Fast Inbound Top-K Query for Random Walk with Restart

Chao Zhang, Shan Jiang, Yucheng Chen, Yidan Sun, Jiawei Han

University of Illinois at Urbana Champaign
czhang82@illinois.edu

Outline

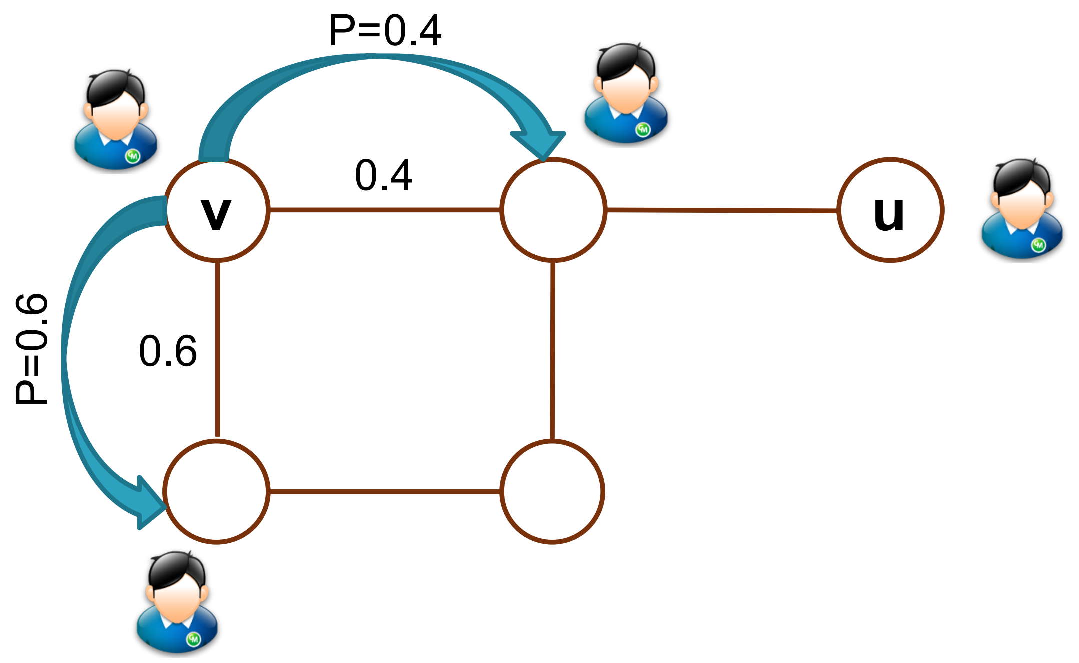
- **Background**
- Our Methods
- Experiments
- Summary

Background

- Random walk with restart (RWR) is a very popular graph proximity measure.
- It is widely used in various applications:
  - link prediction
  - Web search
  - graph clustering
  - and many others…

Random Walk with Restart

- The RWR process
  - A user starts random walk from node v
  - At each step, the user jumps back to v with probability ***c*** (0 < c < 1) and continues walking with probability ***1 - c***
  - The RWR from v to any node u is the probability that the user resides on u after infinite steps



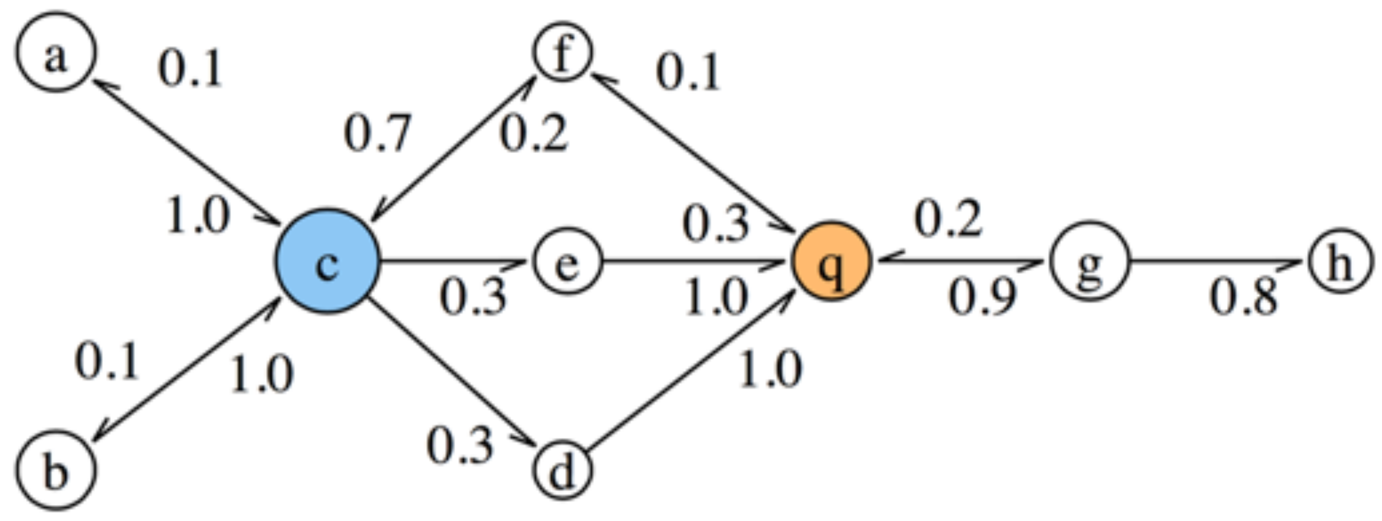
RWR captures the holistic graph structure and is robust to noise.

Top-K Query for RWR

- Prior art: **outbound top-K query**
  - Given a query node q, which K nodes have the largest RWR scores ***from*** q?
- Our work: **inbound top-K query**
  - Given a query node q, which K nodes have the largest RWR scores ***to*** q?

Why Inbound?

- A motivating example
  - A traffic network: each node is a road intersection; each edge is a road segment (the number denotes the traffic volume).
  - *What are the nodes from which the traffic flows to q and causes traffic jams there?*



The inbound top-K query identify the source nodes that have a large amount of information flowing to a query node.

Existing Methods are NOT Fast Enough

- Can we use existing methods for answering inbound top-K queries?
    - Existing techniques can compute the RWR for any node pair in O(n) time, but it is time-consuming to compute all the RWRs to the query node q.
    - Branch-and-bound methods have been proposed for outbound top-K queries, but the bounds are not applicable for inbound top-K queries.

[1] Fujiwara, Y., Nakatsuji, M., Onizuka, M., Kitsuregawa, M.: Fast and exact top-k search for random walk with restart. PVLDB, 2012.

[2] Gupta, M.S., Pathak, A., Chakrabarti, S.: Fast algorithms for top-k personalized pagerank queries. In: WWW. pp. 1225–1226 (2008)

Outline

- Background
- **Our Methods**
- Experiments
- Summary

Overview of Our Methods

- We propose two methods for processing the inbound top-K query: **Squeeze** and **Ripple**.
- Highlights of the two methods:
    - Squeeze is a global method, while Ripple is a local one;
    - Both are guaranteed to obtain the correct top-k results, without any pre-computation.

The Decomposition Theorem

- The RWR from u to q is the linear combination of the RWRs of u's out-neighbors, with extra score if u = q:

$$\mathbf{r}_u(q) = \begin{cases} (1-c)\sum\limits_{v\in O_u} p_{uv}\mathbf{r}_v(q) & \text{if } u \neq q \\ (1-c)\sum\limits_{v\in O_u} p_{uv}\mathbf{r}_v(q) + c & \text{if } u = q. \end{cases}$$

$\mathbf{r}_u(q)$	: the RWR score from u to q
O_u	: u's out-neighbors
p_{uv}	: the transition probability from u to neighbor v

[1] Jeh,G., Widom,J.: Scaling personalized web search. In WWW 2003.

The Squeeze Method

- With the Decomposition Theorem, the RWR scores from all nodes to q is the solution to the following linear system:

$$\mathbf{x}_q = \mathbf{A}\mathbf{x}_q + c\mathbf{e}_q$$

$\mathbf{x}_q$: the RWR vector where $\mathbf{x}_q(u)$ is the RWR from u to q

$\mathbf{A}$: the transition matrix

$\mathbf{e}_q$: a length-n unit vector where the element of q is 1

The Squeeze Method

- Directly solving the linear system has time complexity O(n^3), which is too expensive for large graphs.
- **Key idea of Squeeze**: to find the top-K results, it is unnecessary to obtain the exact RWR scores.
- If we have lower and upper bounds for each node, then we can obtain the top-K results at a cheaper cost.

The Squeeze Method

- Squeeze obtains the top-K results iteratively:
  - Set the lower bound vector to $\mathbf{x}_q^{(0)} = \mathbf{0}$
  - Update the lower bound vector with $\mathbf{x}_q^{(i+1)} = \mathbf{A}\mathbf{x}_q^{(i)} + c\mathbf{e}_q$
  - For each node u, the lower bound is $\mathbf{x}_q^{(i)}(u)$, and the upper bound is $\mathbf{x}_q^{(i)}(u) + (1-c)^i$
  - The gap between the lower bound and upper bound is $(1-c)^i$, which shrinks exponentially with the number of iterations.
  - When the gap is small enough to produce the top-K results, Squeeze terminates the iteration and returns the results.

The Ripple Method

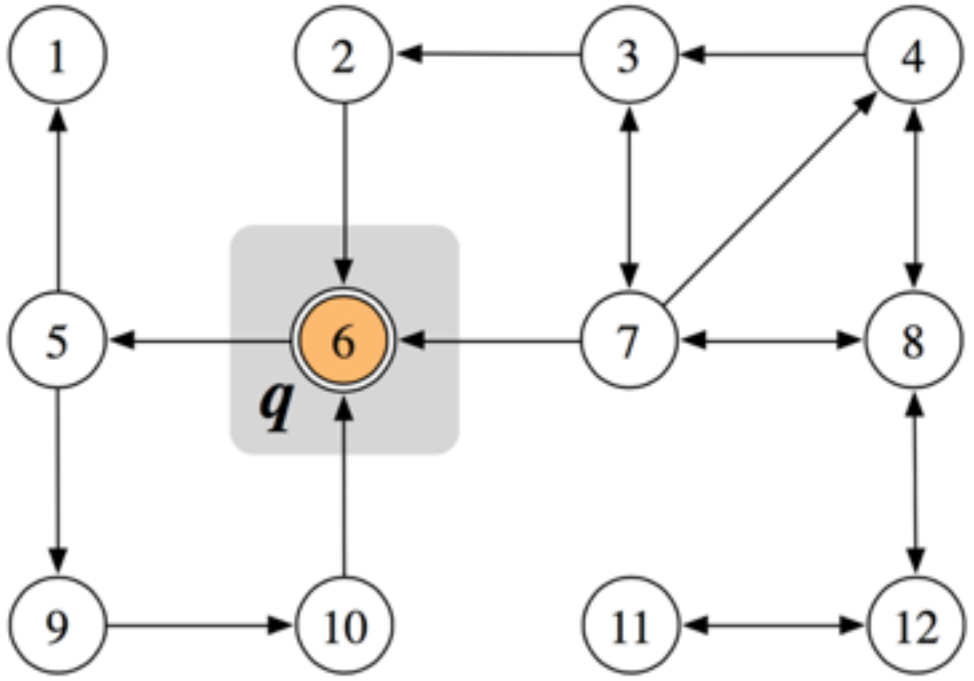
- Squeeze saves cost by computing lower and upper bounds for each node, can we do better?
- **Key idea of Ripple**: RWR has locality, namely the nodes around q tends to have larger RWR scores.
- Ripple maintains a vicinity around q and computes RWRs for only the nodes in the vicinity; for the outside nodes, Ripple maintains a uniform upper bound.

The Ripple Method

- Initialization: $\mathbf{X}_q(q) = c, \boxed{N_q} = \{q\}, \boxed{B_q} = \{q\}$

The vicinity around q

The boundary of the vicinity

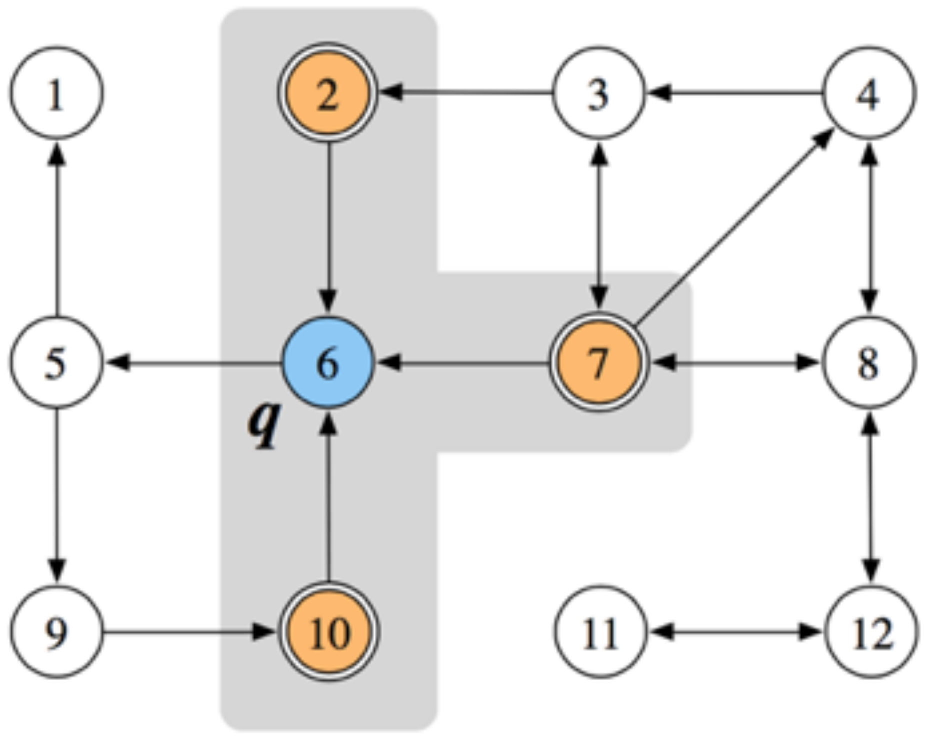


The RWR for any node not in the vicinity is set to 0.

The Ripple Method

- Iterative update:
  - Expand the vicinity from the large-score boundary nodes
  - Propagate the RWR scores among the vicinity nodes
  - Derive a uniform upper bound for the outside nodes

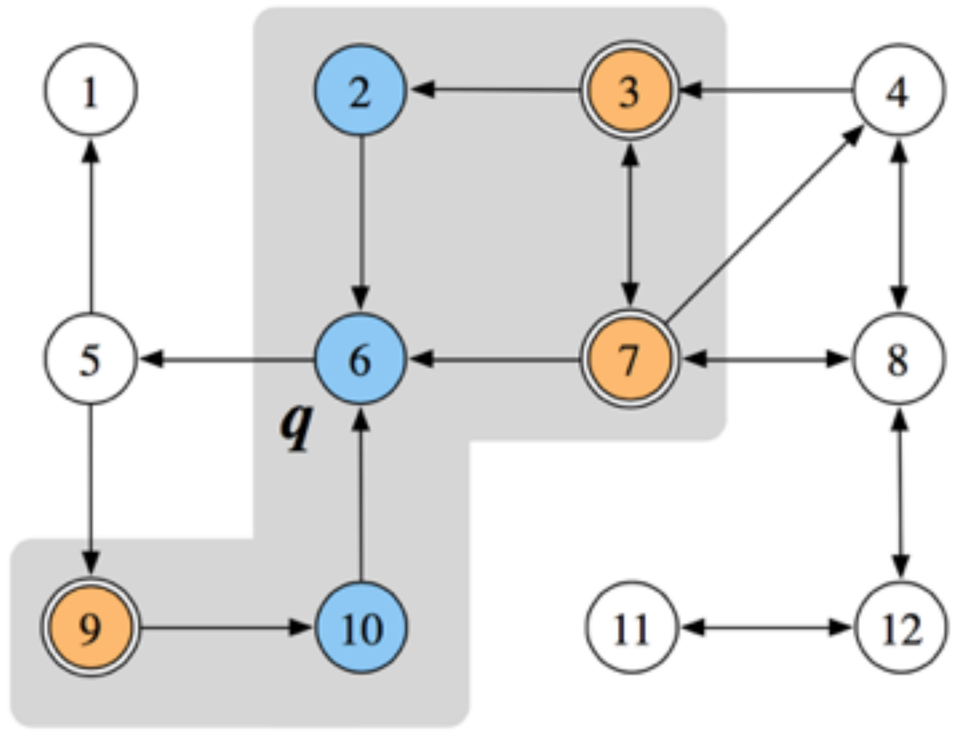
The lower and upper bounds for the inside nodes are refined as more nodes are included in the vicinity.



The RWR upper bound for outside nodes is determined by the boundary nodes that has the largest score.

The Ripple Method

- The vicinity expansion continues until the approximate scores in vicinity are good enough to produce the top-K results.



Ripple saves cost as it never access the nodes outside the vicinity, but only keeps a uniform upper bound for them.

Outline

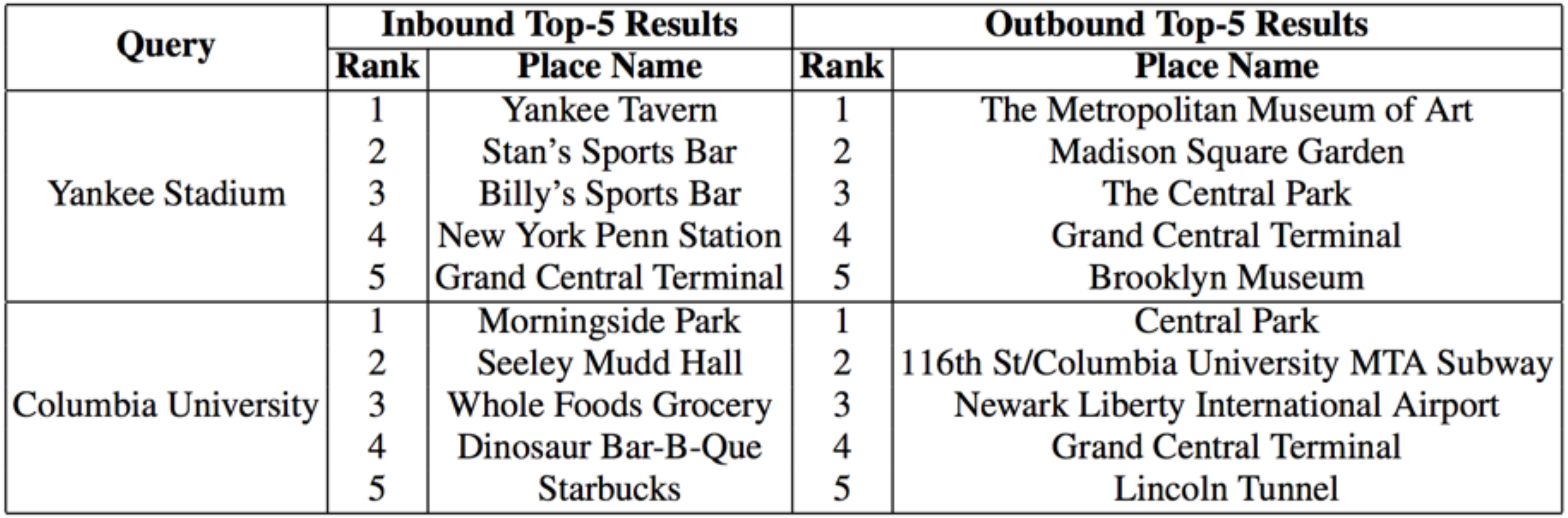
- Background
- Our Methods
- **Experiments**
- Summary

Experiments

- Data sets
  - Wiki: ~4 million nodes and ~100million edges. Each node is a Wikipedia page, each edge is the link from one page to anther.
  - Foursquare: ~50k nodes and ~120k edges. Each nodes is a Foursquare venue, each edge is people's transition from one place to another.
- Compared method
  - LU: state-of-the art method for RWR computation, it computes the RWR scores to q for all the nodes and then select the top-K nodes.

Inbound Top-K v.s. Outbound Top-K

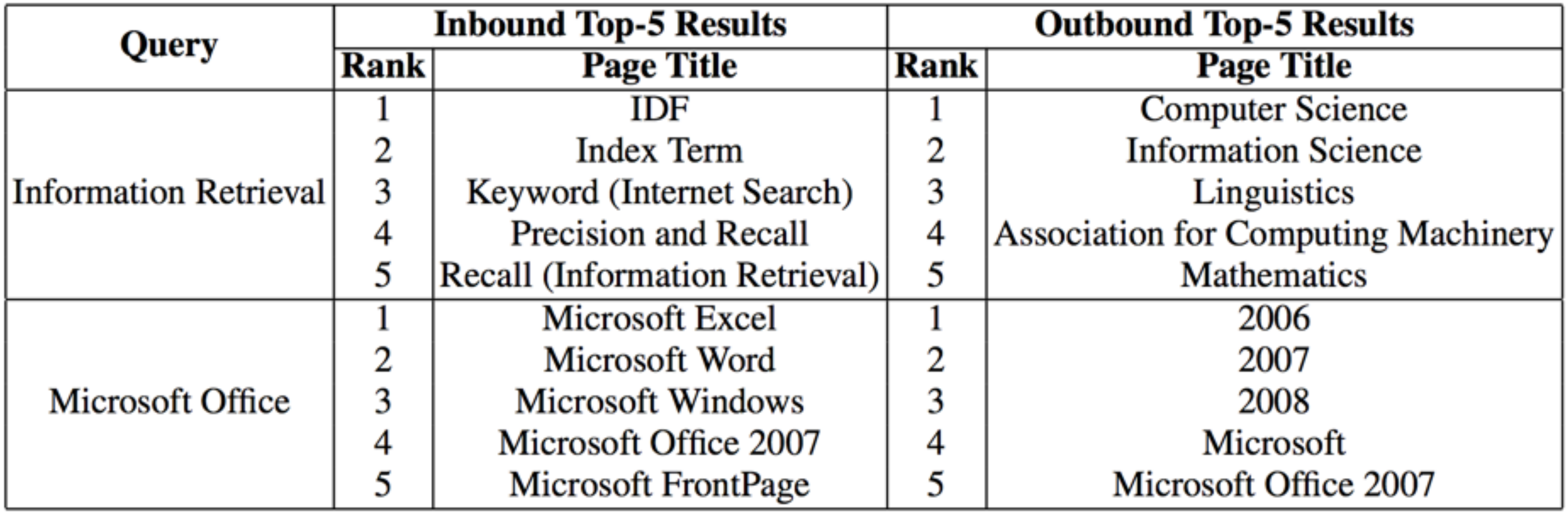
- Illustrative examples on Foursquare
  - Outbound top-k queries tend to retrieve famous venues
  - Inbound top-k queries retrieve less famous but highly correlated venues for the query.



Query	Inbound Top-5 Results		Outbound Top-5 Results	
	Rank	Place Name	Rank	Place Name
Yankee Stadium	1	Yankee Tavern	1	The Metropolitan Museum of Art
	2	Stan's Sports Bar	2	Madison Square Garden
	3	Billy's Sports Bar	3	The Central Park
	4	New York Penn Station	4	Grand Central Terminal
	5	Grand Central Terminal	5	Brooklyn Museum
Columbia University	1	Morningside Park	1	Central Park
	2	Seeley Mudd Hall	2	116th St/Columbia University MTA Subway
	3	Whole Foods Grocery	3	Newark Liberty International Airport
	4	Dinosaur Bar-B-Que	4	Grand Central Terminal
	5	Starbucks	5	Lincoln Tunnel

Inbound Top-K v.s. Outbound Top-K

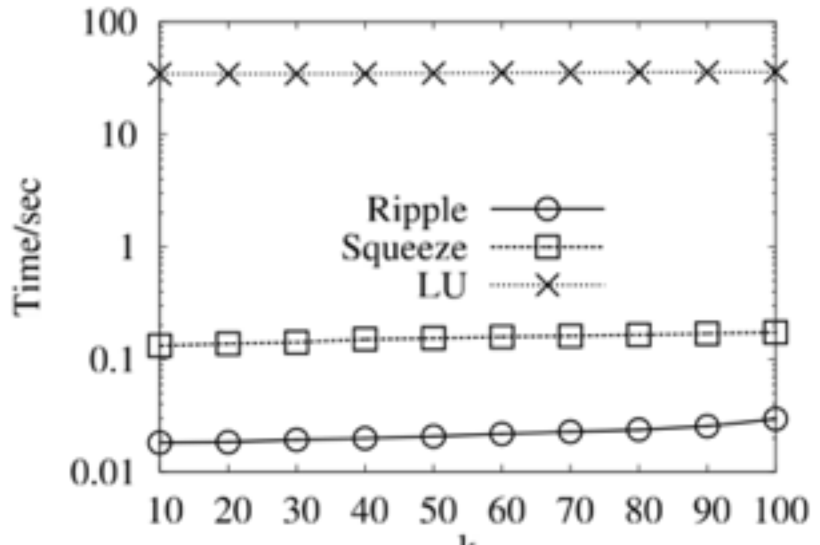
- Illustrative examples on Wikipedia
  - Similarly, outbound top-k queries retrieve prestigious pages
  - Inbound top-k queries retrieve more correlated pages



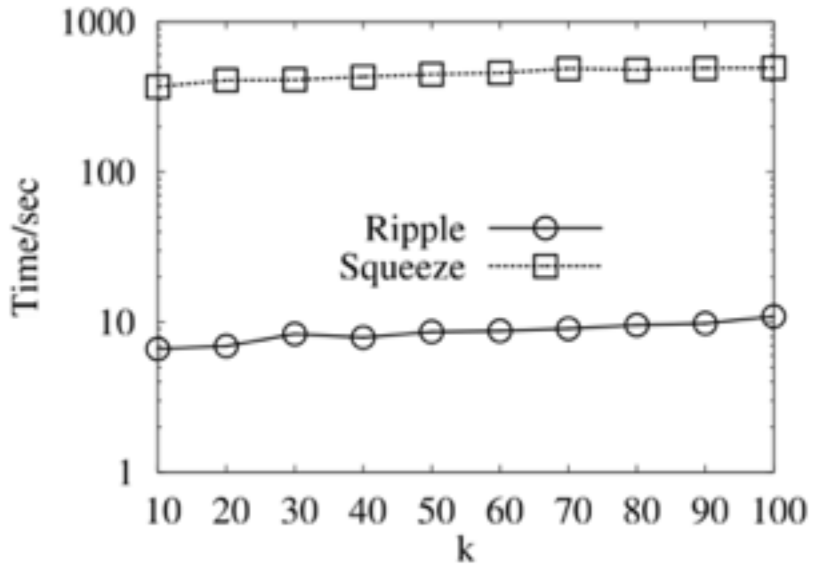
Query	Inbound Top-5 Results		Outbound Top-5 Results	
	Rank	Page Title	Rank	Page Title
Information Retrieval	1	IDF	1	Computer Science
	2	Index Term	2	Information Science
	3	Keyword (Internet Search)	3	Linguistics
	4	Precision and Recall	4	Association for Computing Machinery
	5	Recall (Information Retrieval)	5	Mathematics
Microsoft Office	1	Microsoft Excel	1	2006
	2	Microsoft Word	2	2007
	3	Microsoft Windows	3	2008
	4	Microsoft Office 2007	4	Microsoft
	5	Microsoft FrontPage	5	Microsoft Office 2007

Running Time

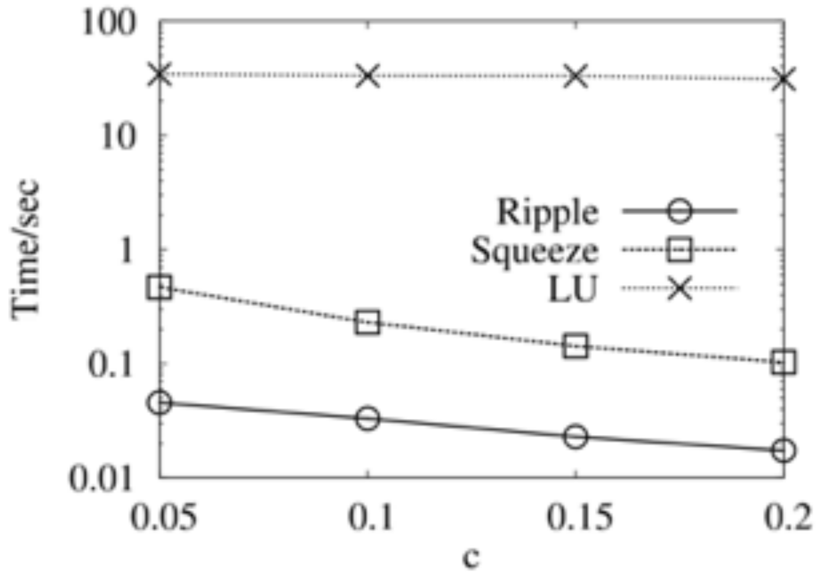
- Varying K and c:



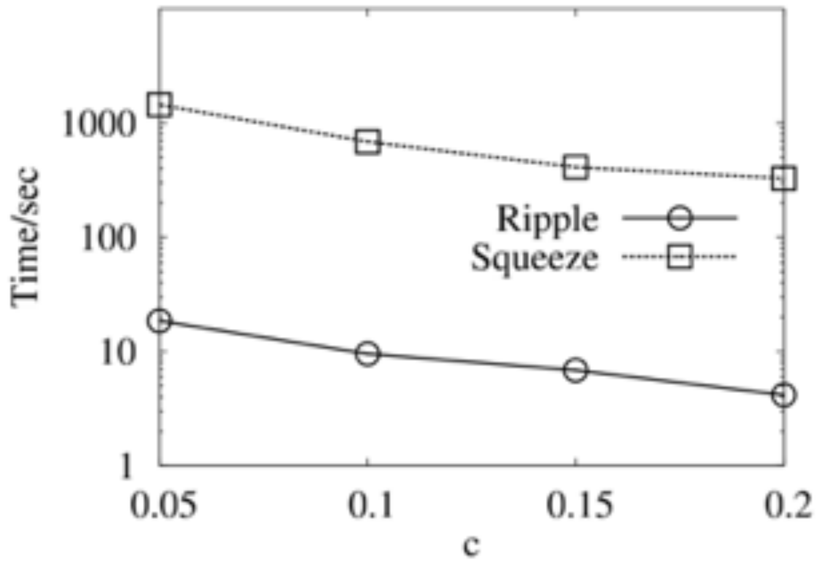
(a) Varying k on 4SQ.



(b) Varying k on Wiki.



(a) Varying c on 4SQ.



(b) Varying c on Wiki.

Observations

1. Both method are much faster than the baseline.
2. Ripple is suitable for extremely large graphs.

Outline

- Background
- Our Methods
- Experiments
- **Summary**

Summary

- We introduced a new top-K query based on random walk with restart.

- We proposed two efficient query processing methods
  - Squeeze: it performs power iteration to obtain approximate scores and the top-K results.
  - Ripple: it uses locality to obtain top-K results without accessing faraway nodes.

- The experiments show that the inbound top-K query retrieves interesting results, and the two methods have excellent efficiency.

Thank You!