



# Ranking Objects by Following Paths in Entity-Relationship Graphs

Minsuk Kahng, Sangkeun Lee and Sang-goo Lee

School of Computer Science and Engineering  
Seoul National University, Seoul, Republic of Korea

---

4<sup>th</sup> Workshop for Ph.D. Students in  
Information and Knowledge Management (PIKM 2011)  
in conjunction with CIKM 2011

October 28<sup>th</sup>, 2011

# Problem

---

- Objective
  - Given some query objects,  
rank objects of a specified type
- Examples
  - For a given user and current timestamp,  
which songs are most preferable?
  - For a given paper,  
which conferences are most relevant to be submitted?



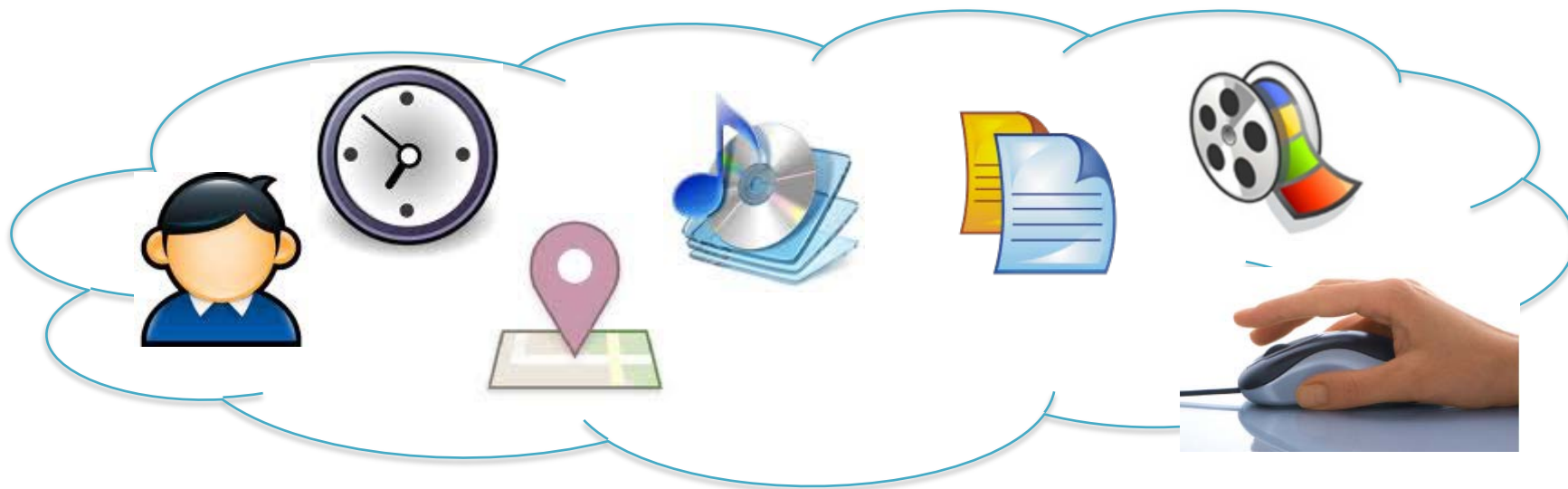
# Outline

---

- Introduction
- Data Model
- Method
- Preliminary Experiments
- Related Work
- Challenging Issues
- Conclusion

# Heterogeneous Data for Search

- Various types of data for search and recommendation
  - Traditionally, only documents and words
  - Now, **various objects** and **relationships** between them
    - Documents, Products, Music, User, Clickthrough Log, Map, Contextual Data (e.g. time & place)
- Incorporating these heterogeneous data is key!

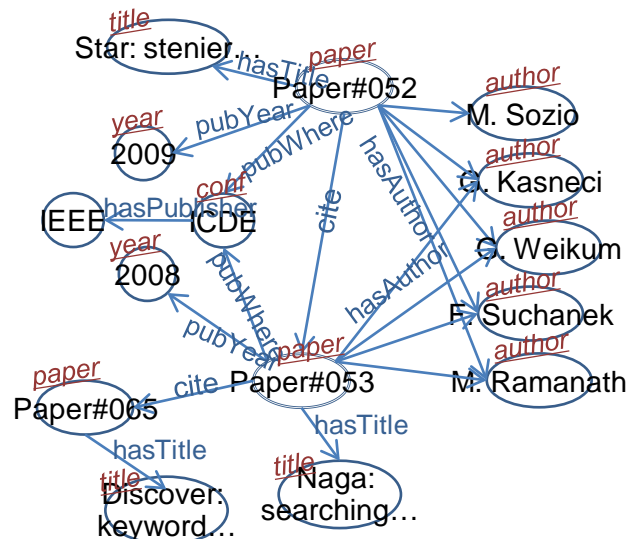


# Utilize Graph-based Data Model

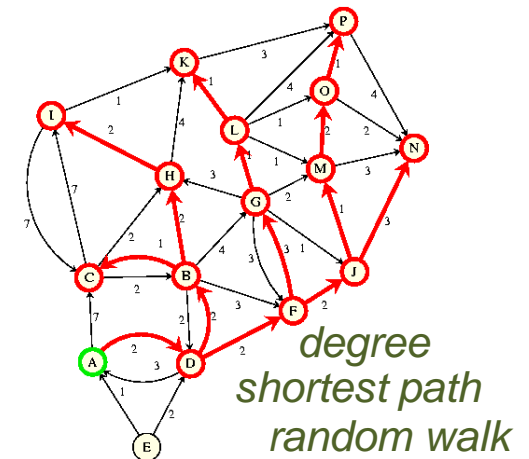
- Graph-based data models have gained popularity for dealing with these heterogeneous data.
  - Objects and relationships can be modeled by nodes and edges.
  - It is called “Entity-Relationship Graphs”.
- Solve the problem with graph operations

User	Doc	Click	Time
#12	cikm10.org	0	11:00
#12	cikm11.org	1	11:01
#500	...	0	13:45
#500	...	0	13:46
#500	...	1	13:46
#904	...	1	13:51

*Original database*



*Graph data model*



*Graph problem*

# Object Ranking with Graphs

---

- With graph-based data models
  - Problem Definition
    - Given query objects, rank objects of a specified type
  - With graph models, it becomes:
    - Which target nodes are most **similar** to the query nodes?
- Challenges
  - How to **define similarity** between nodes in the graph?
    - Not just shortest distances
    - Capture **semantics** of structured data

# Existing work, but...

---





- Homogeneous Graph
  - focus on structure of graph, not consider ‘type’ of nodes/edges
  - PageRank – inlink/outlink of nodes, random walks, ... [1]
- Consider the type of edges: Edge-level feature
  - Each edge has a ‘type’.
  - ObjectRank - Each edge type has different weight [2].
- But still, hard to capture semantics
  - Importance of edge type is always same regardless of tasks

*[1] Brin and Page. ... hypertextual web search engine. In WWW, 1998.*

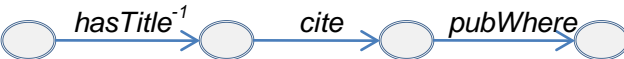

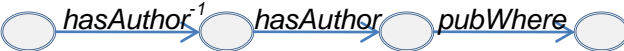

*[2] Balmin et al. Objectrank: authority-based keyword search .... In VLDB, 2004.*

# Path-level feature instead of edge

- Recently, the **path-level feature** has been mentioned as a replacement for the **edge-level feature** [3, 4].
  - Each path, a sequence of edge types, has weight.
  - An edge has a different role depending on the tasks.

feature	weight
	0.90
	0.50
	0.35
	0.15

edge-level feature

feature	weight
	0.90
	0.50
	0.35
	0.15

path-level feature

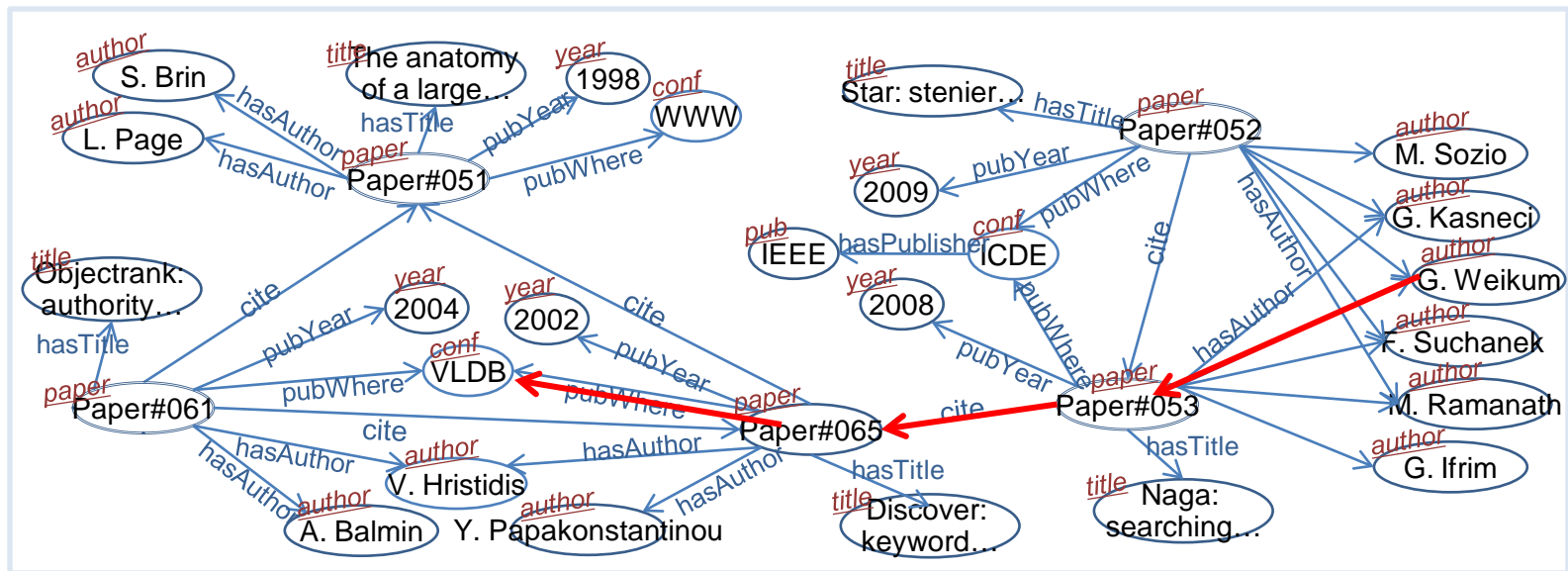
[3] Sun et al. PathSim: meta path-based top-k similarity .... In VLDB, 2011.

[4] Lao et al. ... retrieval ... path constrained random walks. In ECML-PKDD, 2010.



# Proposed: Object Ranking with Paths

- We propose an **object ranking** method.
  - Transform original data into graph-based data models
  - Use **paths** in the graphs for ranking



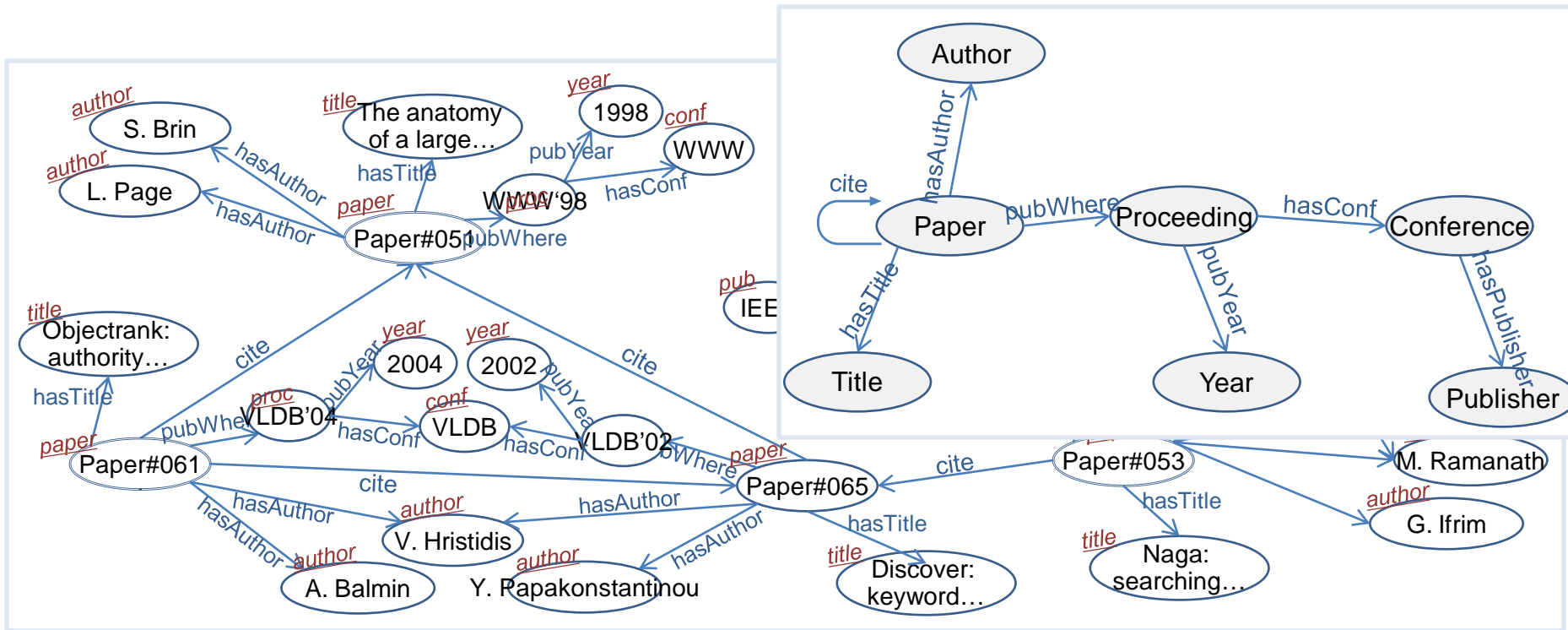
# Outline

---

- Introduction
- **Data Model**
- Method

# Data Graph & Schema Graph

- Data graph – Instances (types both on nodes & edges)
- Schema graph – Schema of data graph



## Example of data graph

Using DBLP dataset, we build a data graph which has types on both nodes & edges.

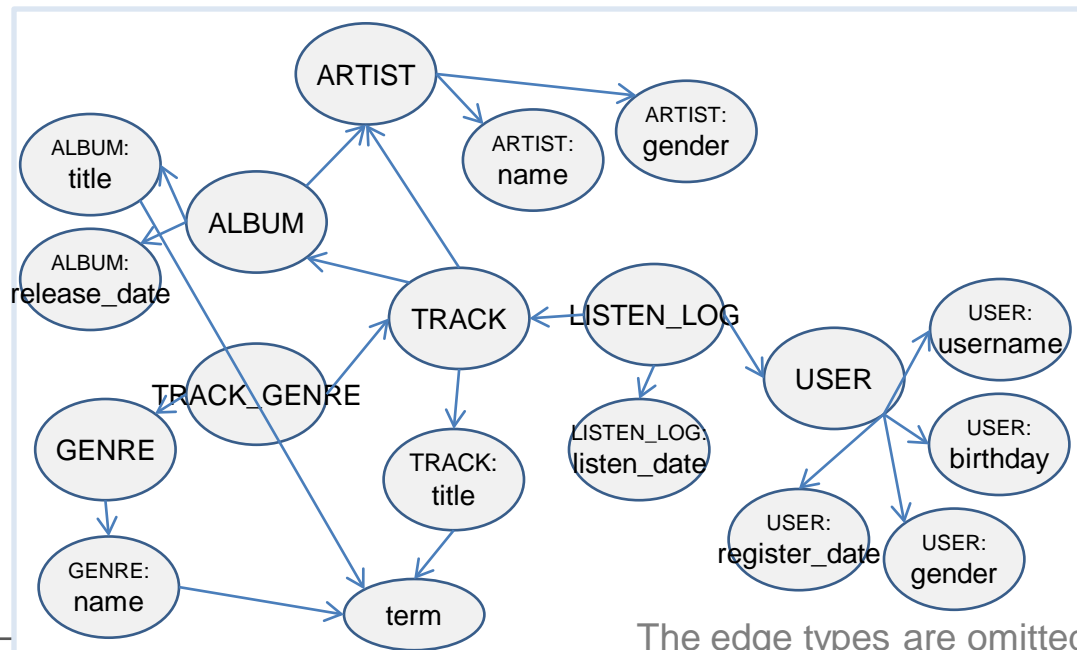
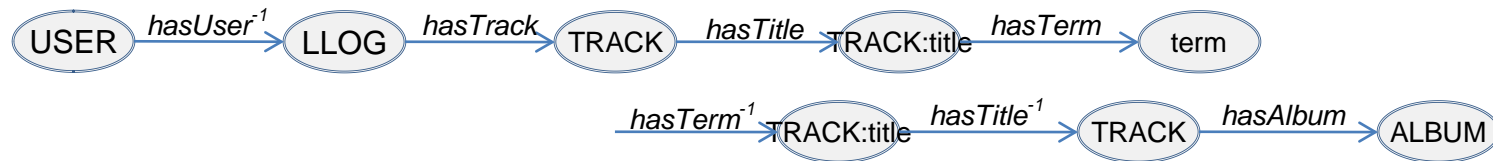
# Method Overview

---

- Problem
  - Given query nodes and target type, which target nodes are most relevant to the query nodes?
- Steps
  1. Choose schema paths from the schema graph
  2. For each path, look into data graph, and score target nodes
  3. Combine those results of each path

# 1-1) What is Schema Path?

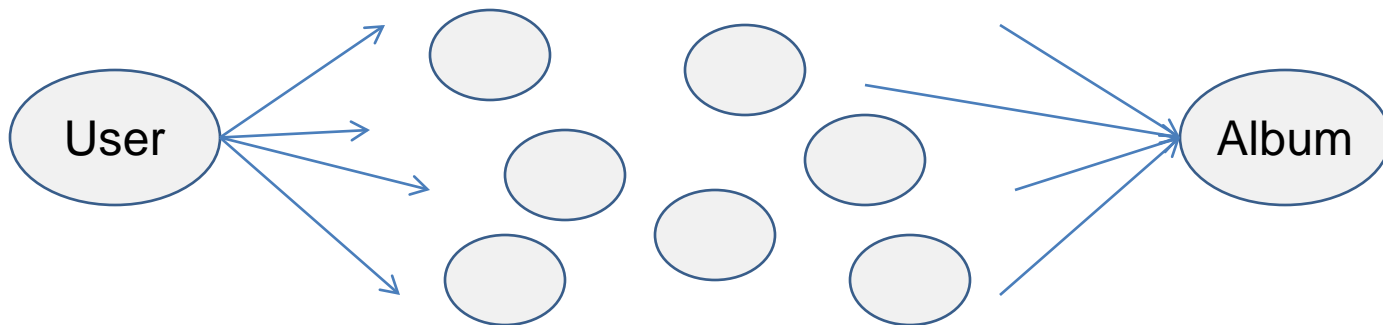
- A schema path  $p$  is a sequence of edge types.
- Represents a workflow of how we traverse the graph.



The edge types are omitted.

# 1-2) Choose Schema Paths

- Discover candidate schema paths, and select some of them.
  - Simply, find all paths between two nodes on schema graph
    - 1) User > Log > Song > Album
    - 2) User > Log > Song > Artist > Album



- Symmetric Paths
  - a) User > Log > Song > Log > User
  - b) Song > SongTitle > Term > SongTitle > Song
- Expanding with Symmetric Paths
  - 1+a) User > Log > Song > Log > User > Log > Song -> Album (CF)
  - 1+b) User > Log > Song > STitle > Term > STitle > Song > Album

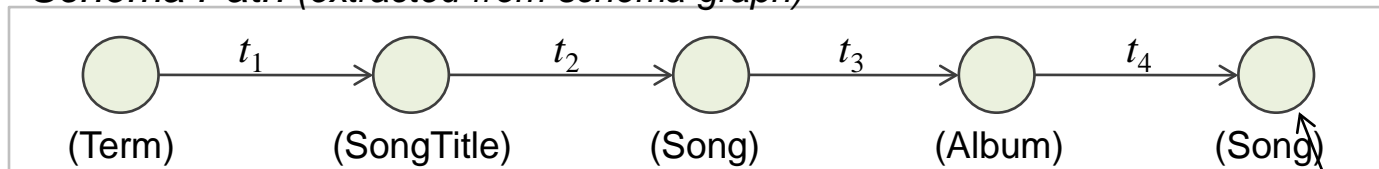
symmetric path

## 2-1) For each path, look into data graph

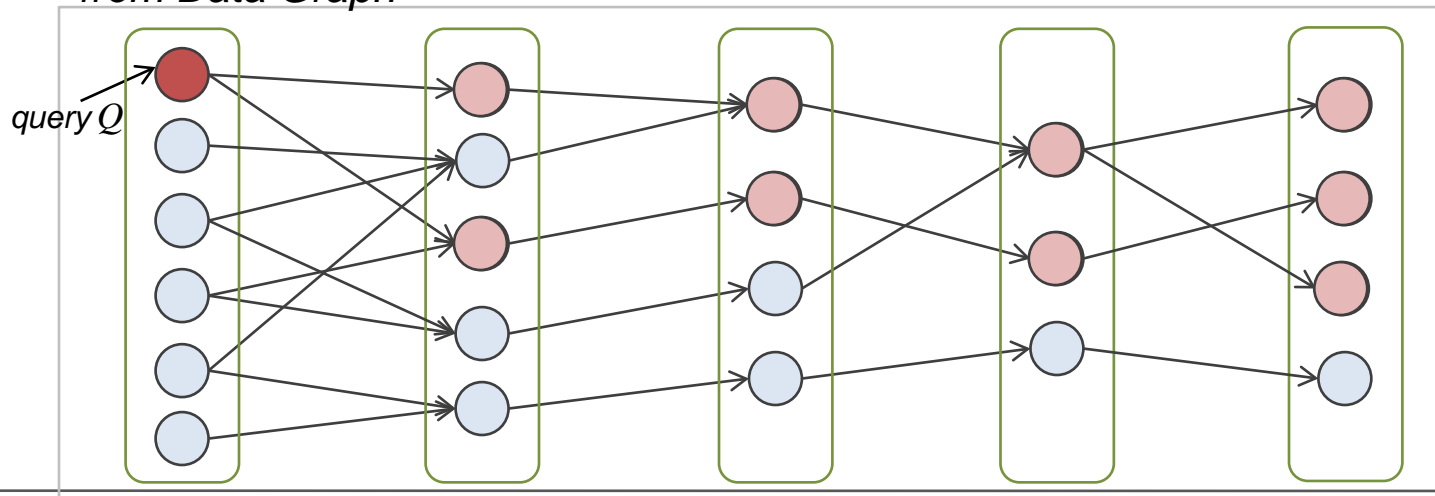
- For each path, look into data graph, and score target objects.
- From query to target, follow edges in the path.

$$r = M^{(m)} \times \dots \times M^{(1)} \times q = A \times q$$

*Schema Path (extracted from schema graph)*



*from Data Graph*

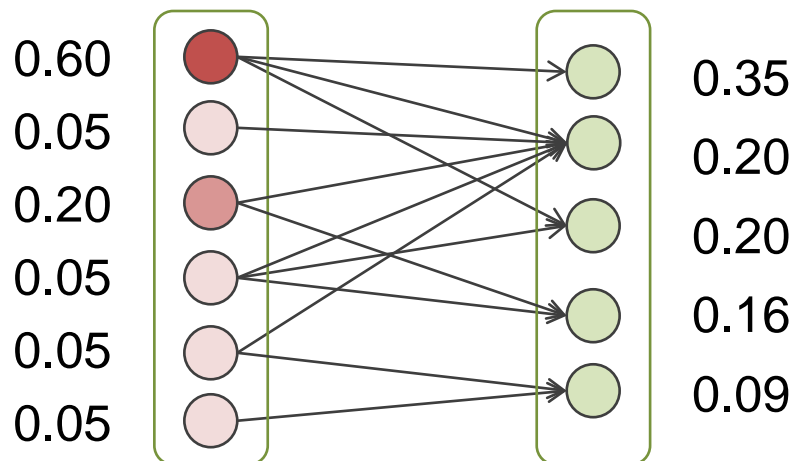


## 2-2) How to give scores to nodes?

- How to give scores to the nodes during propagation?
- Define scoring functions
  - Number of paths
  - Distribute the current state equally (random walk)
  - Distribute, but not equally (e.g. for diminishing popularity effect)

$$r = M^{(m)} \times \dots \times M^{(1)} \times q = A \times q$$

Define scoring functions  
= Define how to fill value of the matrix  $M$

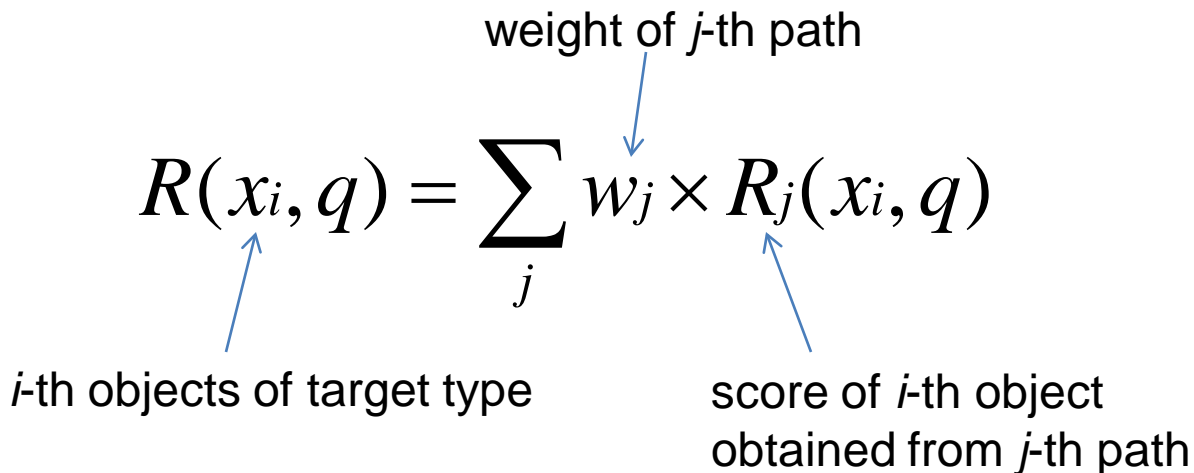




### 3) Combine the results

---

- Now, we have scores of nodes obtained from each path.
- Weighted combination of each result
  - Manually
  - End user
  - Learning (e.g. learning-to-rank)

$$R(x_i, q) = \sum_j w_j \times R_j(x_i, q)$$


weight of  $j$ -th path

$i$ -th objects of target type

score of  $i$ -th object obtained from  $j$ -th path

# Outline

---

- Introduction
- Data Model
- Method
- **Preliminary Experiments**
  - Datasets
  - Tasks (Scenarios)

## Two Real-world Datasets

---

- Music streaming service (Bugs Music)
  - Music Metadata (song, album, artist, genre)
  - Users' Listening Log
    - e.g. "User #12' listened to 'Song #59' at time  $t$ ."
  - Node types: song, album, artist, genre, user, date, log, etc.
- DBLP publication
  - All papers published in 20 major conferences
  - Node types: paper, author, conference, etc.

# Task #1: Basic Scenario

- Scenario: Recommend related conferences
  - Input: a set of terms
  - Output: Conferences
- Path
  - (Term) -> (PaperTitle) -> (Paper) -> (Conference)

Query #1:  
“twitter”

1	WWW
2	ICWSM
3	CHI
4	ECIR
5	WSDM

Query #2:  
“graph clustering”

1	KDD
2	CIKM
3	ICDE
4	ICDM
5	SDM

Query #3:  
“transaction lock  
protocol optimizing”

1	ICDE
2	SIGMOD
3	VLDB
4	DASFAA
5	CIKM

Query #4:  
“language model  
smoothing”

1	SIGIR
2	ECIR
3	CIKM
4	AAAI
5	WWW

## Task #2: Different paths

- Research Q: How two different paths produce different results
- Scenario: Find similar papers
  - Input: particular Paper(w/ title)
  - Output: Papers(w/ title)

Query: “DISCOVER: Keyword Search in Relational Databases”

Path #1: “(Paper) -> (PaperTitle) -> (Term) -> (PaperTitle) -> (Paper)”

1	DISCOVER: Keyword Search in Relational Datab.
2	Discover Relaxed Periodicity in Temporal Datab.
3	Discover Relev. Env. Feature Using Concurrent Reinf. Learning
4	Mining Propositional Knowl. Bases to Discover Multi-level Rules
5	Mining Multivar. Time-Ser. Sensor Data to Discover Behav. Envel.

Path #2: “(Paper) -> (PaperAuthorRelation) -> (Author) -> (PaperAuthorRelation) -> (Paper)”

1	DISCOVER: Keyword Search in Relational Datab.
2	ObjectRank: Authority-Based Keyword Search in Datab.
3	ObjectRank: A System for Auth.-based Search on Datab.
4	Keyword Proximity Search on XML Graphs
5	Efficient IR-Style Keyword Search over Relational Datab.

## Task #3: Combine Paths

- Research Q: Combine results from two different paths
- Scenario: Given a user and current date, recommend artists based on user's listening log and daily popularity
  - Input: particular User, Current Date
  - Output: Artists

Path #1: “(User) -> (ListenLog) -> (Song) -> (ListenLog) -> (User) -> (ListenLog) -> (Song) -> (Artist)”

Path #2: “(Date) -> (ListenTimestamp) -> (ListenLog) -> (Song) -> (Artist)”

	Artist's name	Path #1	Path #2
1	Black Eyed Peas	1	21
2	8Elight (local)	8	2
3	V.O.S. (local)	-	1
4	Eminem	4	37
5	Ciara	2	86

# Outline

---

- Introduction
- Data Model
- Method
- Preliminary Experiments
- Related Work
  - Keyword search in DB
  - Recommender systems
  - Graph-based Ranking
- Challenging Issues
  - Learning
  - Efficiency
  - Flexible Querying Framework
  - Result Presentation

# Related Work

---

- Keyword Search in DB
  - Also deal with heterogeneous objects
  - More concerned with performance, not ranking
- Recommender Systems
  - Focus more on semantic similarity
  - Only two types of entities: User & Item
- Graph-based Ranking
  - Mostly edge-level
  - Recently, path-level methods have been introduced.
  - Some work with RDF & SPARQL



# Challenging Issues (Future Work)

---

- Learning
  - can improve the quality of ranked results
  - Weights of paths can be determined (e.g. learning-to-rank) [4]
- Efficiency
  - Performance issues when data size go up
  - Materializing [3], Filtering
- Flexible Querying Framework
  - Expressiveness of query [5]
  - Provide operators, conditional clauses, choose score functions
- Result presentation
  - Explanation of results [6] (why it is ranked 1<sup>st</sup>)
  - Schema path would be helpful.

[5] Varadarajan et al. *Flexible and efficient querying and ranking .... In EDBT, 2009.*

[6] Yu et al. *Recommendation diversification using explanations. In ICDE, 2009.*

# Conclusion

---

- Propose an object ranking method.
- Heterogeneous Data
  - Utilize graph-based data model to capture heterogeneity.
- Paths in Graph
  - Use schema path in the graph which implies the semantics.
- Ranking
  - By following these paths, objects are ranked.
- Discussion
  - Many challenging issues & much room for improvements

# Thank you

[minsuk@europa.snu.ac.kr](mailto:minsuk@europa.snu.ac.kr)