TELLING EXPERTS FROM SPAMMERS: EXPERTISE RANKING IN FOLKSONOMIES

Michael G. Noll, Ching-Man Au Yeung, Nicholas Gibbins, Christoph Meinel, Nigel Shadbolt (SIGIR'09)

Presenter: Xiang Gao (Vincent)

Introduction

- Collaborative tagging – organizing and sharing
  - Documents relevant to a specified domain
  - Other users who are experts in a specified domain
- Existing systems only provide a list of resources or users
  - Large volume of data
  - Spammers
- SPEAR: our approach to assess the expertise
  - Be able to detect the different types of experts
  - More resistant to spammers

Outline

- Background
- SPEAR algorithm
- Experiments and Evaluation
- Conclusions and Discussions

Collaborative Tagging

- Allows users to assign tags to resources
  - User-generated classification scheme: *folksonomies*
- Definition of folksonomy
  - A folksonomy F is a tuple $F = (U, T, D, R)$
  - U: Users, T: Tags, D: Documents
  - $R = \{(u, t, d) | u \text{ gives } t \text{ to } d, (u, t, d) \in U \times T \times D\}$
  - $R_t = \{(u, d) | (u, t, d) \in R\}$
  - U_t, D_t

Related Work: HITS Algorithm

- *J. Kleinberg.* Authoritative sources in a hyperlinked envorinoment*. J. ACM, 1999*
- Precursor to PageRank
- Algorithm
  - Start with each node having a hub score and authority score of 1.
  - Run the Authority Update Rule
  - Run the Hub Update Rule
  - Normalize the
  - Repeat as necessary.

Expertise and document quality

- By the number of times he tags on some documents
  - Used by many existing systems
  - Quantity does not imply quality – spammers
- The ability to select most relevant information
- NOT enough alone to identify the experts

Discoverer vs. Follower

- An expert is able to give usefulness BEFORE others do
  - Expert is a discoverer, rather than a follower
  - The earlier a user has tagged a document, the more likely that he should be an expert
- The tagging time is an approximation of how sensitive he is to new information

Algorithm Design: Step 1

- Implement the idea of document quality
  - Mutual reinforcement
  - Similar to HITS

Algorithm 1

- Inputs
  - Number of users M
  - Number of documents N
  - Tagging $R_t = \{(u, t, d)\}$
  - Number of iterations k
- Output
  - A ranked list of users L

Algorithm 1 (cont.)

$\vec{E} \leftarrow (1,1,\ldots,1) \in \mathbb{Q}^M$

$\vec{Q} \leftarrow (1,1,\ldots,1) \in \mathbb{Q}^N$

$\vec{A} \leftarrow \{a_{i,j} = 1$ if user i tagged document $j, 0$ otherwise$\}$

For $i = 1$ to k do

$\vec{E} \leftarrow \vec{E} \times \vec{A}^T$

$\vec{Q} \leftarrow \vec{E} \times \vec{A}$

Normalize $\vec{E}$

Normalize $\vec{Q}$

End for

Similar to HITS

$L \leftarrow$ Sort users by expertise score in $\vec{\mathrm{E}}$

Return L

Algorithm Design: Step 2

- Implement the idea of discoverers and followers
- Include timing information in the tagging
  - $R = \{(u, t, d, c)\}$
- Prepare the adjacent matrix in a different way
  - ~~$\vec{A} \leftarrow \{a_{i,j} = 1$ if user i ...$\}$~~
  - $\vec{A} \leftarrow \{a_{i,j} =$ #followers if user i ...$\}$
    - #followers = $\left|\{u|(u_i, t, d_j, c_i) \in R_t \wedge c_i < c\}\right| + 1$

Credits

Algorithm 2

- Inputs
  - Number of users M
  - Number of documents N
  - Tagging $R_t = \{(u, t, d, c)\}$
  - Number of iterations k
- Output
  - A ranked list of users L

Algorithm 2 (cont.)

$\vec{E} \leftarrow (1,1,\ldots,1) \in \mathbb{Q}^M$

$\vec{Q} \leftarrow (1,1,\ldots,1) \in \mathbb{Q}^N$

$\vec{A} \leftarrow$ *Generated adjacent matrix*

For $i = 1$ to k do

$\vec{E} \leftarrow \vec{E} \times \vec{A}^T$

$\vec{Q} \leftarrow \vec{E} \times \vec{A}$

Normalize $\vec{E}$

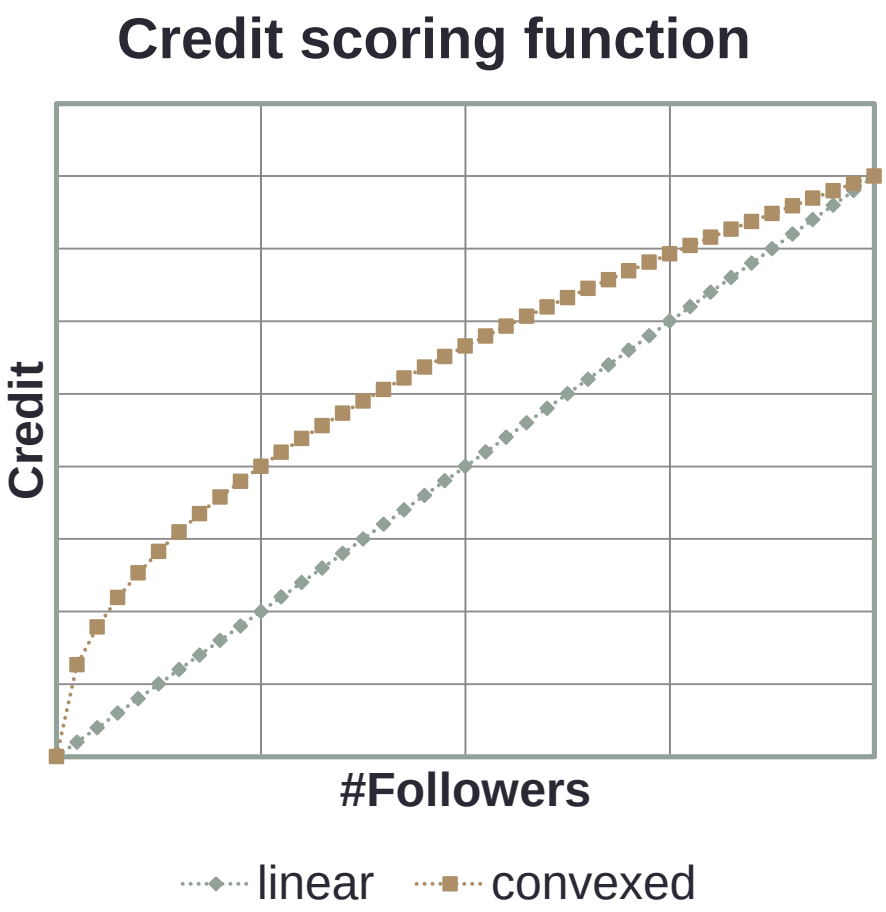
Normalize $\vec{Q}$

End for

$L \leftarrow$ Sort users by expertise score in $\vec{\mathrm{E}}$

Return L

Algorithm Design: Step 3



- The discoverer of a popular document will receive a high score
  - Even if he discovered the document by accident
  - and no other contribution
- The function C should have such a convexity
  - $C'(x) > 0, C''(x) \leq 0$
  - Here we use $C(x) = \sqrt{x}$
- ~~$\vec{A} \leftarrow \{a_{i,j} =$ #followers if ...$\}$~~
- $\vec{A} \leftarrow$ $\{a_{i,j} = C($#followers$)$ if ...$\}$

Final Algorithm: SPEAR

- Inputs
  - Number of users M
  - Number of documents N
  - Tagging $R_t = \{(u, t, d, c)\}$
  - Number of iterations k
- Output
  - A ranked list of users L

Final Algorithm: SPEAR

$\vec{E} \leftarrow (1,1,\ldots,1) \in \mathbb{Q}^M$

$\vec{Q} \leftarrow (1,1,\ldots,1) \in \mathbb{Q}^N$

$\vec{A} \leftarrow$ *Generated adjacent matrix, with the scoring function*

For $i = 1$ to k do

$\vec{E} \leftarrow \vec{E} \times \vec{A}^T$

$\vec{Q} \leftarrow \vec{E} \times \vec{A}$

Normalize $\vec{E}$

Normalize $\vec{Q}$

End for

$L \leftarrow$ Sort users by expertise score in $\vec{\mathrm{E}}$

Return L

Experiments

- Challenge: No ground truth
  - We never know whether someone is ACTUALLY an expert
  - Use simulated experts and spammers, and inject them into real world data
- Compare with FREQ and HITS

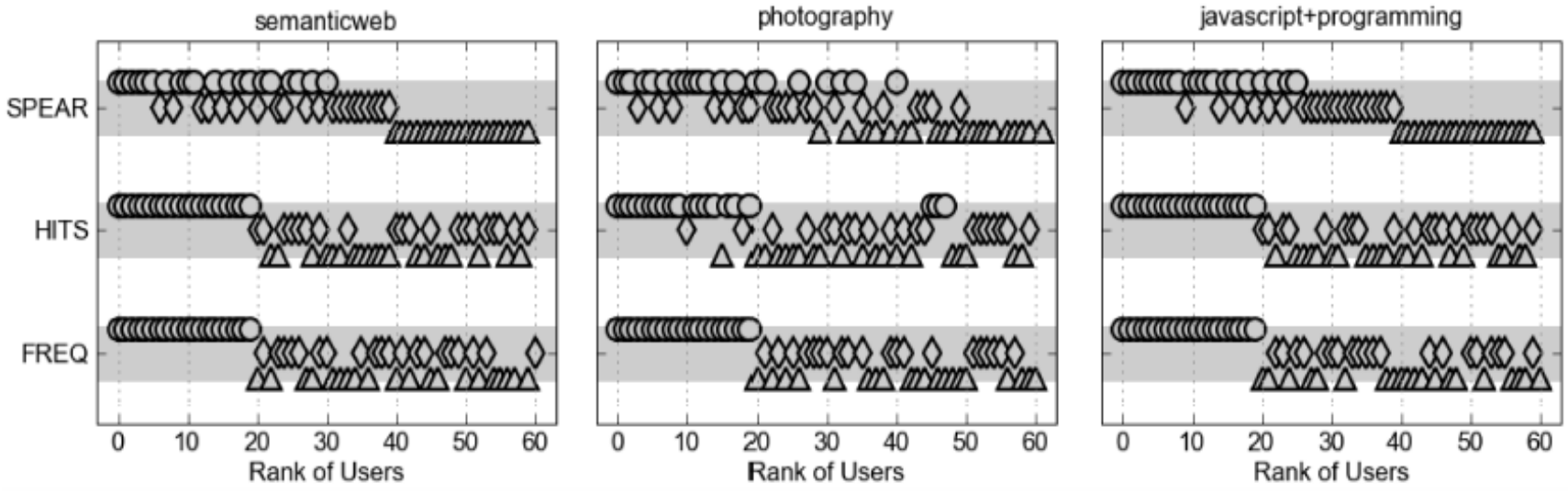
Types of simulated experts

- Veteran
  - Bookmarks significantly more documents than average user
- Newcomer
  - Only sometimes among the first to discover
- Geek
  - Significantly more bookmarks than a veteran
- Geek > Veteran > Newcomer

Types of simulated spammers

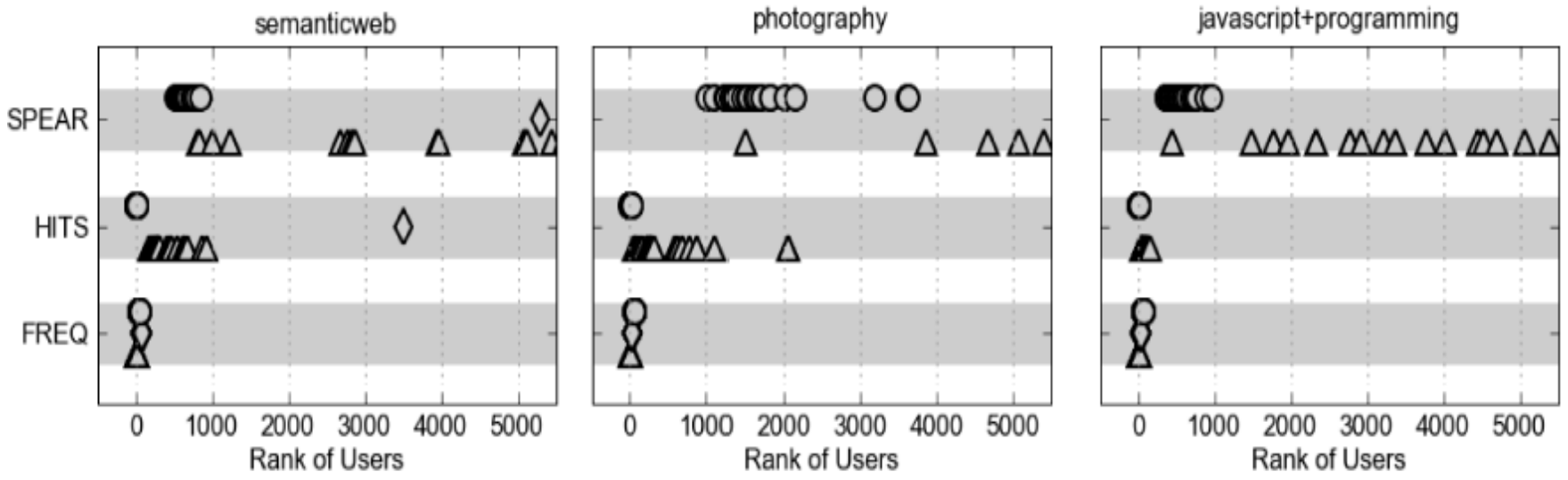
- Flooder
  - Tags a huge number of documents
  - Usually one of the last users in the timeline
- Promoter
  - Tagging his own documents to promote their popularity
  - Does not care about other documents
- Trojan
  - To mimic regular users
  - Sharing some traits with a so-called slow-poisoning attack.

Promoting Experts



Detect the differences between the three types of experts

Demoting Spammers



- Effectively demotes flooders and promoters,
- More resistant to Trojans than HITS and FREQ

Conclusions and Future Work

- SPEAR is
  - better at distinguishing various kinds of experts
  - More resistant to different kinds of spammers
- Future work:
  - Better credit score functions
  - Consider expertise in closely related tags
  - Activity of users

Limitations

- Validity of simulated input
  - Data mining bias – the input is generated according to an known conclusion
  - No evaluation using real data

THANKS