UNIVERSITY of LOUISIANA LAFAYETTE

Lecture 8

TweetScore: Scoring Tweets via Social Attribute Relationships for Twitter Spammer Detection

Xu Yuan
University of Louisiana at Lafayette

Social Network

- **The online social network (OSN) is indispensable in our daily life.**
  - Facebook, 2.4 billion monthly active users (MAU).
  - In Twitter, around 6,000 tweets per second.
  - In Tiktok, 1 billion MAU spend around 52 minutes per day.

Spammer in Social Network

- **Pervasively annoying users, grossly detrimental to social network.**
    - Degrading the quality of user experience
    - Stealing sensitive information
    - Economic loss
    - Changing political opinions

Collecting and classifying spammers have been the critical problem!

Challenges

challenge

Spammers are hidden among benign users.

Low accuracy

Unrealistic to process the entire dataset.

Low efficiency

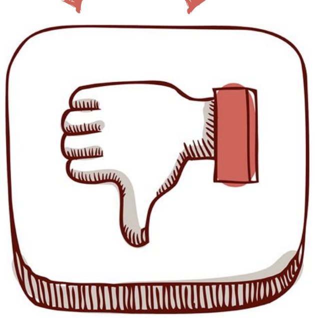
State-of-the-art Solutions

Blindly Spam Collection

- Classifying the spam messages from large-scale network contents or the social relationships.
- Time-consuming and inefficient.

Honeypot-based Solution

- Build honeypots to lure spammers.
- High deployment overhead, low attribute variability
  low deployment flexibility, low network scalability



Graph-based Solution

- Analyzing user relationships.
- Overlook attribute relationships.

Our Goals

- **Propose an effective solution to monitor and capture spammers.**
  - Monitoring users having potentials of attracting spammers.
  - Take advantages of users' diversity.

- **Design a novel solution to classify spams.**
  - User activities.
  - User attribute relationships.
  - User relationships.

Outline

- **Pseudo-honeypot Monitoring System**
- **TweetScore Spam Classification Solution**
- **Experiment Results**

Pseudo-honeypot Spam Monitoring System

Pseudo-honeypot

- **Screen normal users from a pool of normal users that are more vulnerable to spammers.**
  - Identify features meeting spammers' taste
  - Select users having such attributes

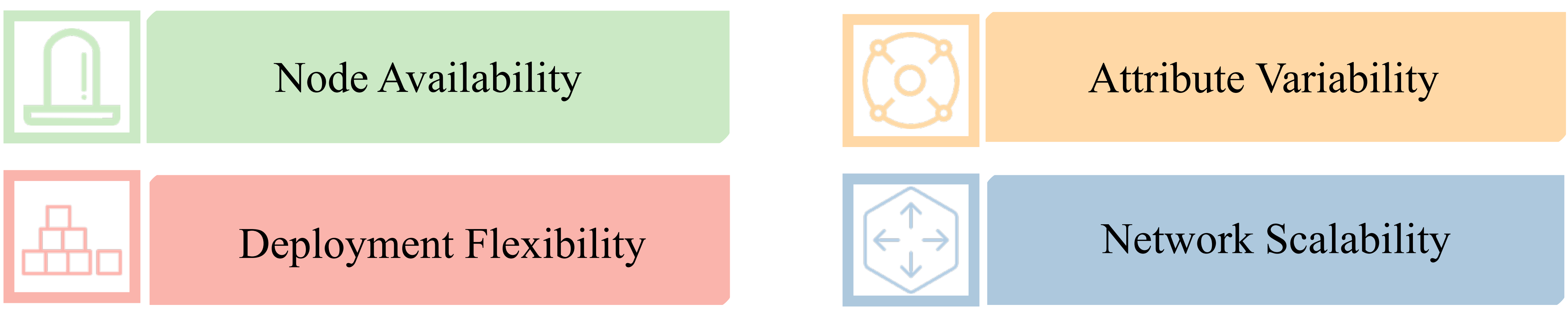
- **Harness such normal users serving as the pseudo-honeypot.**

- **Pseudo-honeypot can monitor their streaming posts and behaviors patterns.**
  - Having a higher probability of including spam messages.

Pseudo-honeypot vs. Honeypot

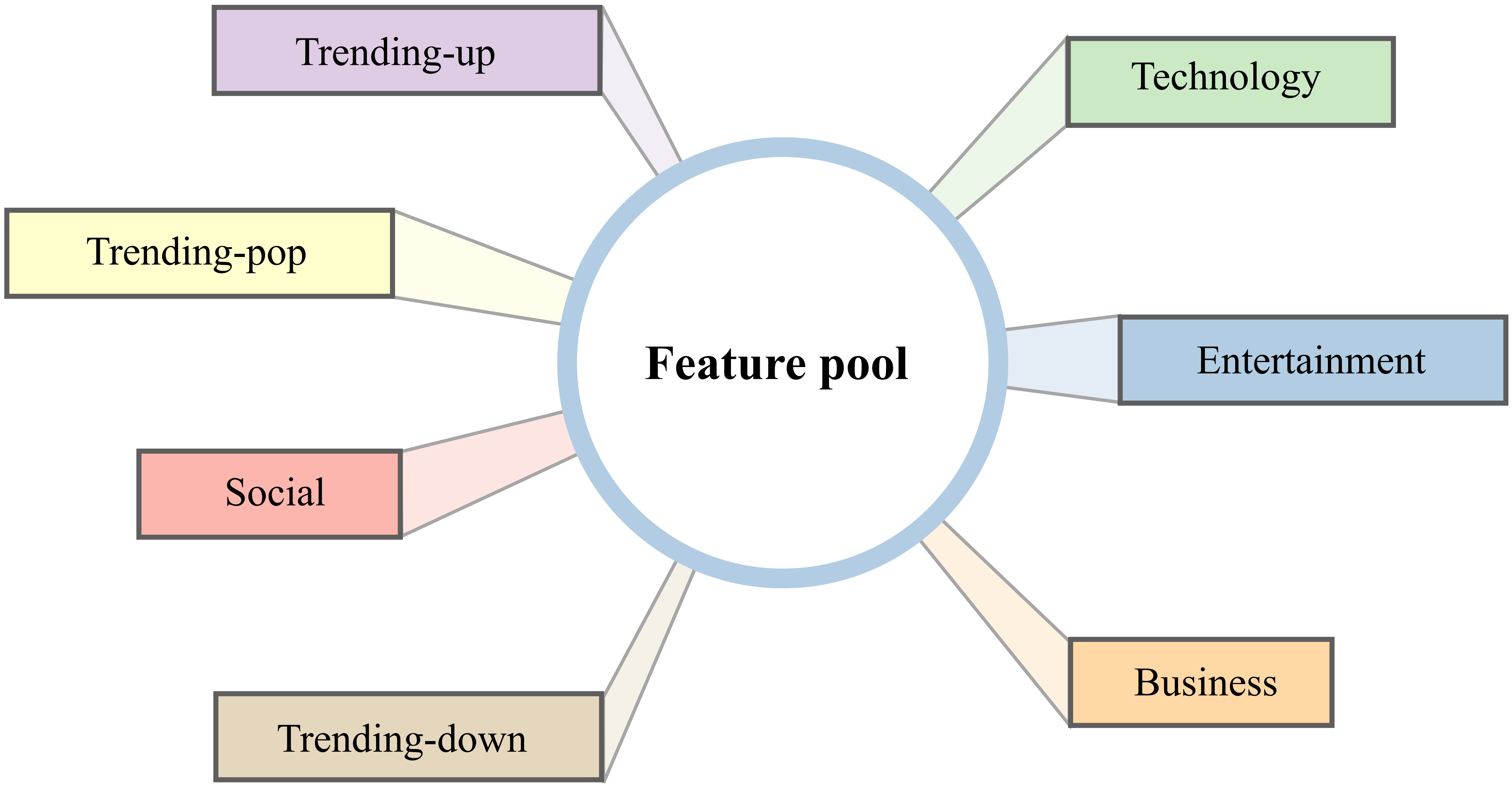
- **Similar function in attracting and trapping spammers.**

- **Possess salient advantages as follows:**



Pseudo-honeypot Construction

- **Selecting effective features for constructing Pseudo-honeypot.**



Pseudo-honeypot Monitoring

- **Streaming API construct and monitor Pseudo-honeypots.**

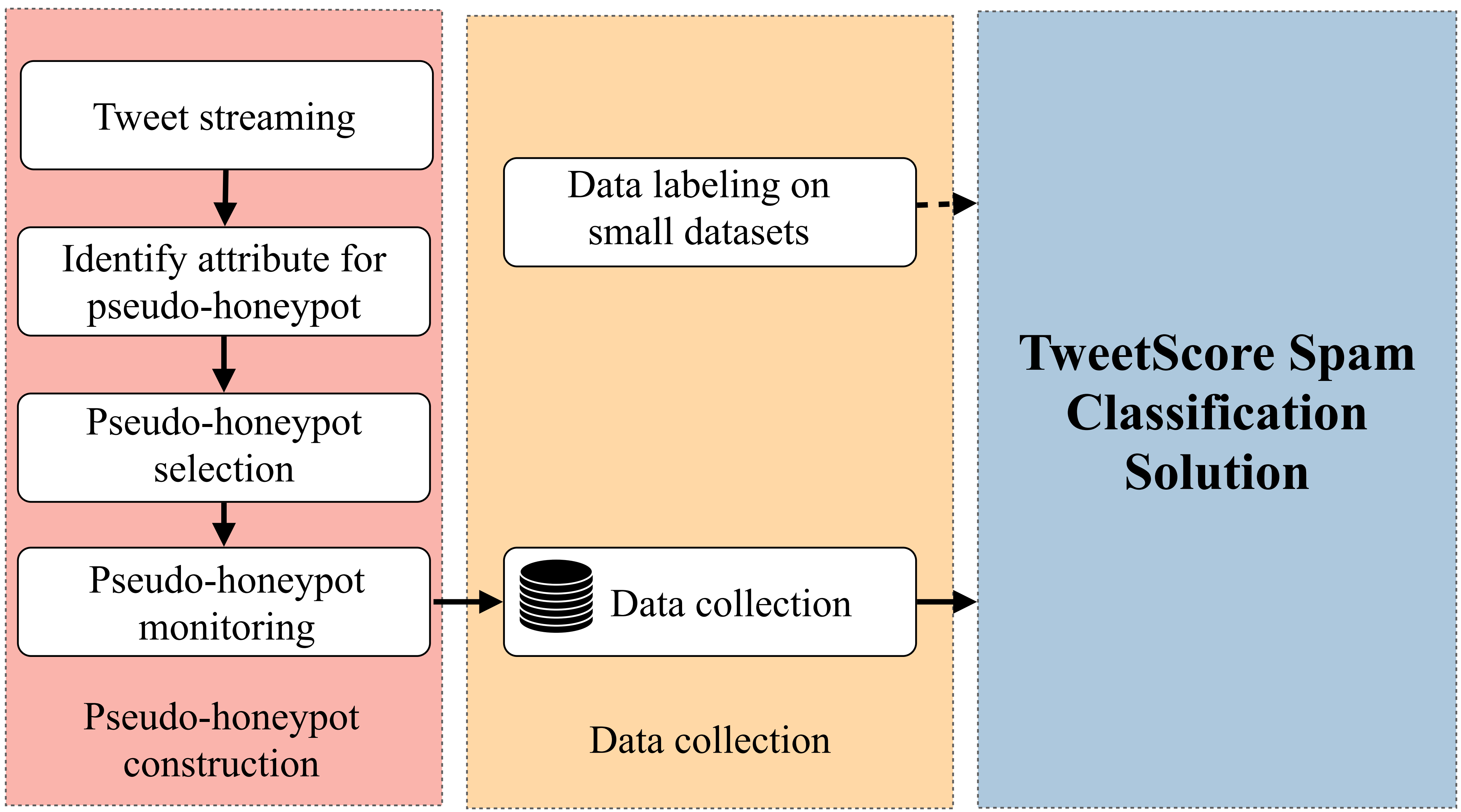
Screen the user that possess the selected features.

Monitor Pseudo-honeypot activities.

Shift Pseudo-honeypot among candidates.

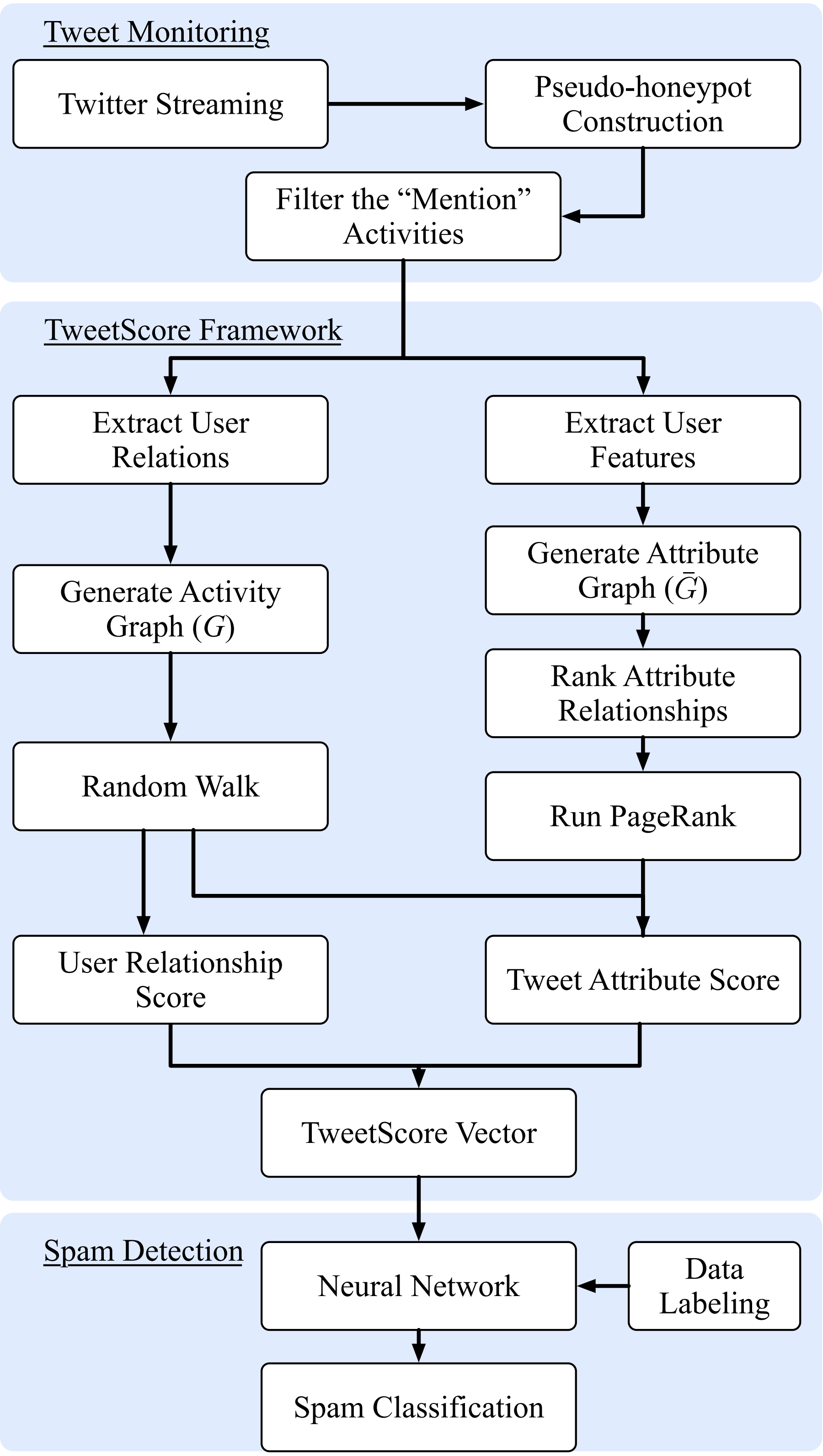
Significantly reduce the spammer detection workload.

Pseudo-honeypot Framework



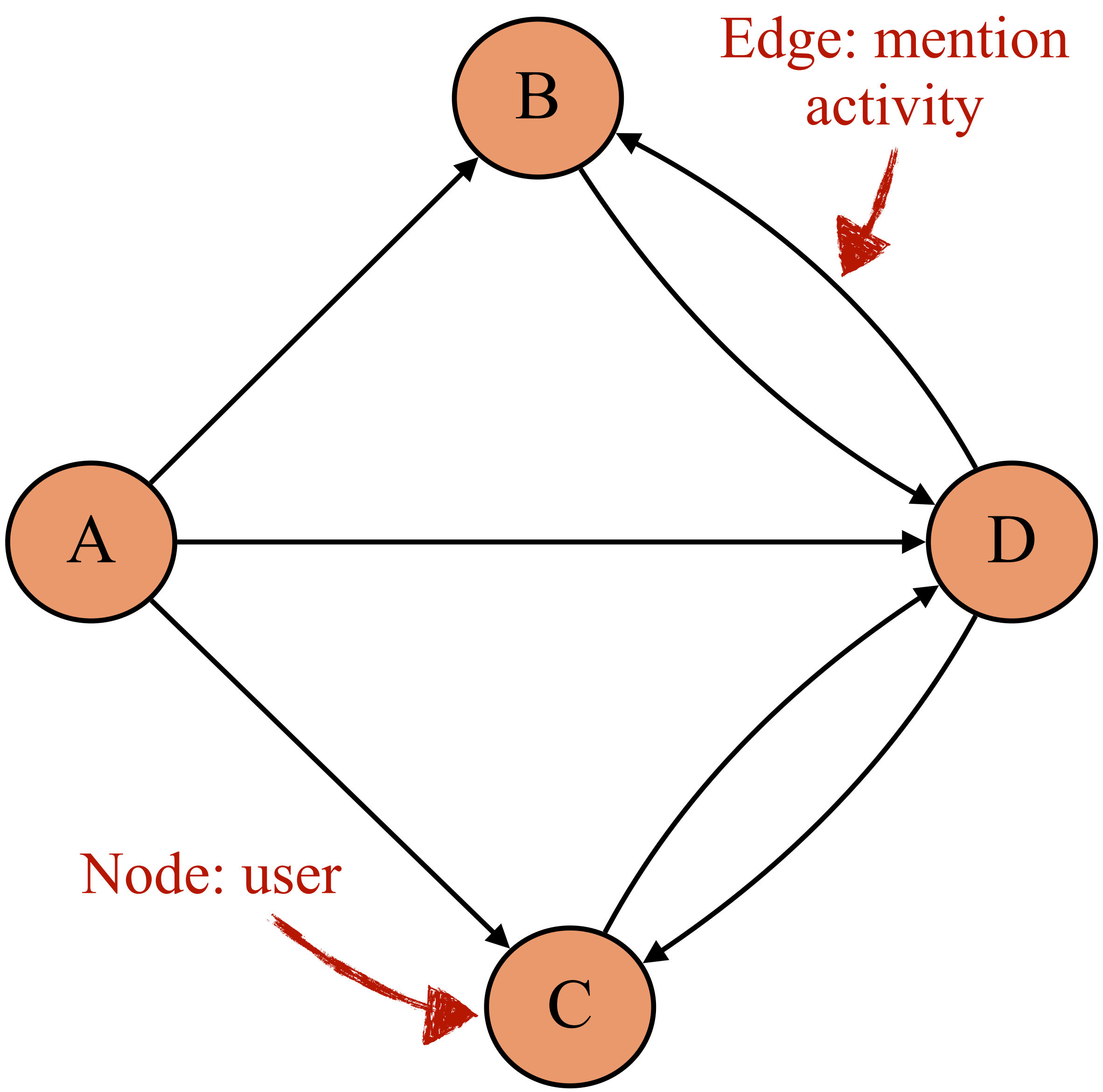
TweetScore Spam Classification Solution

TweetScore Framework



Activity Graph

- **Directed weighted graph denotes mention activities.**



Edge Weight

$$w(i,j) = (1 + \eta \cdot \boxed{Sim(i,j)})\boxed{M(i,j)}$$

Node similarity

Mention frequency

Node Similarity

$$Sim(A,D) = \frac{\vec{A} \cdot \vec{D}}{||\vec{A}|| \cdot ||\vec{D}||}$$

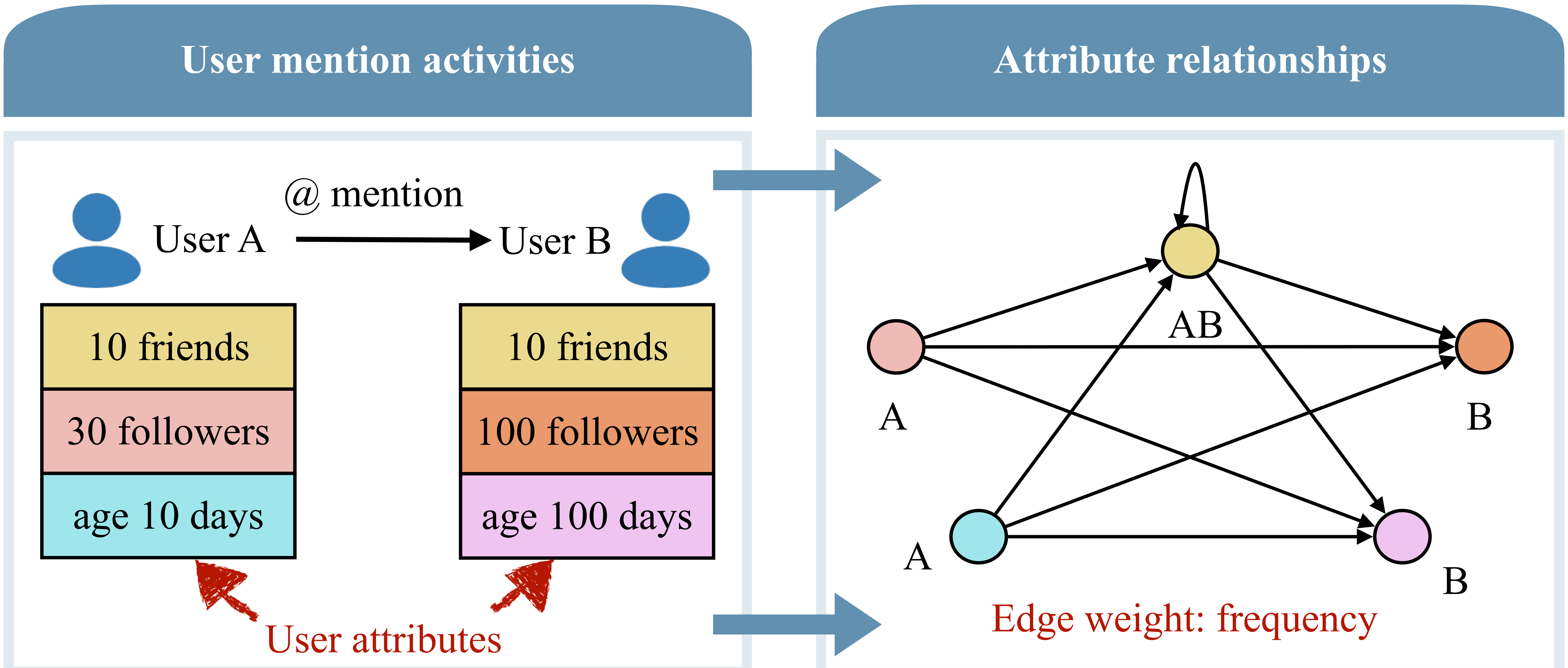
D mention B and C

$$= \frac{<0,1,1,1> \cdot \boxed{<0,1,1,0>}}{||<0,1,1,1>|| \cdot ||<0,1,1,0>||}$$

$$= 0.8165$$

Attribute Graph

- **Directed graph model attribute relationships.**



How to score relationships between any two attributes?
How to score attributes?

Scoring Attribute Relationships

- **Score the relationships between any two attributes.**

	a	*b*	*c*	*d*
a	4		1	
b		3		
c	2			6
d		5	4	9

- Attribute graph transform to **sparse matrix A**.
- Each entry denotes the score of attribute relationship.
- **UV-Decomposition** can be used to predict.

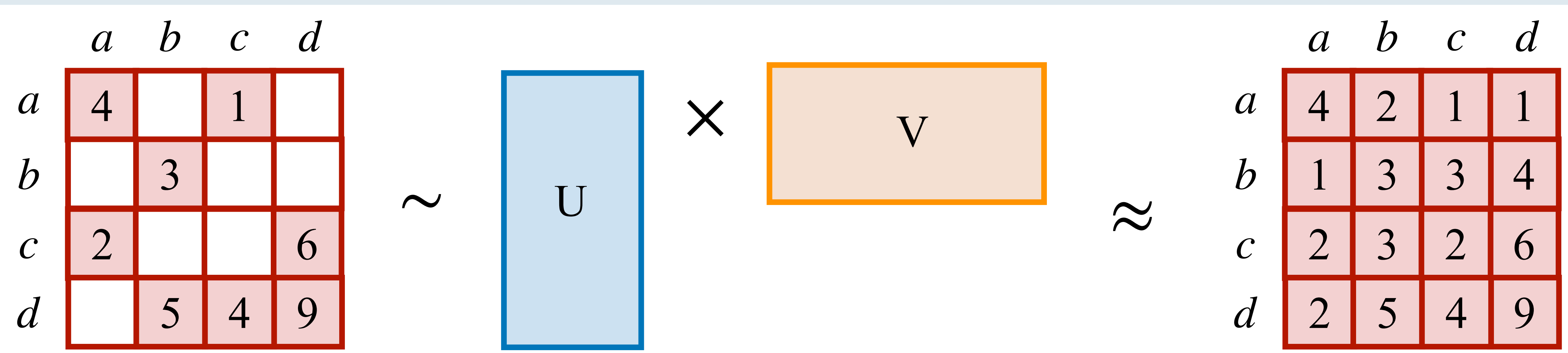
UV-Decomposition

$$\begin{array}{c|cccc} & a & b & c & d \\ \hline a & 4 & & 1 & \\ b & & 3 & & \\ c & 2 & & & 6 \\ d & & 5 & 4 & 9 \end{array} \sim \boxed{U} \times \boxed{V} \approx \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 4 & 2 & 1 & 1 \\ b & 1 & 3 & 3 & 4 \\ c & 2 & 3 & 2 & 6 \\ d & 2 & 5 & 4 & 9 \end{array}$$

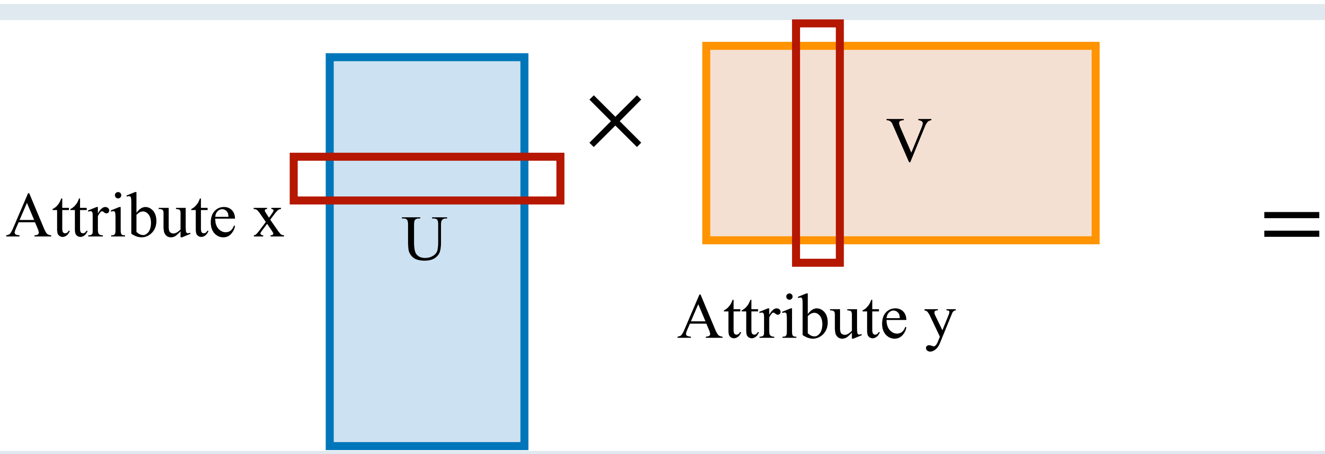
Scoring Attribute Relationships

- **Score the relationships between any two attributes.**

UV-Decomposition



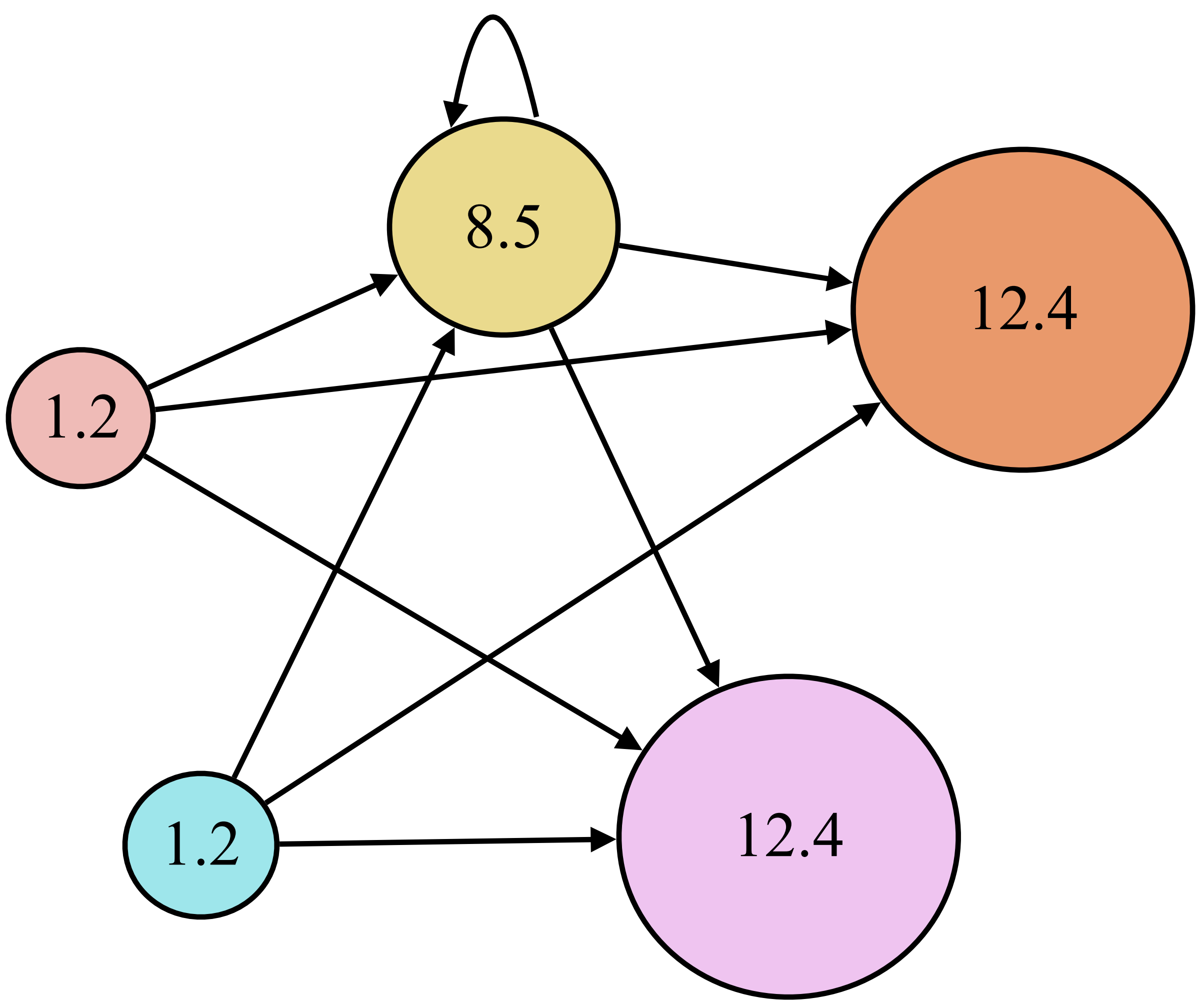
Attribute Inference



Inferred score for attribute x to attribute y. The higher, the more importance between two attributes.

Scoring Attributes

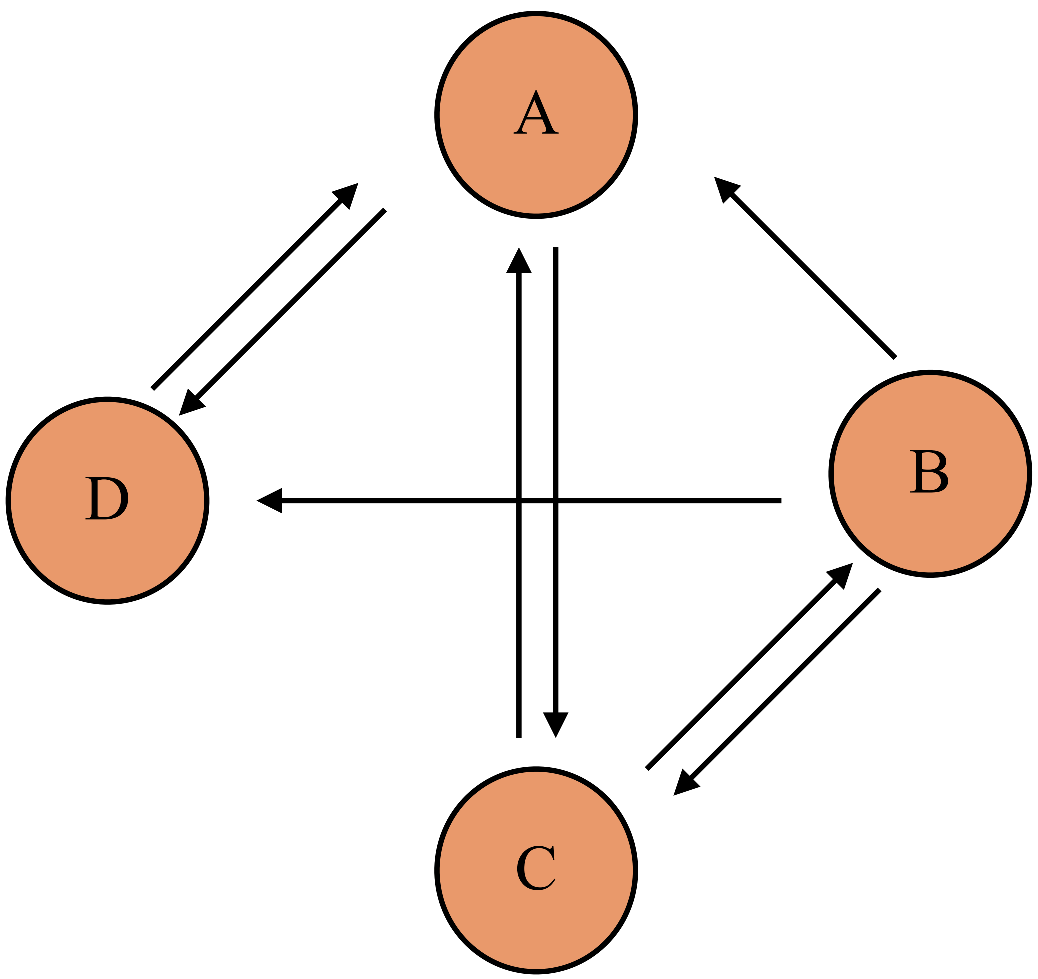
- **Score attribute by using PageRank.**



- Run PageRank on attribute graph.
- Attributes have different potentials of attracting spammer's interest.

PageRank

- **PageRank Algorithm**

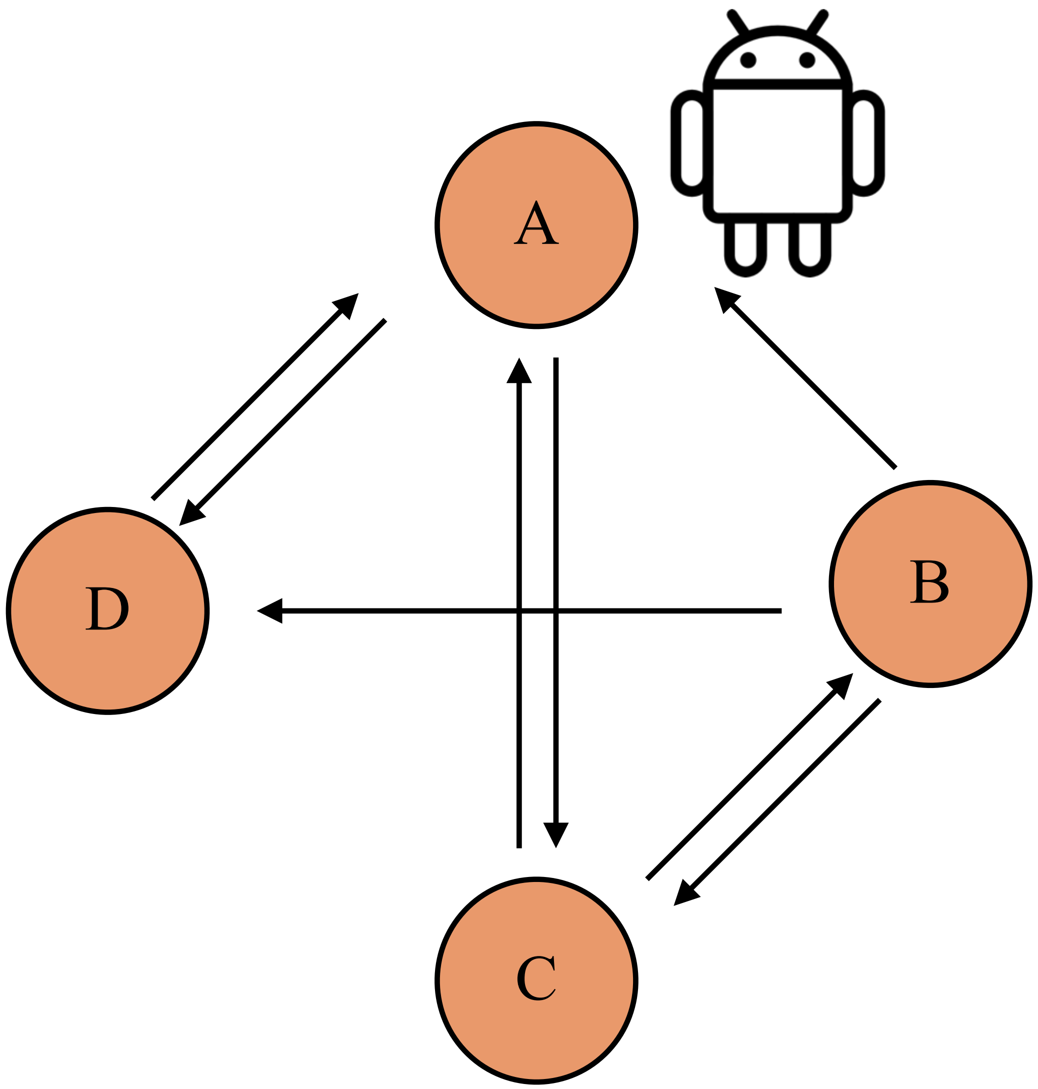


		From			
		A	B	C	D
To	A	0.00	0.33	0.50	1.00
	B	0.00	0.00	0.50	0.00
	C	0.50	0.33	0.00	0.00
	D	0.50	0.33	0.00	0.00

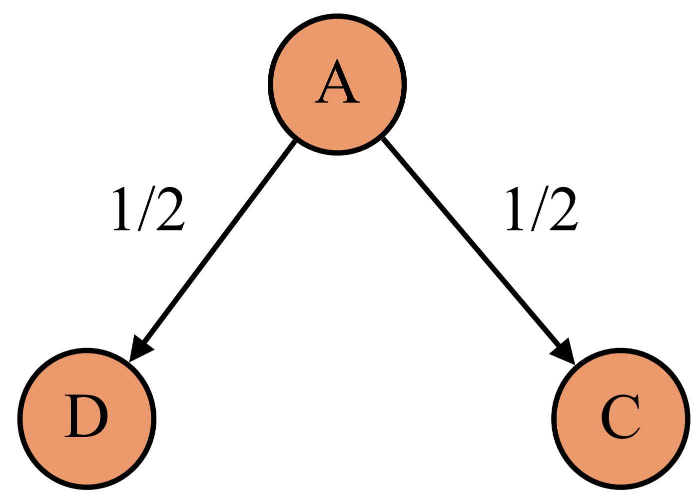
From Node A to Node C: 50%
From Node A to Node D: 50%

PageRank

- **PageRank Algorithm (2)**

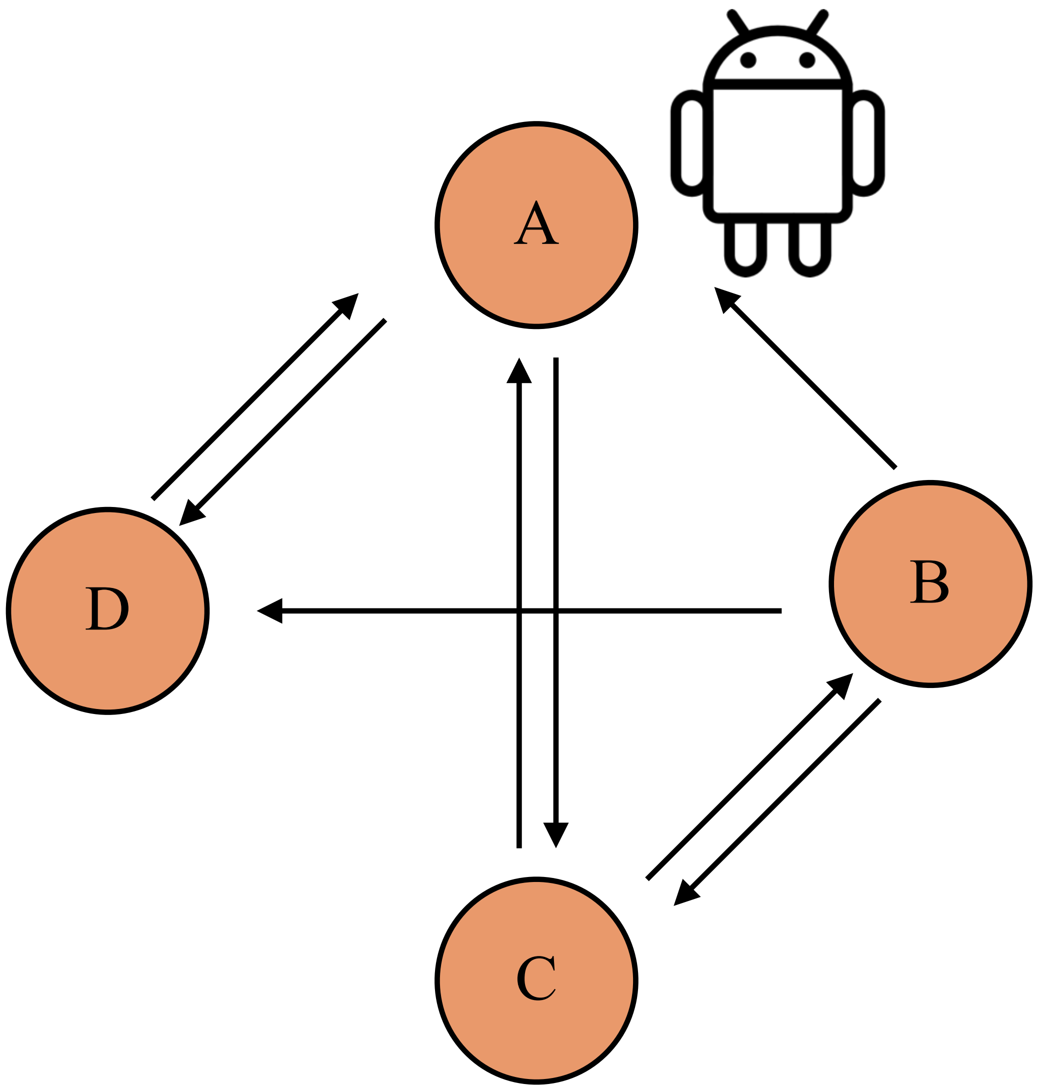


Now we imagine that if there were a bot which will follow all the outgoing links, what will be the total time spent by this bot on each of these nodes.

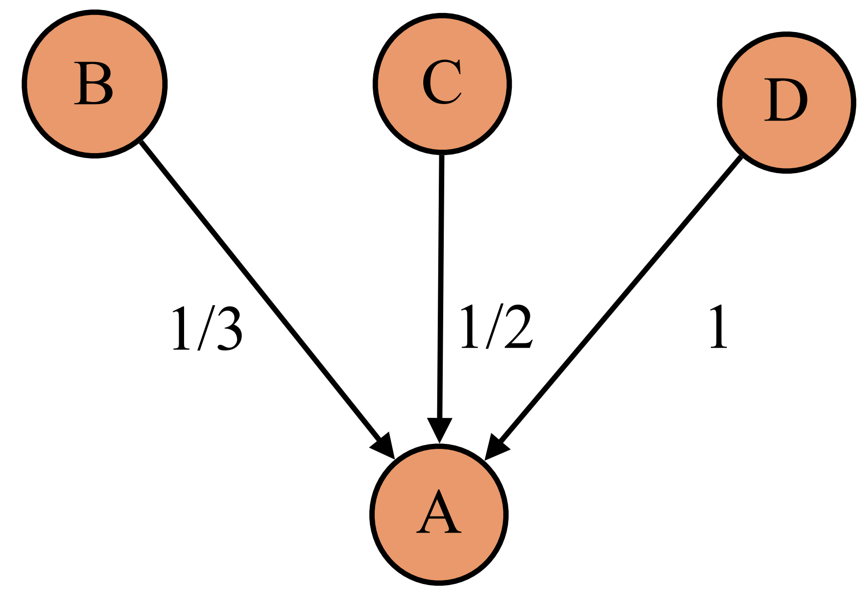


PageRank

- **PageRank Algorithm (2)**



The probability for the bot to go to node A

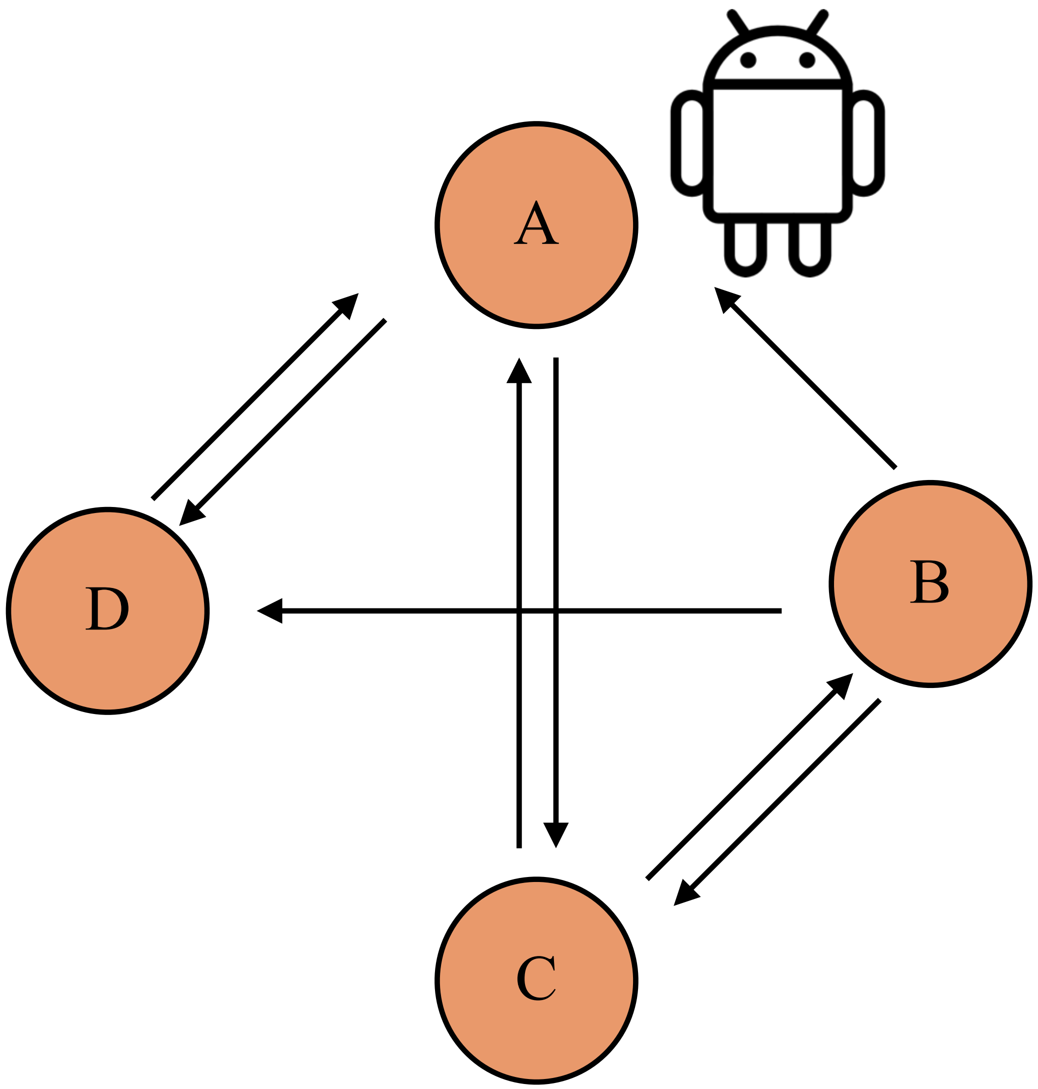


$$P(A) = \frac{1}{3}P(B) + \frac{1}{2}P(C) + P(D)$$

Let's guess the initial probability is 25% for each of the node.

PageRank

- **PageRank Algorithm (3)**



Step 1:

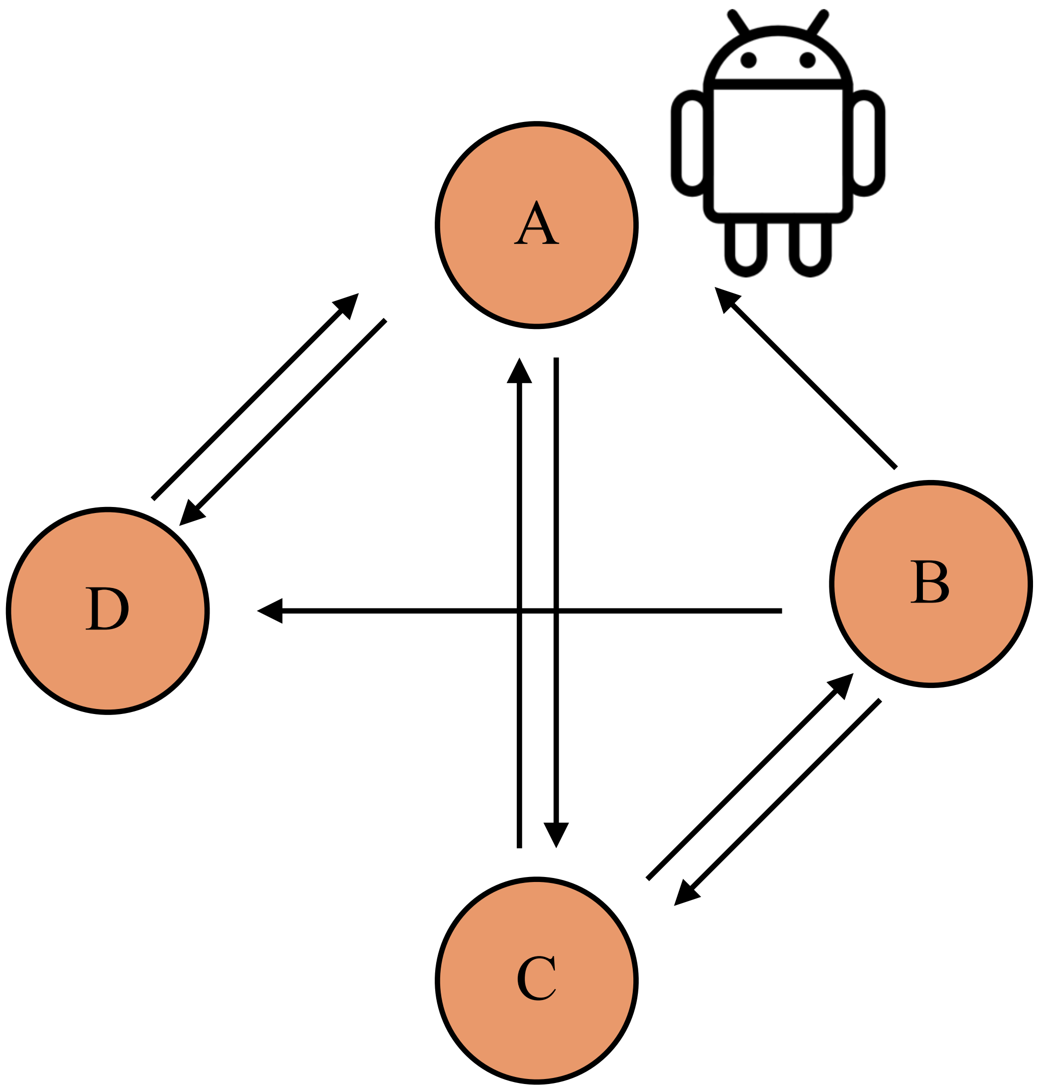
Let's guess the initial probability for the bot on each node is 25%.

$$\begin{bmatrix} 0.00 & 0.33 & 0.50 & 1.00 \\ 0.00 & 0.00 & 0.50 & 0.00 \\ 0.50 & 0.33 & 0.00 & 0.00 \\ 0.50 & 0.33 & 0.00 & 0.00 \end{bmatrix} \begin{bmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{bmatrix} = \begin{bmatrix} 0.458 \\ 0.124 \\ 0.208 \\ 0.208 \end{bmatrix}$$

Node A has 3 inward edges, the probability on node A increase. Node B has only 1 inward edge, the probability on node B decrease

PageRank

- **PageRank Algorithm (4)**



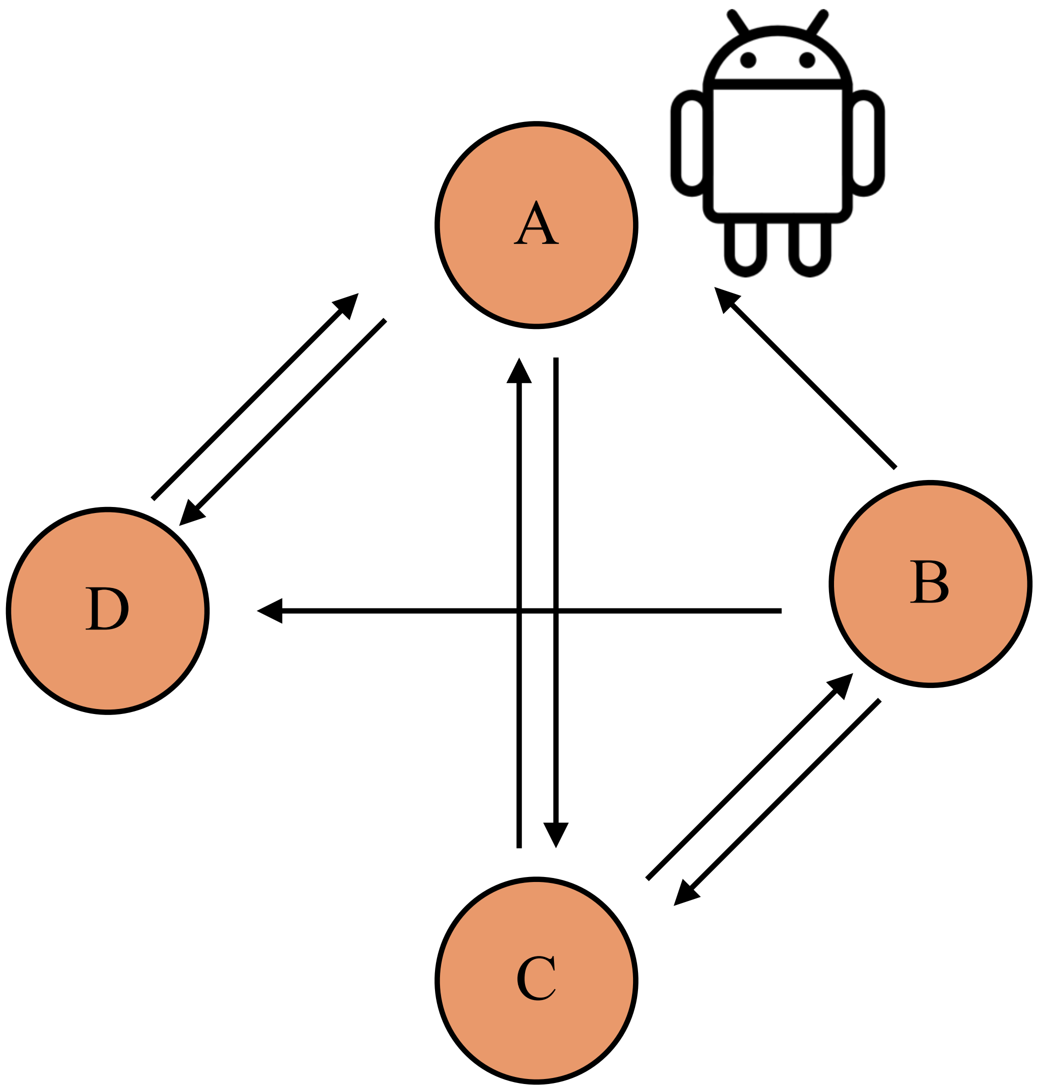
Step 2:

Let’s keep updating:

$$\begin{bmatrix} 0.00 & 0.33 & 0.50 & 1.00 \\ 0.00 & 0.00 & 0.50 & 0.00 \\ 0.50 & 0.33 & 0.00 & 0.00 \\ 0.50 & 0.33 & 0.00 & 0.00 \end{bmatrix} \begin{bmatrix} 0.458 \\ 0.124 \\ 0.208 \\ 0.208 \end{bmatrix} = \begin{bmatrix} 0.354 \\ 0.104 \\ 0.271 \\ 0.271 \end{bmatrix}$$

PageRank

- **PageRank Algorithm (4)**



Step 1000:

Let's keep updating:

$$\begin{bmatrix} 0.00 & 0.33 & 0.50 & 1.00 \\ 0.00 & 0.00 & 0.50 & 0.00 \\ 0.50 & 0.33 & 0.00 & 0.00 \\ 0.50 & 0.33 & 0.00 & 0.00 \end{bmatrix} \begin{bmatrix} 0.40 \\ 0.12 \\ 0.24 \\ 0.24 \end{bmatrix} = \begin{bmatrix} 0.40 \\ 0.12 \\ 0.24 \\ 0.24 \end{bmatrix}$$

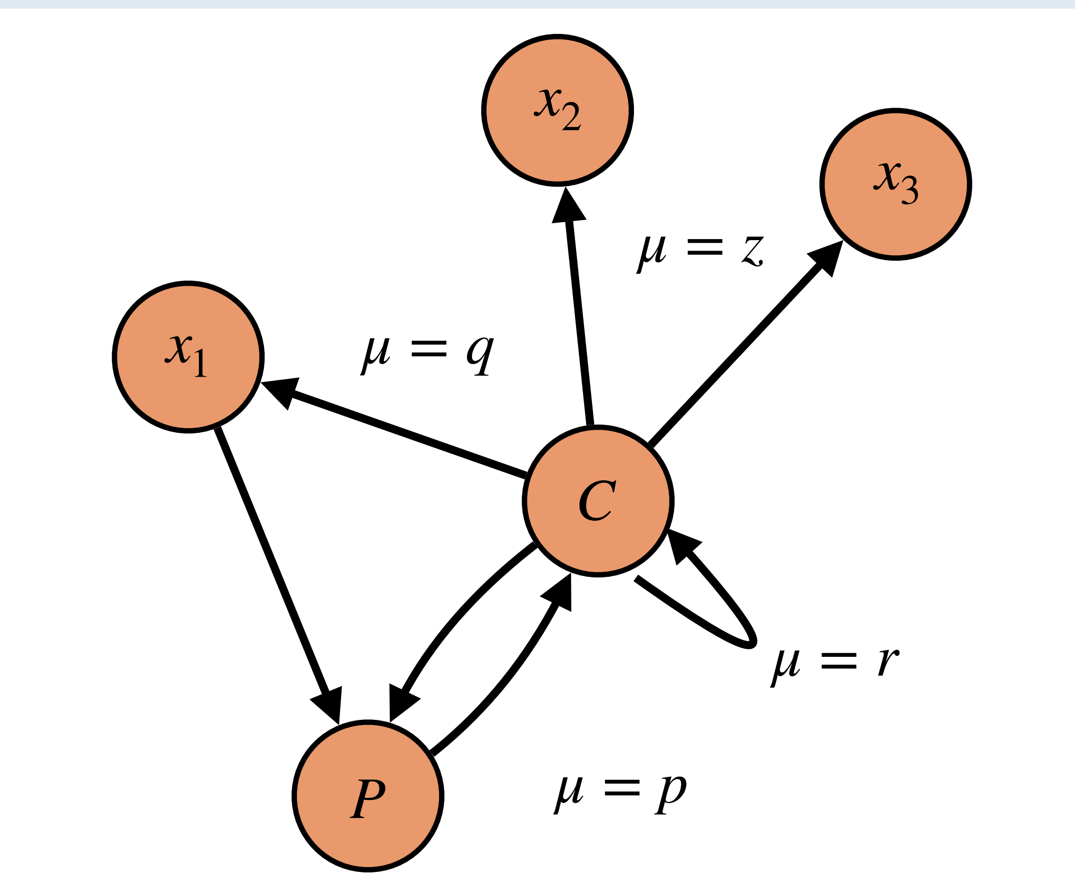
It is stable !

The meaning of PageRank score is the importance of the node in a graph.

Scoring Tweets (1)

- **Evaluate tweet from most relevant user.**

Random walk on activity graph



Notation

C	Current node	r	Self-directed score
P	Previous node	p	Mutual behavior score
x_1	Inward node	q	Inward score
x_2	Outward node	z	Outward score

Node Selection

$$P_{i,j} = \frac{\mu_{i,j} \boxed{w_{i,j}}}{\sum_{\boxed{j' \in Nr(i)}} \mu_{i,j'} w_{i,j'}}$$

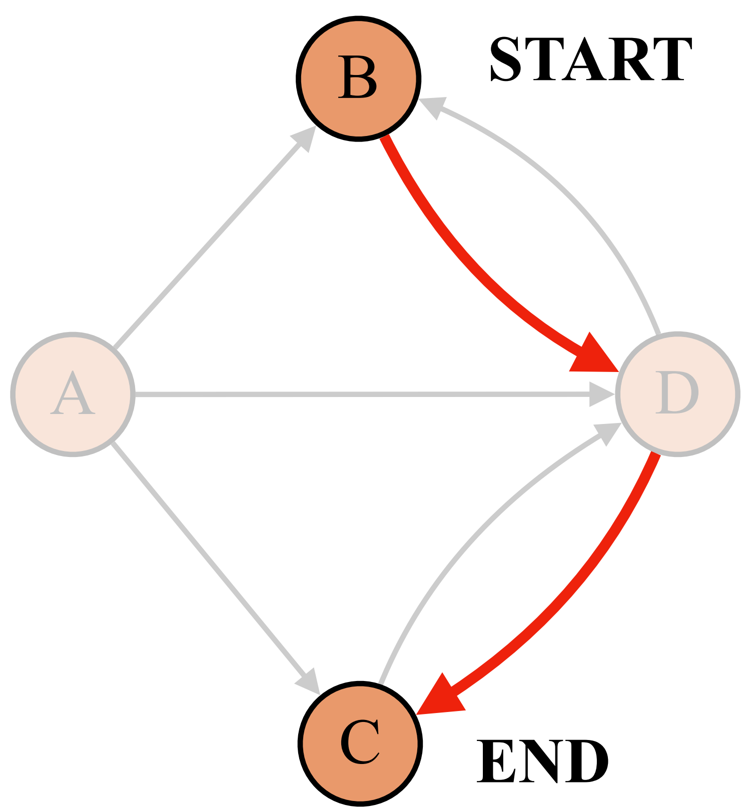
Edge wight

Neighborhoods

Scoring Tweets (2)

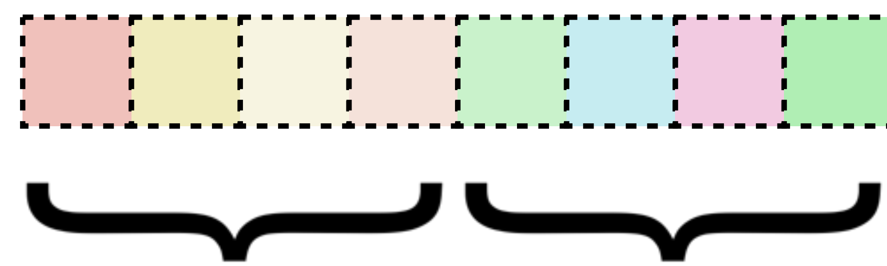
- **Generate Tweet Vector.**

An Example, tweet B @ D



Random walk start from node (B) and end at node (C).

Tweet Vector S

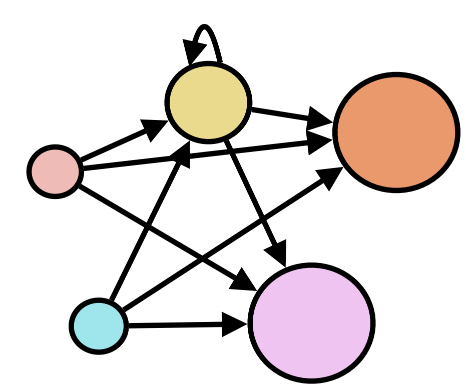


$$S[j] = \frac{B_j + C_j}{2}$$

$$S[j] = B_j - C_j$$

the j-th attribute

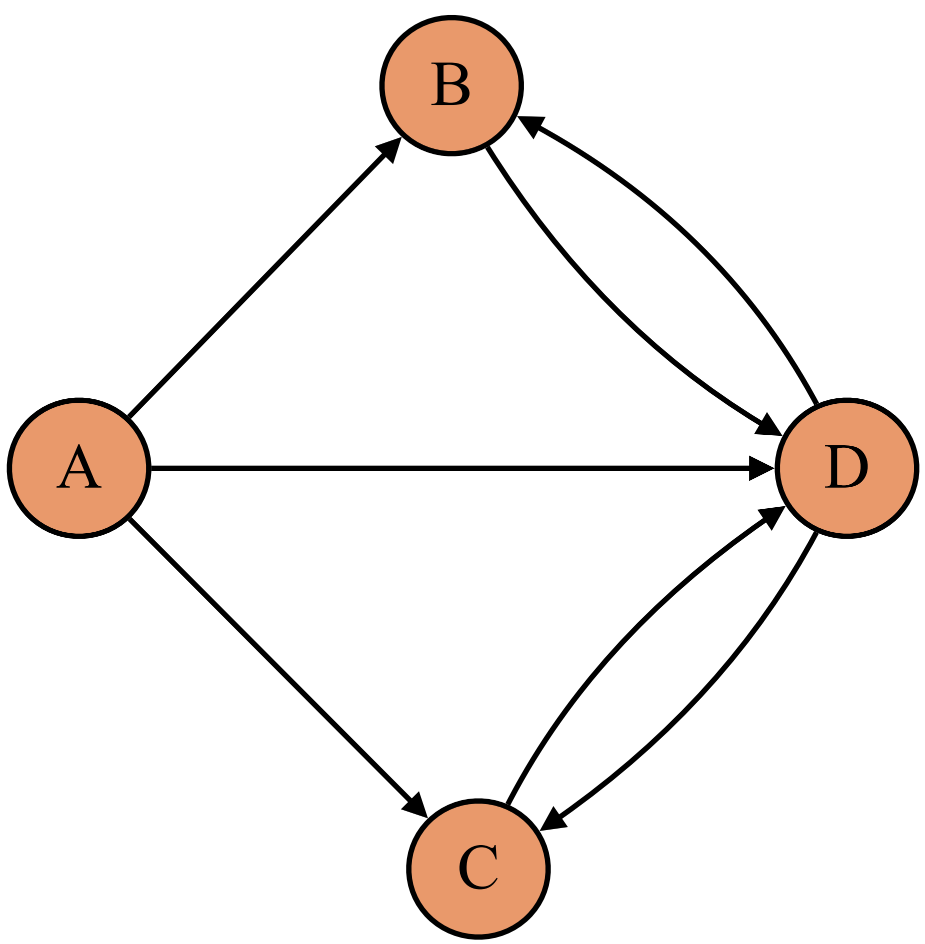
Check attribute score from attribute graph



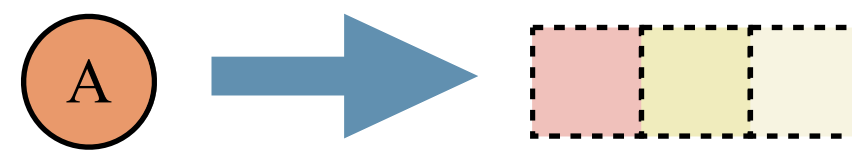
Scoring User' Dependence Relationships

- **Score** **User Dependence Vector** **by using random walk path.**

User activity graph

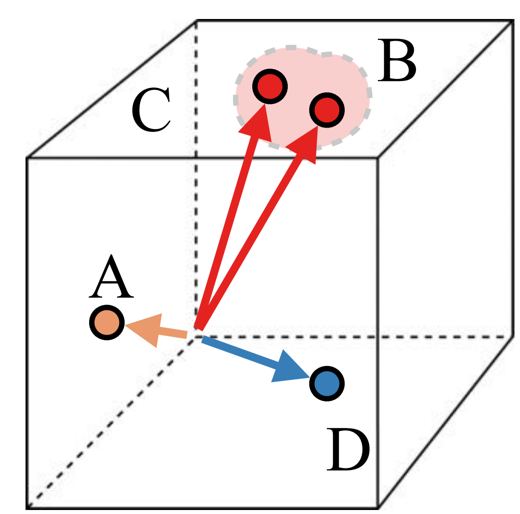


High-dimensional dense vector



Represent more fine-grained and affluent features

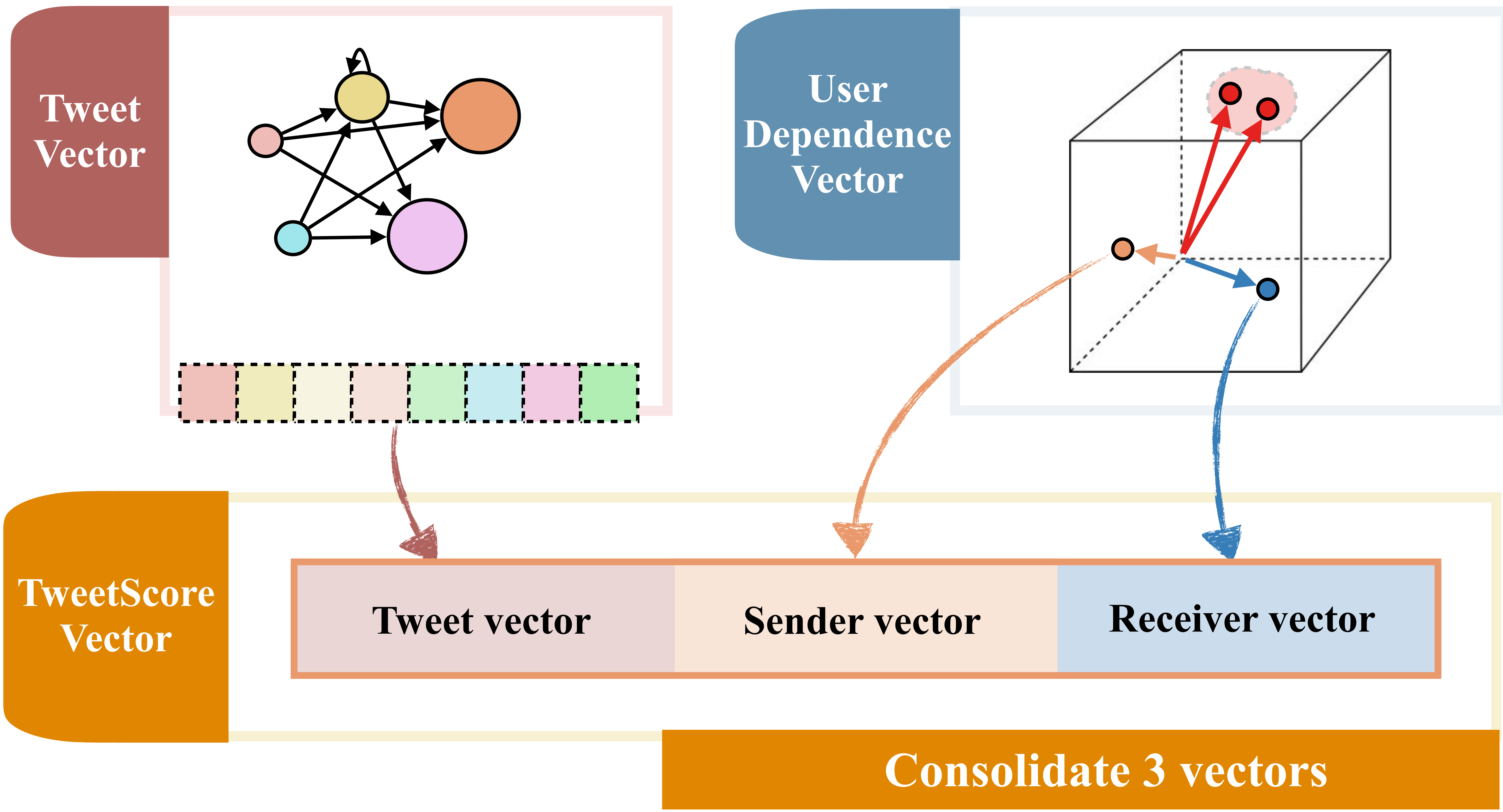
Vector illustration



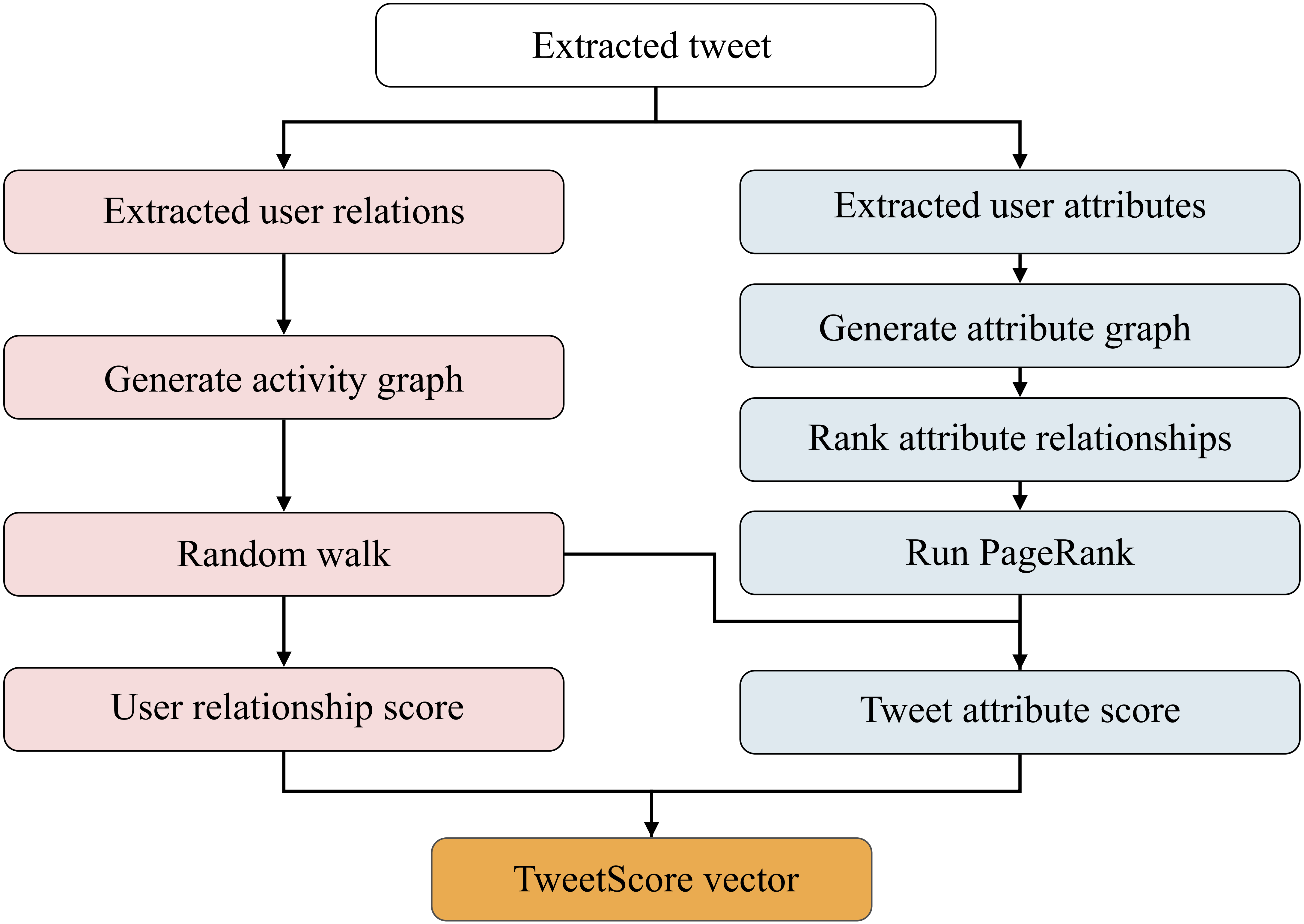
Similar user will get similar vectors.

TweetScore Vector

- Consolidate Tweet Vector, sender Dependence Vector, receiverDependence Vector.

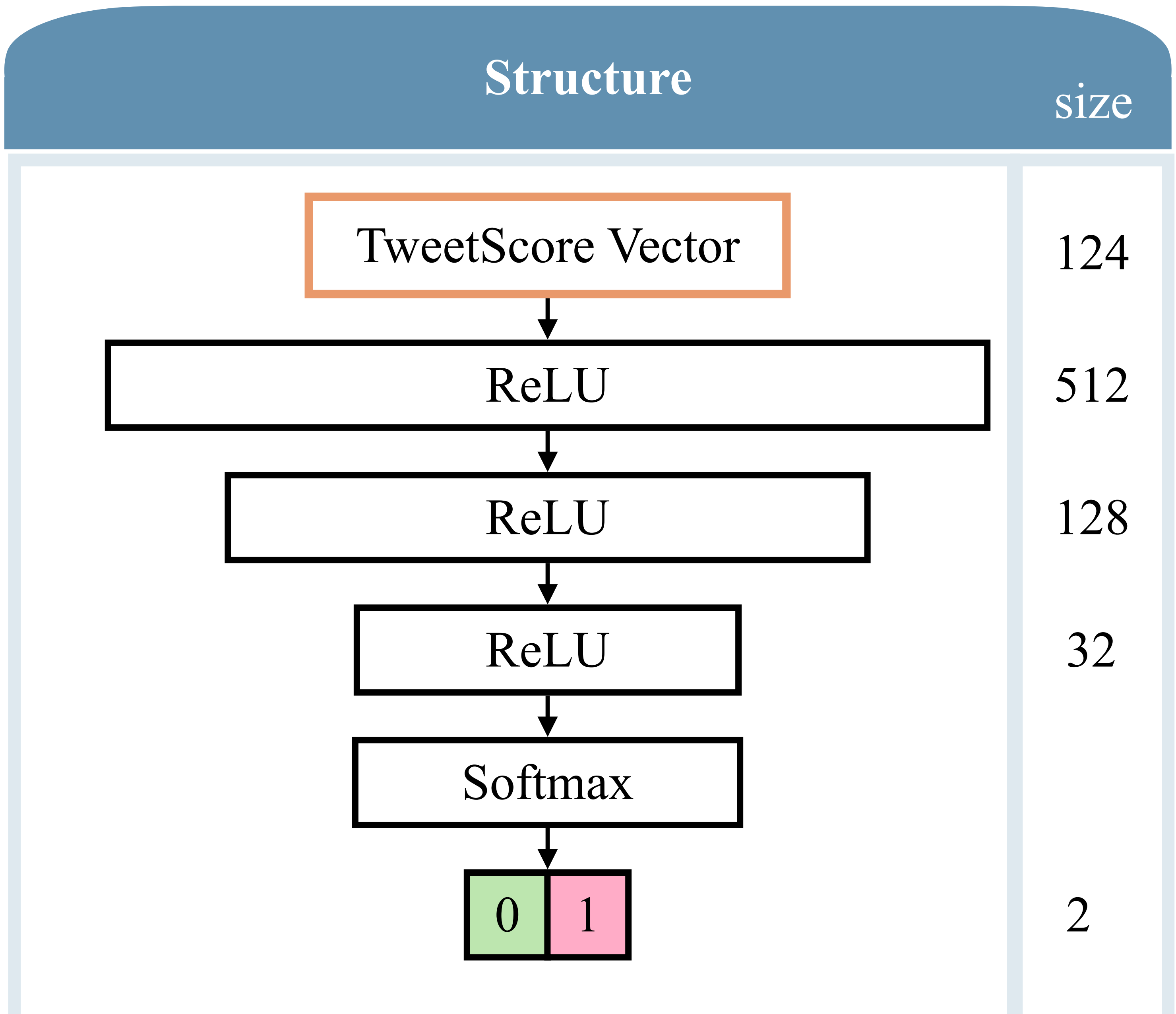


TweetScore Framework



Neural Network Model

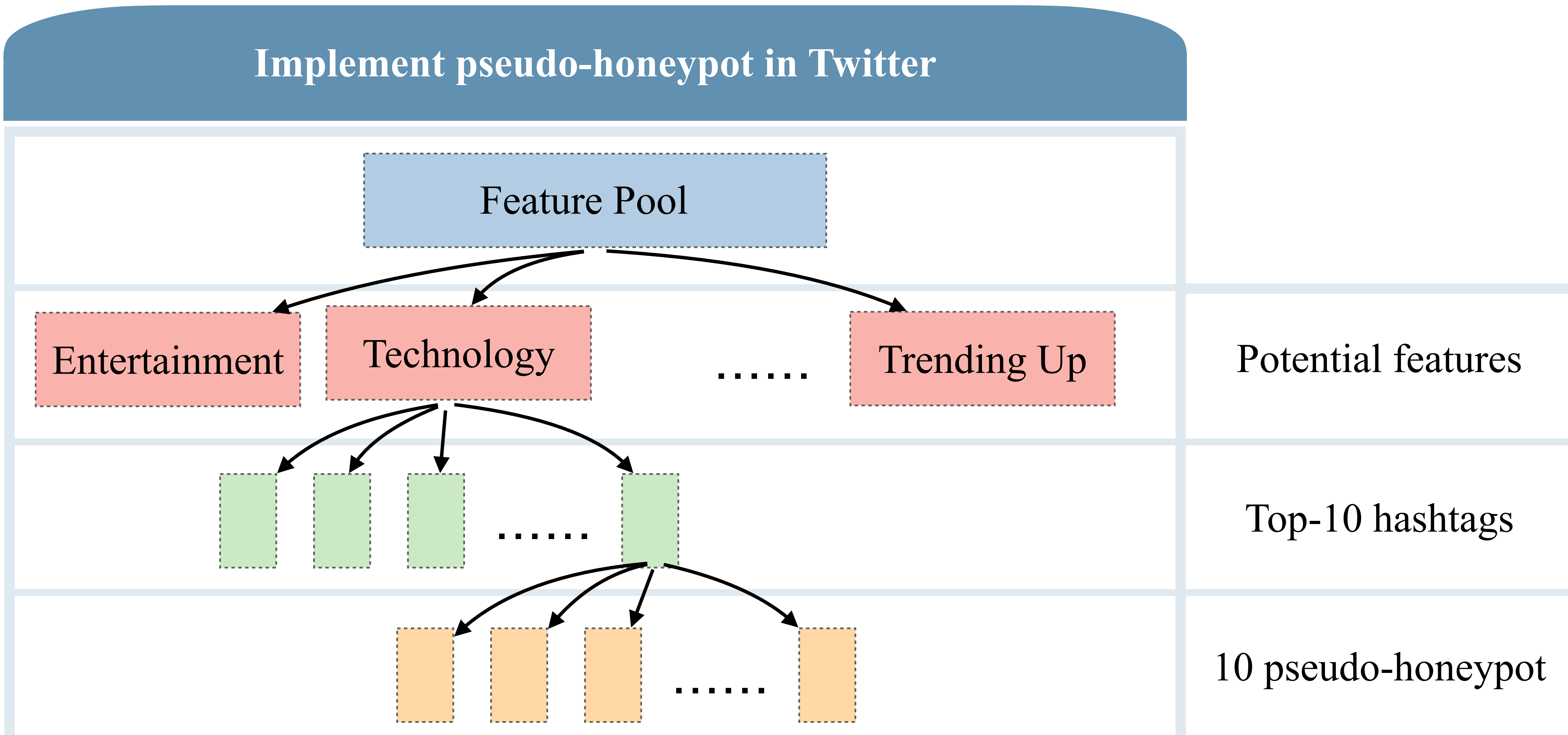
- **We employ the neural network to learn the affluent information**



Training process	
1	Extract TweetScore Vector.
2	Label a small set of ground truth dataset.
3	Training neural network on the ground truth dataset.
4	Label all other collected data by the trained classifier.

Implementation

- **Pseudo-honeypot are constructed by using hashtag-based and trending-based features.**



Performance

- Performance of 100-hour data captured by pseudo-honeypot

TABLE. 10-fold cross-validation

Classifier	Precision	Accuracy	Recall	F1-macro
AB	0.855	0.872	0.797	0.831
GB	0.811	0.926	0.835	0.852
k-NN	0.760	0.901	0.722	0.820
SybilSCAR	0.661	0.454	0.436	0.445
Chen6M	0.976	0.955	0.852	0.927
TweetScore	0.989	0.967	0.914	0.946

Higher AUC means better performance.

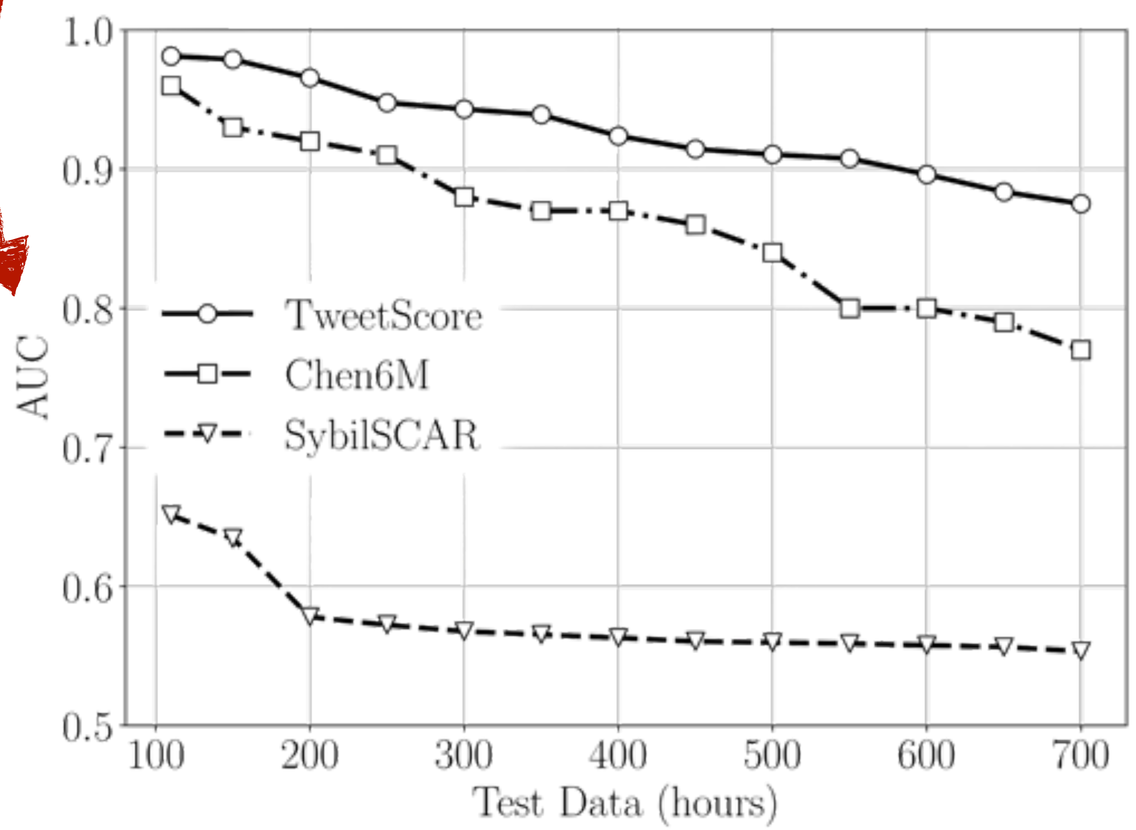


Figure. The AUC curves on the 600-hour data.

- Pseudo-honeypot report collected tweets in every 10 hours.
- 600-hour online testing

Pseudo-honeypot vs. other methods

Hit Ratio

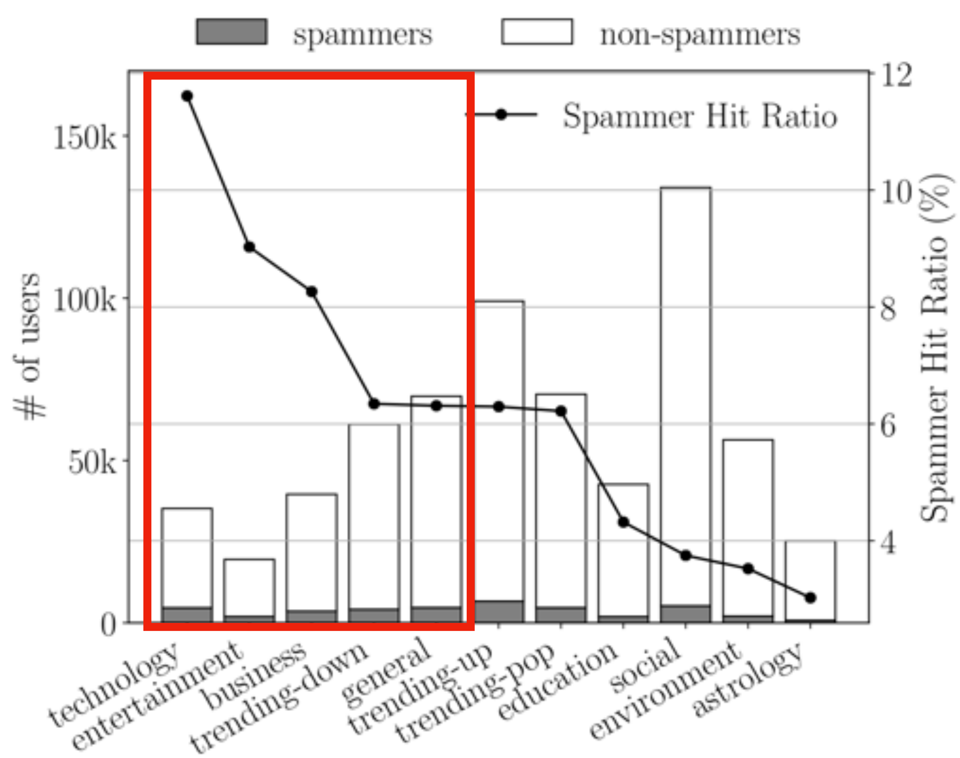


Figure. Number of spammers and hit ratio.

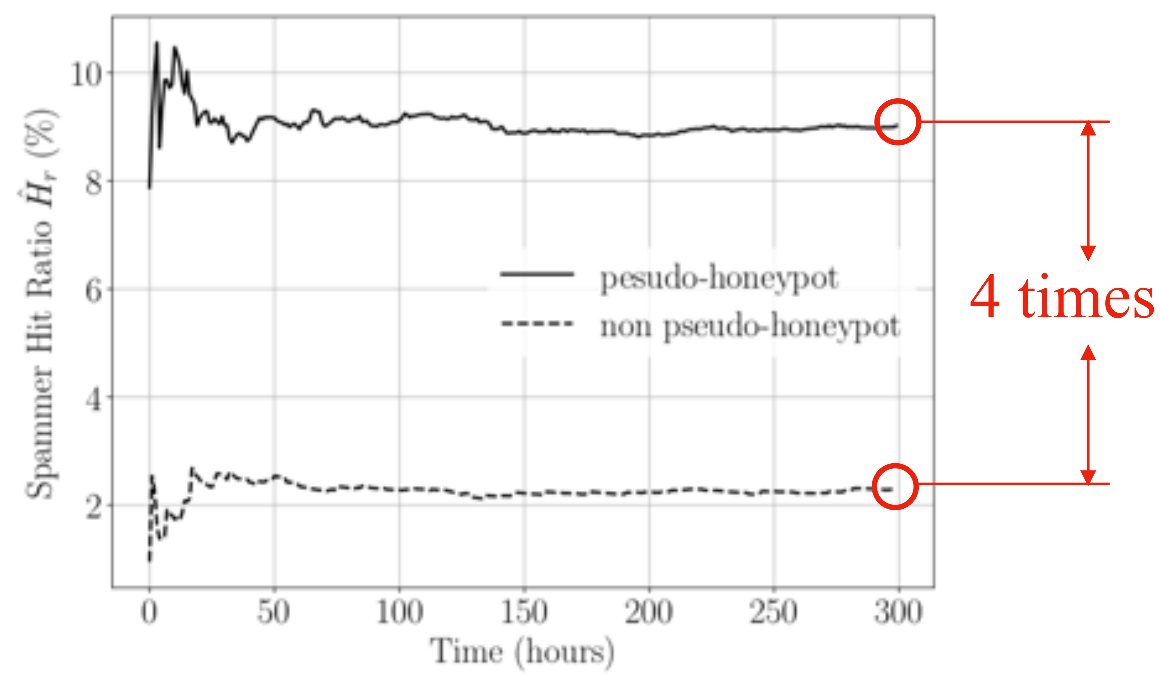


Figure. Top 5 hit ratio pseudo-honeypot vs. non pseudo-honeypot in 300 hours.

Effectiveness (# of spammer /hour)

Stringhini	Yang	Lee		Pseudo-honeypot
0.0067	0.087	0.12	8.6 times	1.03

Please check our articles:

https://ieeexplore.ieee.org/abstract/document/8809491

https://dl.acm.org/doi/abs/10.1145/3321705.3329836