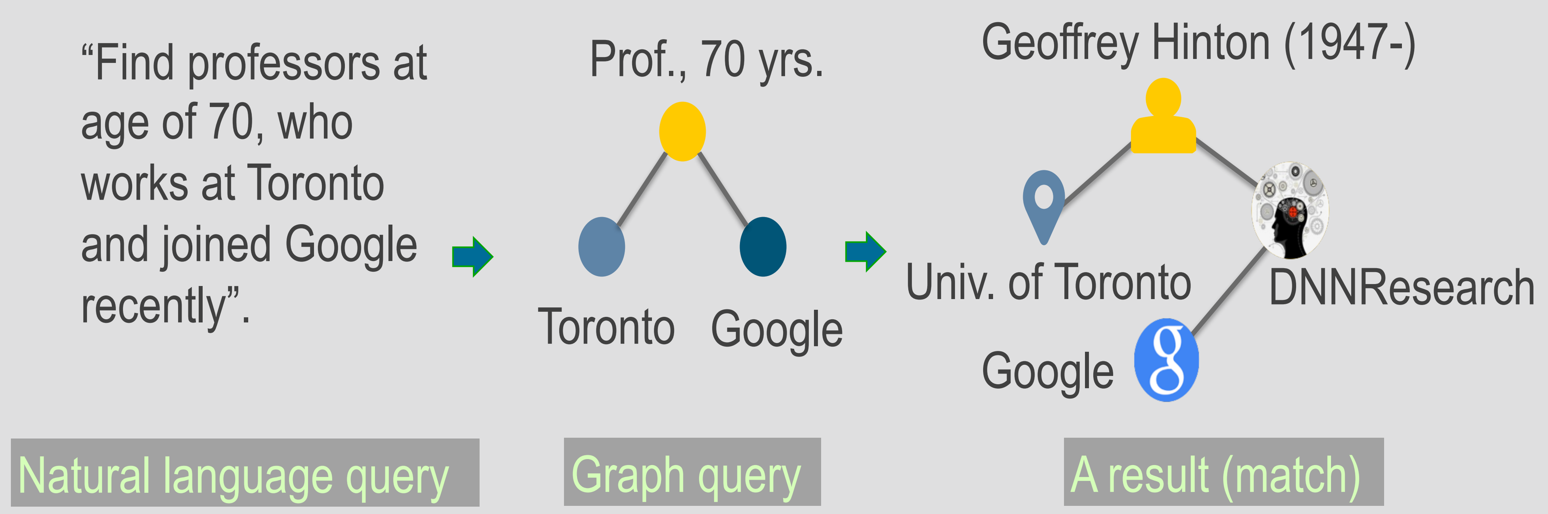
Exploiting Relevance Feedback in Knowledge Graph Search

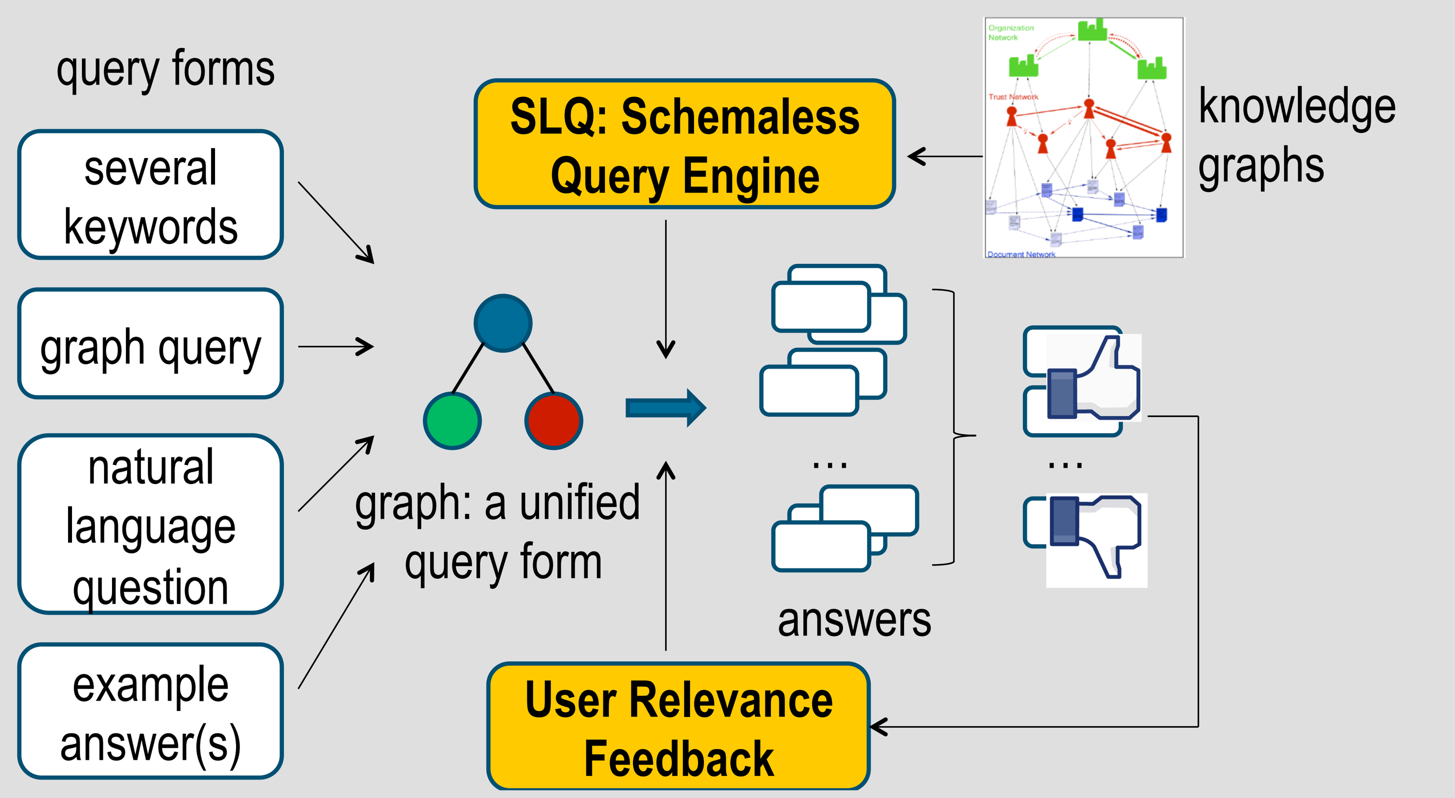
Yu Su[1], Shengqi Yang[1], Huan Sun[1], Mudhakar Srivatsa[2], Sue Kase[3], Michelle Vanni[3], and Xifeng Yan[1]
[1]University of California, Santa Barbara, [2]IBM Research, [3]U.S. Army Research Lab

Introduction

- Graph query is a promising query paradigm for knowledge graphs: It enjoys both the user-friendliness of keyword query and the expressivity of structured query.



- I have no idea about schema/data specification; yet I still want to query knowledge graphs.



SLQ Demo: http://www.cs.ucsb.edu/~xyan/inarc/slq_demo_v4.mp4

- We make the first attempt to study relevance feedback in graph query, i.e., **graph relevance feedback** (GRF). We propose a general GRF framework which mines various relevance information from user feedback. Experiment results show that it can improve the precision of SLQ by **80% to 100%**.

Framework

SLQ Ranking Function

$$F(\phi(Q)\,|\,Q,\boldsymbol{\theta}) = \frac{1}{Z}\exp\Big(\sum_{v\in V_Q} F_V(v,\phi(v)) + \sum_{e\in E_Q} F_E(e,\phi(e))\Big)$$

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

$$F_E(e,\phi(e)) = \sum_i \beta_i \cdot f_i(e,\phi(e)),\ \boldsymbol{\theta} = \{\alpha_i, \beta_i\}.$$

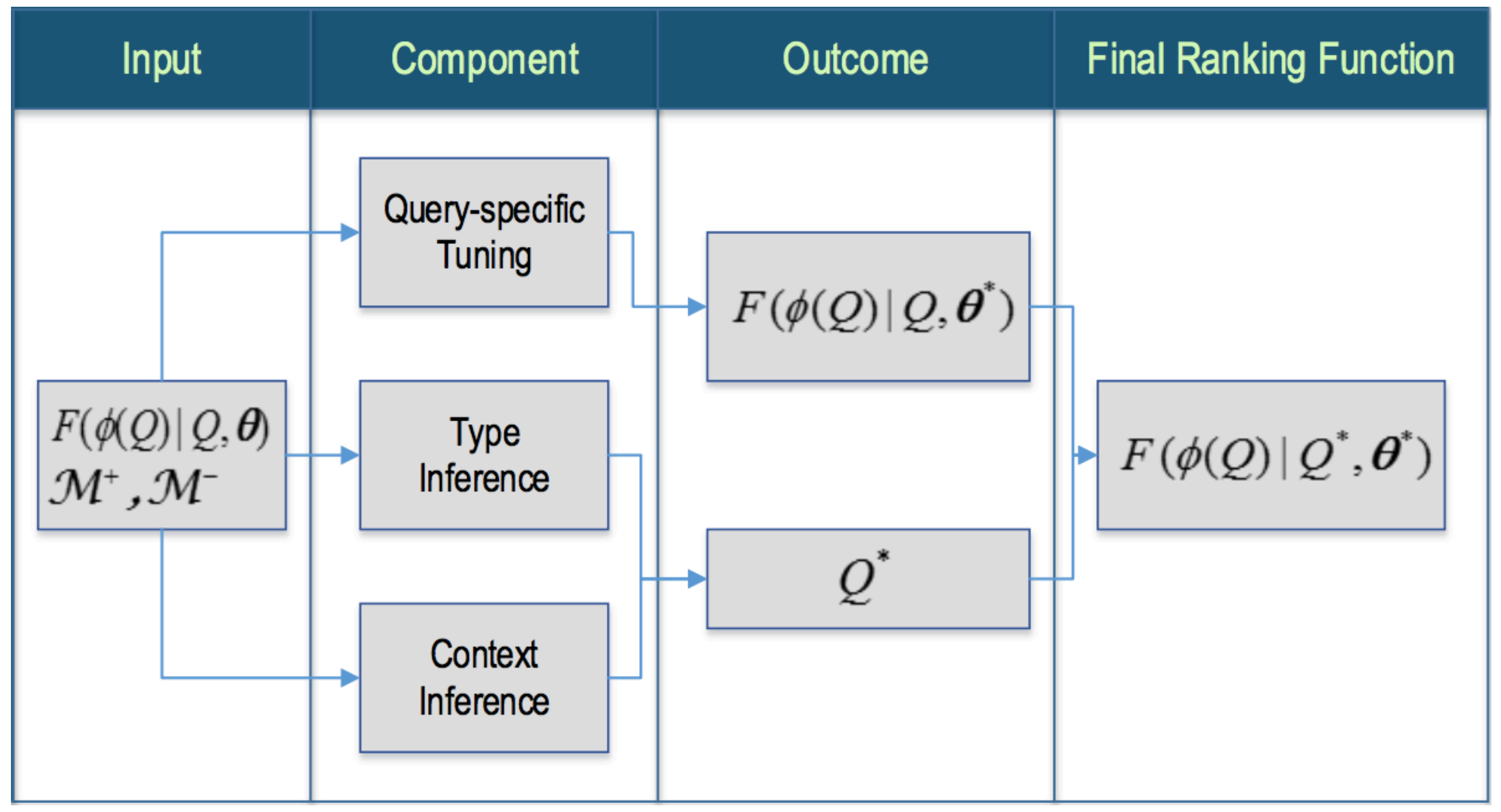


Figure 2. A general GRF framework

Framework (Cont'd)

Query-specific Tuning

[Motivation] The parameters $\boldsymbol{\theta}$ specify **query-general** feature weights, but each query carries its own view of feature importance.
[Optimization with regularization] Find **query-specific** feature weights $\boldsymbol{\theta}^*$ using user feedback

$$g(\boldsymbol{\theta}^*) = \underbrace{(1-\lambda)\Big(\frac{\sum_{\phi(Q)\in\mathcal{M}^+} F(\phi(Q)|Q,\boldsymbol{\theta}^*)}{|\mathcal{M}^+|} - \frac{\sum_{\phi(Q)\in\mathcal{M}^-} F(\phi(Q)|Q,\boldsymbol{\theta}^*)}{|\mathcal{M}^-|}\Big)}_{\text{User Feedback}} + \underbrace{\lambda R(\boldsymbol{\theta},\boldsymbol{\theta}^*)}_{\text{Regularization}}$$

[Balance parameter] λ controls the balance between query-general parameters and user feedback. The goal is to tune the ranking function using user feedback while not **overfitting** to it.

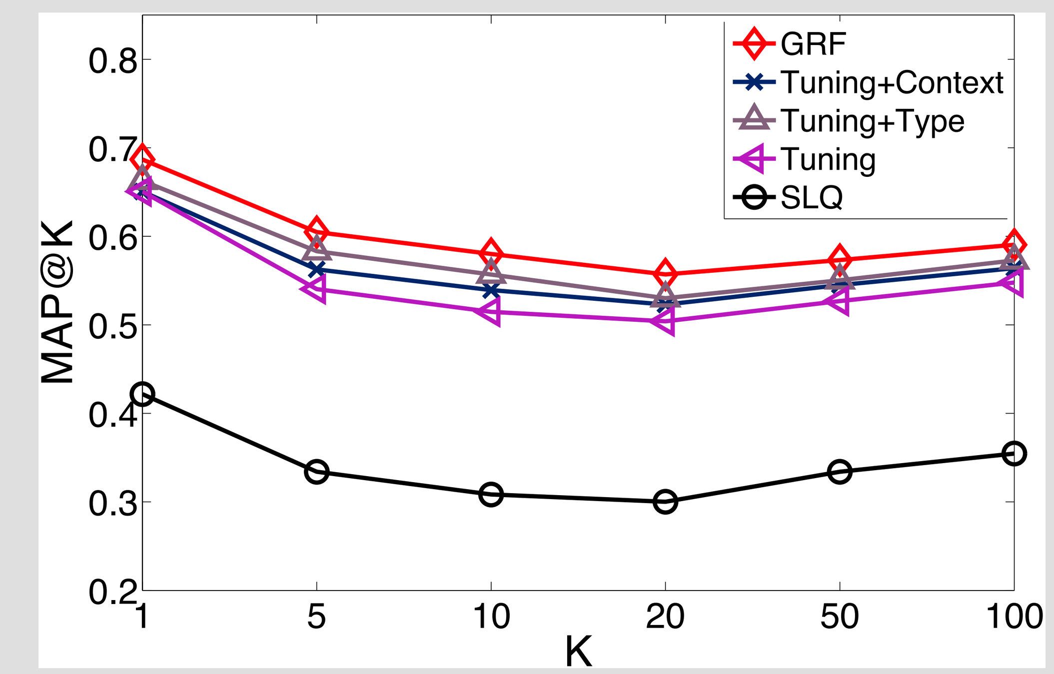
Type and Context Inference

[Motivation] When a user formulates a query, there is more information she has in mind but doesn't state explicitly. For example, by "Toronto", the user implies a university (**type**) that has many professors and students (**context**). Infer such implicit information and add it back to the query may greatly help disambiguation
[Entity Relevance Score] Two relevance scores are defined for each candidate entity by calculating its similarity to the positive entities in terms of type and context (neighborhood type distribution).
The relevance scores are then plugged into the tuned ranking function as **new features**.

Experiments

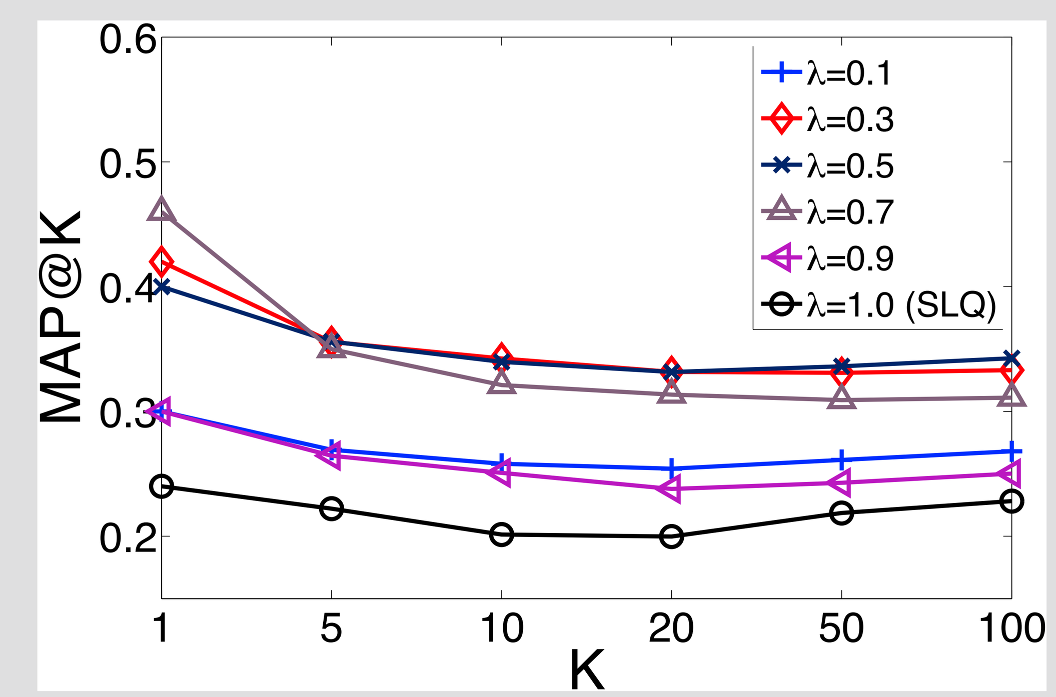
[Knowledge Graph] DBpedia (4.6M nodes, 100M edges)
[Query Sets] Ground-truth queries derived from Wikipedia and YAGO
[Metric] Mean Average Precision at different cutoff K (MAP@K)
[Relevance Feedback] Simulate explicit feedback using ground truth (see the paper for experiments with pseudo feedback)

Overall Performance



The three components complement each other and the full GRF achieves the best mean average precision (80%-100% improvement).

Impact of Balance Parameter



Select an appropriate value for λ to make a good balance between query-general parameters and user feedback. When λ is too small, we overfit to the user feedback, and answer quality decreases.

Conclusion

- We proposed a graph relevance feedback framework which can improve the precision of a state-of-the-art graph query system by 80% to 100%
- One meaningful extension is to study long-term user personalization using relevance feedback for graph query.