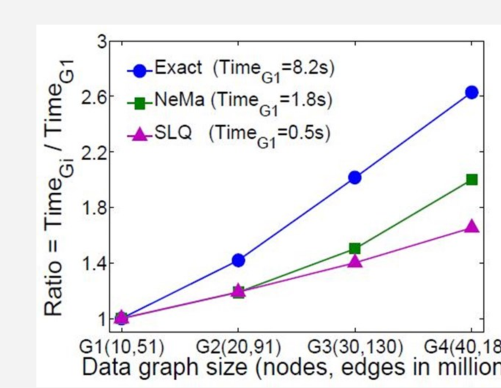
Graph	Nodes	Edges	Node types	Relations	Size
DBpedia	3.7M	20M	359	800	40G
YAGO2	2.9M	11M	6,543	349	18.5G
Freebase	40.3M	180M	10,110	9,101	88G

Baseline

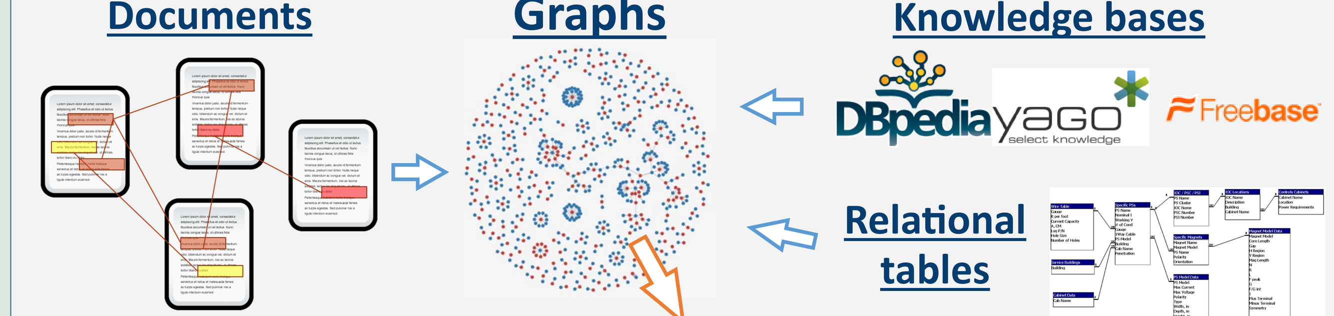
◊ Spark [Luo07] : IR based ranking/searching method.
◊ SLQ: the proposed method in this work.
◊ Unit: a variant of SLQ, with equal parameter in the model.
◊ Card: a variant of SLQ, with the parameter as the selectivity of the corresponding transformation.

Evaluation





APPLICATIONS



Search Portal