Social Networks: Routing Questions to the Right users in Online Communities

Xiaoqi Cao

Outline

- Motivation
- Three models for question routing
- Experiment and Evaluation
- Some deficiencies of this paper and ideas

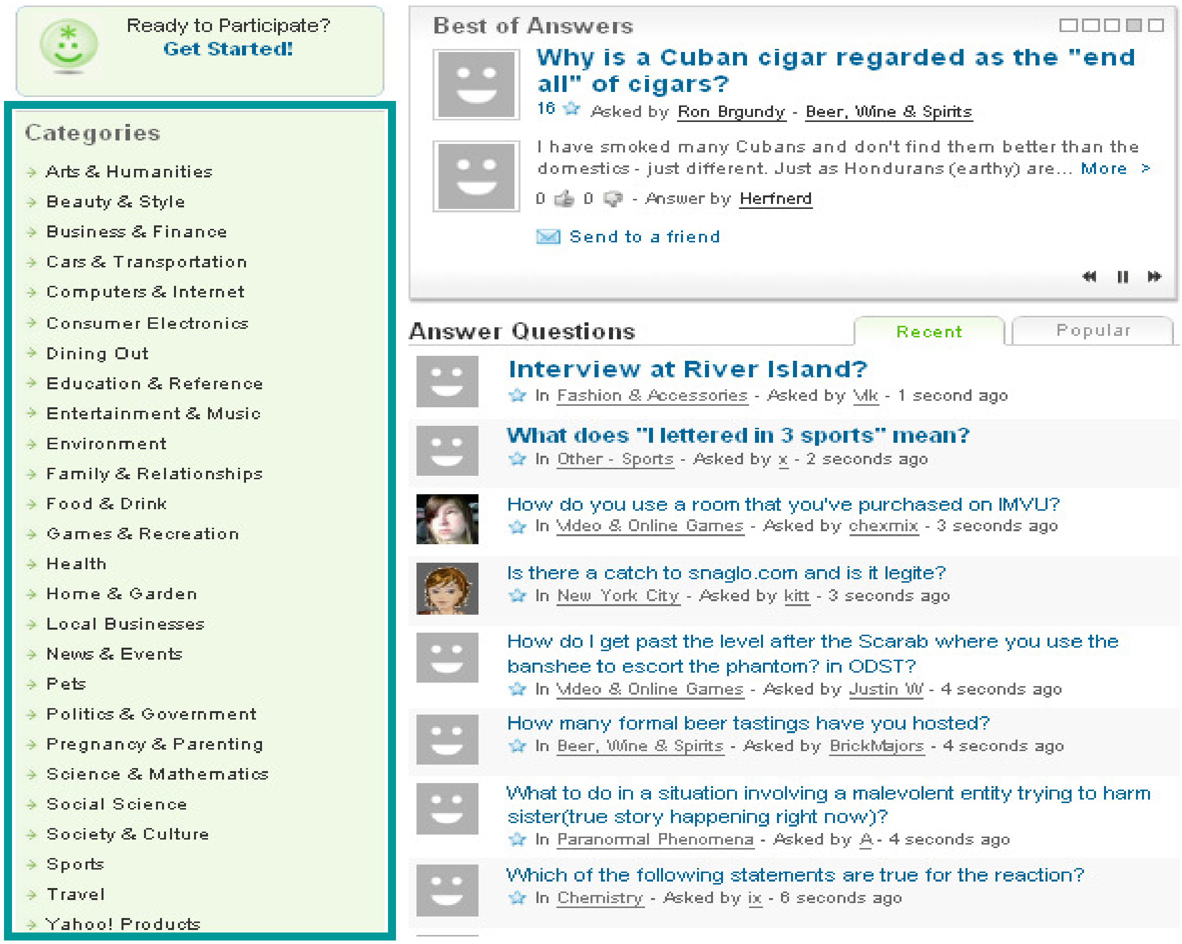
Motivation

- The question-answer forum (community)
  - User asks question
  - Other users reply the question
  - One question followed by 0 or more than 0 replies.
  - Thread

Web page snapshot from http://answers.yahoo.com/

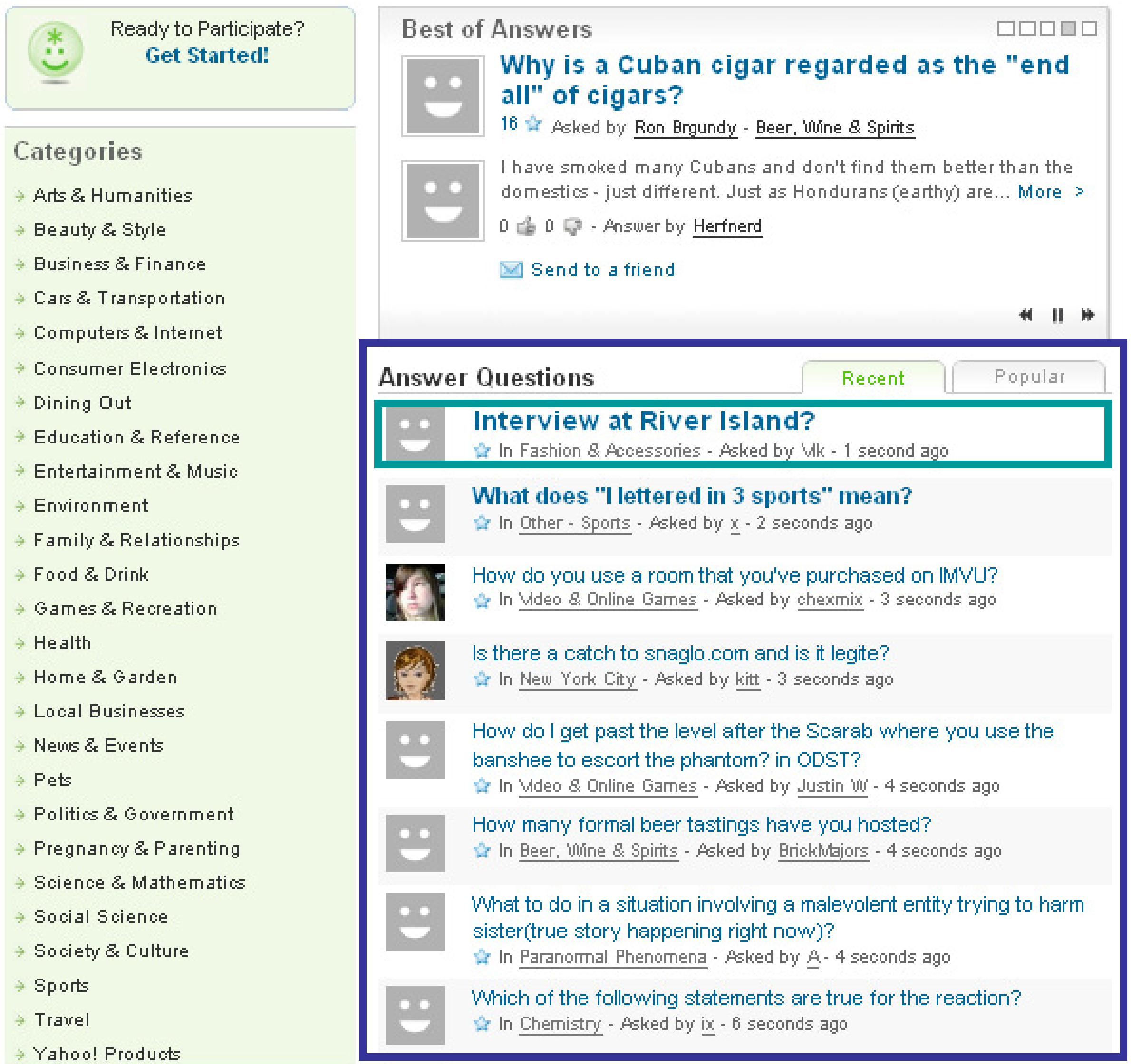
- Question
- Replies
- users
- Thread

Web page snapshot from http://answers.yahoo.com/



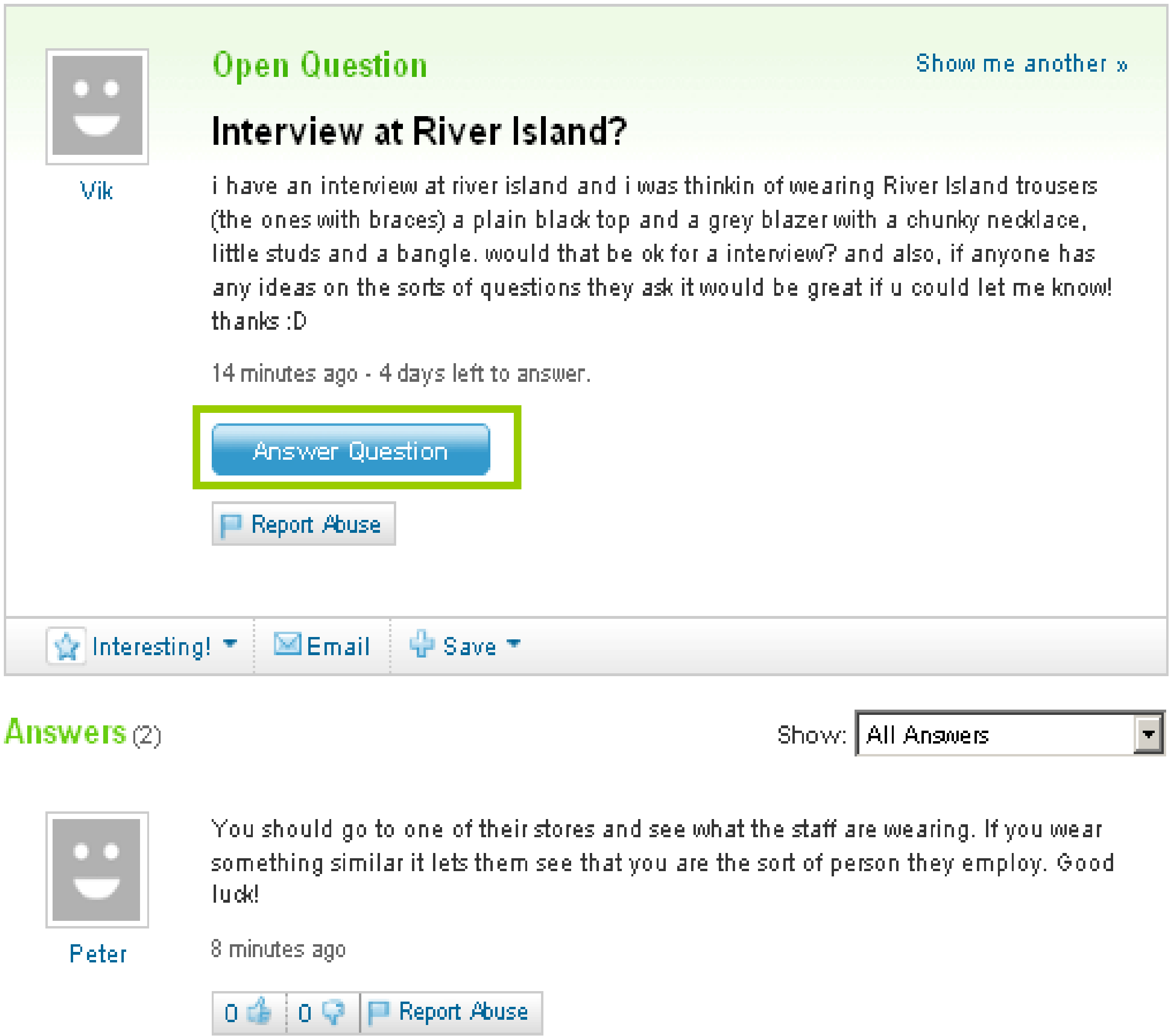
- Sub forum

Question List and picking up a question



- Question List
- Picking up a question

Picking up a question to answer



- Answer this question

Motivation

- People don't like to spend much time to find a question suitable to answer.
- Question are waiting passively.
- The question may be not replied or only followed by useless replies for long time.
- The question without reply may disappear from start page, people can not find it easily any more.
- Where is the right person who can answer my question?
- Is there some approach to deliver a question to the experts in this field?

Motivation

- Goal of this paper:
  - Using user activities data (question/answers) to compute the authorities of user for a certain new question, push this question (route this question) to top-k experts.

Overview of the approach

Overview of the approach

- How to find the top-k experts?
  - For a given question, compute probability of a user being an expert
  - Rank candidate users, get top-k of them, routing question to them

Overview of the approach

- Question and replies are composed of words. This approach is based on language model.
  - Split question and replies to words and count their quantities.
    - e.g Q = "what is the computer algorithm?"
      - tf('what') = 1, tf('is') = 1, tf('the') = 1,…
    - e.g R = "computer algorithm is the algorithm running in computer and towards to resolving some computer computable problems."
      - tf('computer') = 3, tf('algorithm') = 2, …

tf = term frequency

Overview of the approach

How to create profiles?
(**Index creation**)

Profiles
(history)

How to compute
experts rank for question?
(**Question processing**)

Question
(what happened now?)

Compute top-k
experts based
on a model

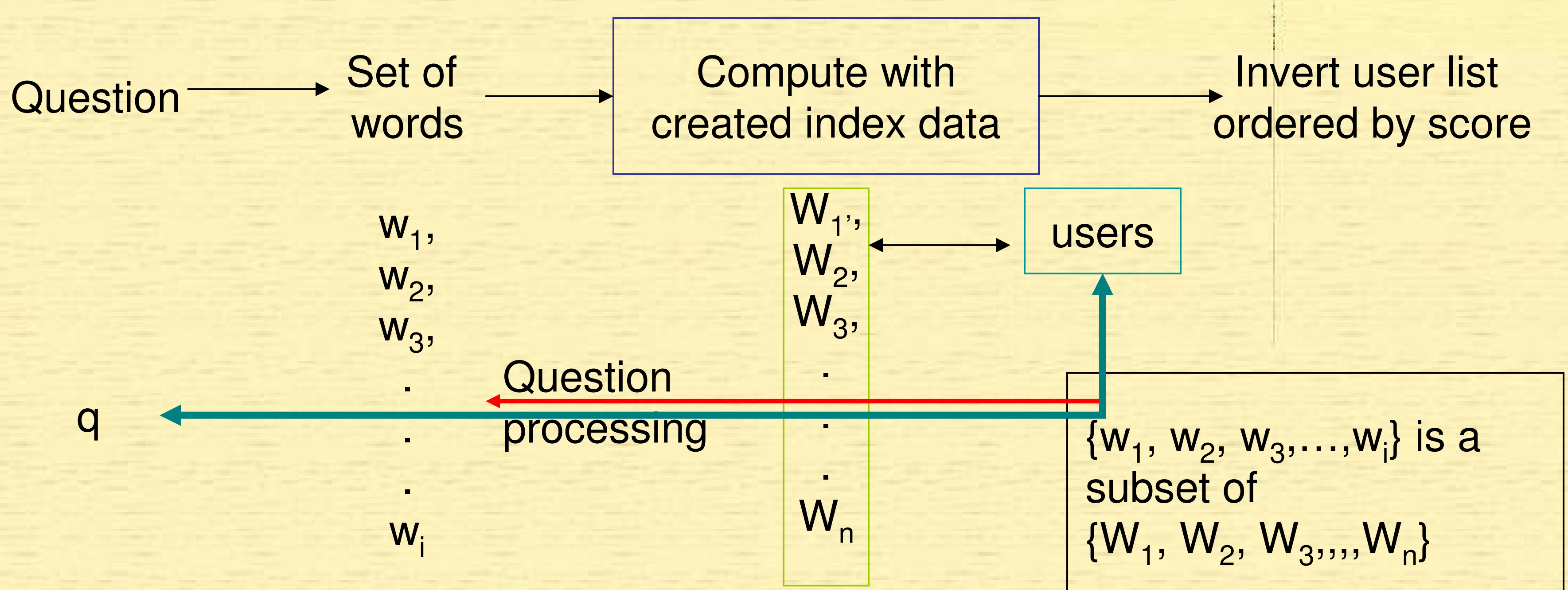
Ranked experts list
(solutions)

Overview of the approach

- Index creation
  - We use word to represent some knowledge field. If a certain word occurs many times in users questions or replies, it means that user may be familiar with that word (knowledge field) e.g 'music', 'travel'.
  - The approach of this paper uses history data to find which knowledge field each user is familiar with and establish index for user to indicate the relation between knowledge fields and users.

Overview of the approach

- Question processing
  – For given question, compute top-k ranked experts based on created index.



Overview of the approach

- The index creation approach used by this paper are all based on Language Model.

$$p(u \mid q) = \frac{p(q \mid u)p(u)}{p(q)}$$

- p(u|q): given a new question q, the probability of a user u being an expert on the question.
- p(q|u): given a user, it describes the expertise of user u on question q.
- p(u): prior probability of a candidate user.
- p(q): probability of a question generated by random user, it is a constant here. Same for all candidate users.

Three Models

- Profile-based model
- Thread-based model
- Cluster-based model

Profile-based model

- Profile that represents the her/his knowledge based on the answers she/he authored and question she/he asked before.

  $$p(w \mid u)$$

- Question contains some words

  $$p(q \mid u) = \prod_{w \in q} p(w \mid u)^{n(w,q)}$$

How user u is related (having knowledge) with the word w

How many times the word w occurs in question q

Profile-based model

td: thread
p(w|td) the prob of word w occurs in td

$$p(q \mid u) = \prod_{w \in q} p(w \mid u)^{n(w,q)}$$

$$p(w \mid u) = \sum_{td} p(w \mid td)\,con(td, u)$$

\# of words in q and reply of user u

$$\begin{cases} \text{single-doc}: p(w \mid td_u) = \dfrac{n(w,q) + n(w,r_u)}{|q \cup r_u|} \\ \text{question-reply}: p(w \mid td_u) = (1-\beta)p(w \mid q) + \beta p(w \mid r_u) \end{cases}$$

$$\beta \in [0,1]$$

$p(w|q)$ and $p(w|r_u)$ are max likelihood estimation of word w in question q and reply r_u, respectively

Profile-based model

- Example of single-doc and question-reply model
  - $td = Q + R1_u$.
  - Q: What is the computer algorithm?
  - $R1_u$: The computer algorithm means the computer understandable and implementable algorithm aiming at solving problem.
- Single-doc:
  p('algorithm'|td)
  =(n('algorithm' ,Q) + n('algorithm',$R1_u$))/|Q∪$R1_u$|
  =(1+2)/|5+14|≈0.16
- Question-reply (β=0.6):
  p('algorithm'|td)
  =(1 - β)p('algorithm', Q) + β p('algorithm', $R1_u$)
  =0.4* (1/5) + 0.6*(2/14) ≈ 0.17

Profile-based model

$$p(q \mid u) = \prod_{w \in q} p(w \mid u)^{n(w,q)}$$

$$p(w \mid u) = \sum_{td} p(w \mid td)\, con(td, u)$$

Contribution of user u to thread td.
It describes the participation
of user u in thread td

$$con(td, u) = \frac{\prod_{w \in q} p(w \mid \theta_{r_u})}{\sum_{td'} \prod_{w \in q'} p(w \mid \theta_{r'_u})}$$

$$p(w \mid \theta_{r_u}) = (1-\lambda) p(w \mid r_u) + \lambda p(w)$$

Smoothing

$$\lambda \in [0,1]$$

C is collection of
all the words in forum

$$p(w) = \frac{n(w, C)}{|C|}$$

$$p(w \mid r_u) = \frac{n(w, r_u)}{|r_u|}$$

Max likelihood estimation
for word w in reply r_u

Profile-based model

- Smoothing

$$p(w \mid \theta_{r_u}) = (1-\lambda)p(w \mid r_u) + \lambda p(w) \quad \lambda \in [0,1]$$

- λ is coefficient to control the influence of the background model
  - e.g λ = 0.9, word w = ‘of’ and p(‘of’) = 0.01
  - p(‘of’) = n(‘of’, C)/|C|

Profile-based model

$$p(q \mid u) = \prod_{w \in q} \underline{p(w \mid u)}^{n(w,q)}$$

$$\boxed{p(w \mid \theta_u) = (1-\lambda)p(w \mid u) + \lambda p(w)}$$

$$\lambda \in [0,1]$$

$$p(w \mid \theta_u)$$

use the smoothed $p(w|\theta_u)$ to compute $p(q|u)$

$$p(q \mid u) = \prod_{w \in q} p(w \mid \theta_u)^{n(w,q)}$$

Profile-based model

- Index creation algorithm
  - For each word w compute inverted list of user score

$$p(w \mid u) = \sum_{td} p(w \mid td)\, con(td, u)$$

$$\xrightarrow{inverted\_ordered\_by\_value\_of\_p(w|u)}$$

word: w_i	→	u_m, $p(w_i, \theta u_m)$	u_n, $p(w_i, \theta u_n)$	u_k, $p(w_i, \theta u_k)$	…

Profile-based model

- Index creation algorithm

```
//Generation stage
 for each user u do
   for each word w, initialize p(w|u) to 0.
   find all threads {td} replied by user u and compute con(td)
   for each td in {td} do
       for each word w in td do
         compute p(w|td);
         p(w|u) += p(w|td)con(td,u);
       end for each word
   end for each thread
   Smooth all p(w|u) to p(w|θu)
   for each word w, store the triple of (w, u, p(w| θu))
 end for each user
```

$$p(w \mid u) = \sum_{td} p(w \mid td)\,con(td, u)$$

$$p(w \mid \theta_u) = (1-\lambda)p(w \mid u) + \lambda p(w) \quad \lambda \in [0,1]$$

```
//Sorting stage
 for each word w do
   find the list of (u, p(w| θu)) and sort it by p(w| θu);
 end for each word
```

w_i	→	u_m, $p(w_i, u_m)$	u_n, $p(w_i, u_n)$	u_k, $p(w_i, u_k)$	…

Profile-based model

- Question Processing
  - Based on sorted inverted user score list for each word w_i, apply threshold algorithm to compute top-k ranked results (users) for new question q.

Profile-based model

- Question Processing (Threshold Algorithm)

Split new question q into l words $\{w_1, w_2, \dots, w_l\}$
Let $Y = \{(u, score(u))\}$ keeps track of current top-k results
in descending order of score (u) (it is empty now)
Round-Robin access to each of the sorted lists L_i,
as entry $(u, p(w_i|\theta_u))$ is seen, find all other $p(w_{-i}|u)$ in other lists
(w_{-i} represents all other words in the question except for word w_i.)
Compute $score(u) = \prod_{w \in q} p(w|\theta_u)^{n(w,q)}$
If Y is not full or score(u) is larger than the minimum score in Y
then store the pair (u, score(u)) in Y
For each list L_i, let $t = \prod_i p(w_i|\theta_{u_j^*})^{n(w_i,q)}$
where $(u^*_l, p(w_i|\theta_{u^*}))$ is the last entry seen under round-robin access for L_i.
If scores of all the k users in Y are no less than t then stop.
Output the top-k results in set Y.

Profile-based model

- Example of threshold algorithm

Q: “computer algorithm?” **w1=computer, w2=algorithm**

w1	u78 0.9	u23 0.6	u10 0.5	u64 0.3	u43 0.2	…
w2	u64 0.9	u43 0.8	u10 0.3	u23 0.2	u78 0.1	…

$$score(u) = \prod_{w \in q} p(w \mid \theta_u)^{n(w,q)}$$

Y = {u, score(u)}

Rank	user	Score
1	u64	0.27
2	u43	0.16

$$t = \prod_i p(w_i \mid \theta_{u_j^*})^{n(w_i,q)}$$

Threshold2=0.16 (u43)

Thread-based model

- Each thread is a latent topic
- Establish the relation (distance) between each thread td and the given question q.
- For each thread td, user gives contribution to td by participating in td (asking or answering the question of td).
- We measure the relation (distance) between user and given question by consider both factors above. If a thread td has close relation with a given question q and some user u has contributed to that thread td, that means u has certain relationship with q.

Thread-based model

- This model builds a language model per thread and then constructs the probability for each user as a mix model of these per-thread language models, weighted by the contribution of user to the thread.
- We also create index first and then compute top-k user based on this index

$$p(q \mid u) = \sum_{td} p(q \mid td)\,con(td, u)$$

$$\xleftarrow{\text{smoothing}} p(w \mid \theta_{td}) = (1-\lambda)p(w \mid td) + \lambda p(w) \quad \lambda \in [0,1]$$

$$p(q \mid u) = \sum_{td} p(q \mid \underline{\theta_{td}})\,con(td, u)$$

Thread-based model

- Index creation (algorithm like before)

Thread list

$\xrightarrow{\text{inverted\_ordered\_by\_value\_of\_p}(w|\theta_{td})}$

word: w_i	→	td_m, $p(w_i, \theta_{tdm})$	td_n, $p(w_i, \theta_{tdn})$	td_k, $p(w_i, \theta_{tdk})$	…

Thread-user Contribution list

$\xrightarrow{\text{inverted\_ordered\_by\_value\_of\_con}(td|u)}$

thread: td_i	→	u_m, $con(td_i, u_m)$	u_n, $con(td_i, u_n)$	u_k, $con(td_i, u_k)$	…

Thread-based model

- Question processing (TA Twice)

First, for given question q, find threads {td} with highest relevant score

word: w_i	→	td_m, $p(w_i, \theta_{tdm})$	td_n, $p(w_i, \theta_{tdn})$	td_k, $p(w_i, \theta_{tdk})$	…

$$\mathrm{score}(td) = \prod_{w \in q} p(w \mid \theta_{td})^{n(w,q)}$$

Y = {(td, score(td))}

Second, based on {td}, compute top-k ranked users for q.

thread: td_i	→	u_m, $con(td_i, u_m)$	u_n, $con(td_i, u_n)$	u_k, $con(td_i, u_k)$	…

$$\mathrm{score}(u) = \sum_{td \in Y} \mathrm{score}(td)\,\mathrm{con}(td, u)$$

Cluster-based model

- Cluster (sub-forum): amount of threads which have similar topics.
  - Travels (Forum) → In Europe (Sub-forum)
  - Music (Forum) → Classic music (Sub-forum)
- Group threads with similar contents (in identical sub-forum)
- Compute ranking score for each user by aggregating all clusters

Cluster-based model

- Cluster consists of many threads {td}
  - Combine all the questions in {td} to Q
  - Combine all the replies in {td} to R
  - We have one big thread **Td = 1 Q + 1 R**.

$$p(w|cluster) = p(w|Td)$$

$$\mathrm{con}(\mathrm{cluster}, u) = \sum_{td \in \{td\}} \mathrm{con}(td, u)$$

Cluster-based model

- Jobs we do here:
  - Generating cluster (use sub-forum)
  - Index creation (create profiles: inverted ordered lists)
  - Question processing (TA algorithm)

$$p(q \mid u) = \sum_{Cluster} \left[\prod_{w \in q} p(w \mid Cluster)^{n(w,q)} con(Cluster, u)\right]$$

$$\xleftarrow{\text{smoothing}} p(w \mid \theta_{cluster}) = (1-\lambda)p(w \mid cluster) + \lambda p(w)$$

$$\lambda \in [0,1]$$

$$p(q \mid u) = \sum_{Cluster} \left[\prod_{w \in q} p(w \mid \underline{\theta_{Cluster}})^{n(w,q)} con(Cluster, u)\right]$$

Cluster-based model

- Index creation

Cluster list

$\xrightarrow{\text{inverted\_ordered\_by\_value\_of\_p}(w|\theta_{cluster})}$

word: w_i	→	clu_m, $p(w_i, \theta_{clum})$	clu_n, $p(w_i, \theta_{clun})$	clu_k, $p(w_i, \theta_{cluk})$	…

Cluster-user Contribution list

$\xrightarrow{\text{inverted\_ordered\_by\_value\_of\_con}(cluster|u)}$

cluster: clu_i	→	u_m, $con(clu_i, u_m)$	u_n, $con(clu_i, u_n)$	u_k, $con(clu_i, u_k)$	…

Cluster-based model

- Question processing

First, access cluster lists and compute score for each cluster

$$\mathrm{score(cluster)} = \prod_{w \in q} p(w \mid \theta_{\mathrm{cluster}})^{n(w,q)}$$

Y = {(clu, score(clu))}

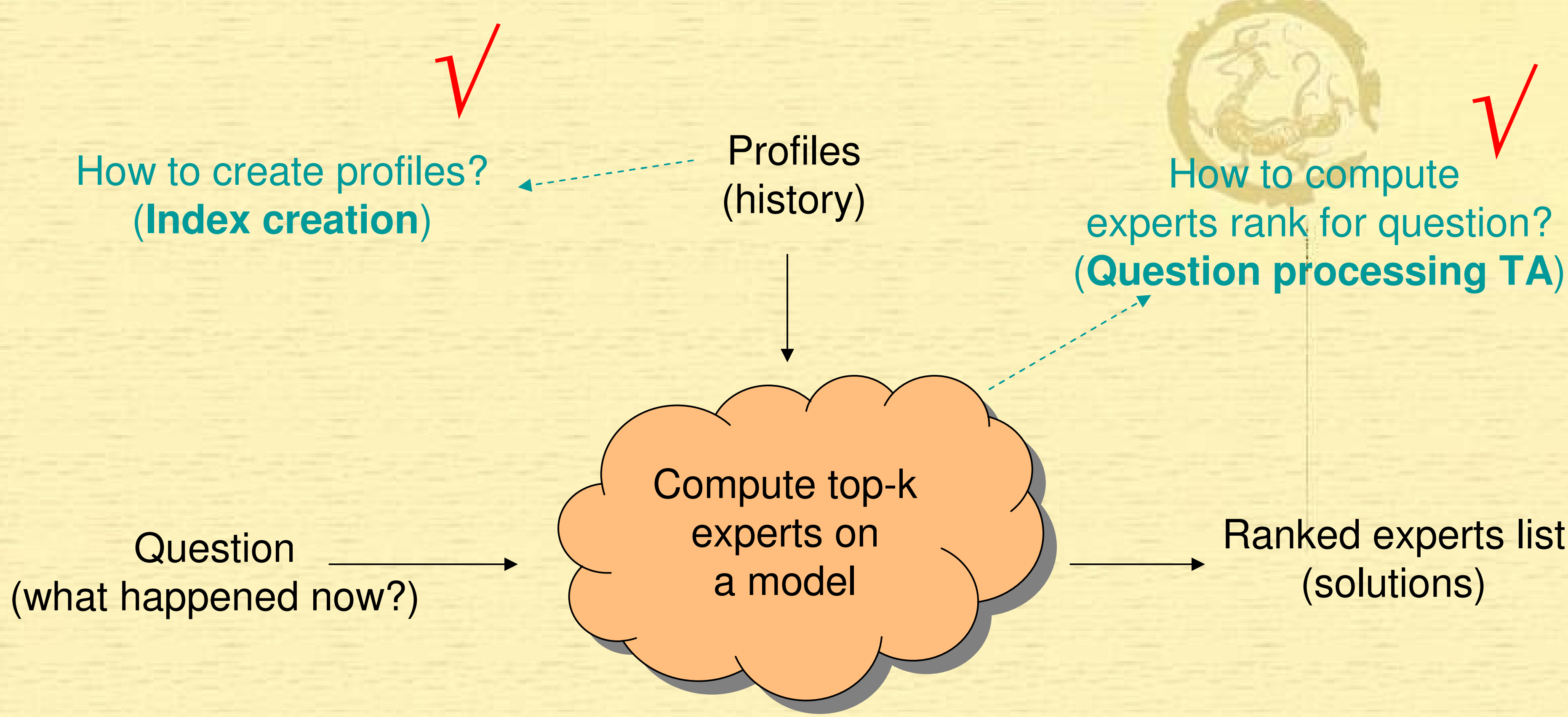
word: w_i	→	clu_m, $p(w_i, \theta_{clum})$	clu_n, $p(w_i, \theta_{clun})$	clu_k, $p(w_i, \theta_{cluk})$	…

Second, based on score(cluster), access cluster-user contribution list to compute top-k ranked users. (Threshold algorithm)

$$\mathrm{score(u)} = \sum_{clu \in Y} \mathrm{score(clu)con(clu,u)}$$

cluster: clu_i	→	u_m, $con(clu_i, u_m)$	u_n, $con(clu_i, u_n)$	u_k, $con(clu_i, u_k)$	…

The jobs we did



Experiments and Evaluation

Experiments and Evaluation

- Evaluate the result of the approach in two aspects
  - Effectiveness
  - Efficiency

Experiment and Evaluation

- Test effectiveness
- Data: from http://www.tripadvisor.com/

	#threads	#posts	#users	#words	#clusters
BaseSet	121,704	971,905	40,248	324,055	17

Extra 10 questions from tripadvisor as new questions,
Sampled102 users from baseSet also answered the 10 questions above

- Mapping: $Q \times U \rightarrow \{0,1\}$ $0<i<11,\ 0<j<103 \quad i, j \in N$
  - Q: the set of new questions; U the set of users who answer new question.
  - $q_i \times u_j = 1$: If user u give the correct answer of question q
  - $q_i \times u_j = 0$: otherwise
- Manually, annotating which user u give helpful (correct) answer to question q in 10 (we have 10 * 102 0 or 1).
- Use models of this paper to estimate experts of user in U to new questions in Q and make comparison with annotated ones.

Experiment and Evaluation

- For different language model (single-doc / question-reply), experiment got results like below (**question-reply got better result**):

Thread LM	MAP	MRR	R-precision	P@5	P@10
Single-doc	0.567	0.761	0.391	0.54	0.54
Question-reply (β=0.5)	**0.584**	**0.8**	**0.391**	**0.58**	**0.54**

- MAP: mean of the average of precisions over a set of query question. It represents the correctness of the results which are generated by question routing.
- MRR: mean of the reciprocal ranks of first correct answers over a set of query questions.
- Precision @N: percentage of the top-N candidates answers retrieved that are correct.
- R-precision: having a set of known relevant answers *Rel*, from which we calculate the precision of the top *Rel* answers returned.

Experiment and Evaluation

- For different *rel*s, based on thread-based model the results like below:

rel	MAP	R-precision	P@5	Top-10 search (second)
200	0.550	0.201	0.56	4.05
400	0.569	0.265	0.58	4.05
600	0.576	0.346	0.58	4.66
800	0.582	0.391	0.58	4.82
All	0.584	0.391	0.58	11.87

- rel: the # of relevant threads

Experiment and Evaluation

- Comparing with “Replies Count” and “Global Rank ”

Method	MAP	MRR	R-precision	P@5	P@10
Replies Count	0.130	0.131	0.121	0.08	0.1
Global Rank	0.134	0.152	0.118	0.08	0.1
Profile	0.563	**0.87**	0.369	0.56	0.52
Thread	**0.582**	0.8	0.391	**0.58**	**0.54**
Cluster	0.532	0.736	**0.452**	0.46	0.49

  - λ=0.7, β=0.5, *rel*=800
- Reply Count: uses # of threads replied by user as user’s score
- Global Rank: estimates the authority score of a user by user’s PageRank value.
- The approach provided by this paper turns out better results.

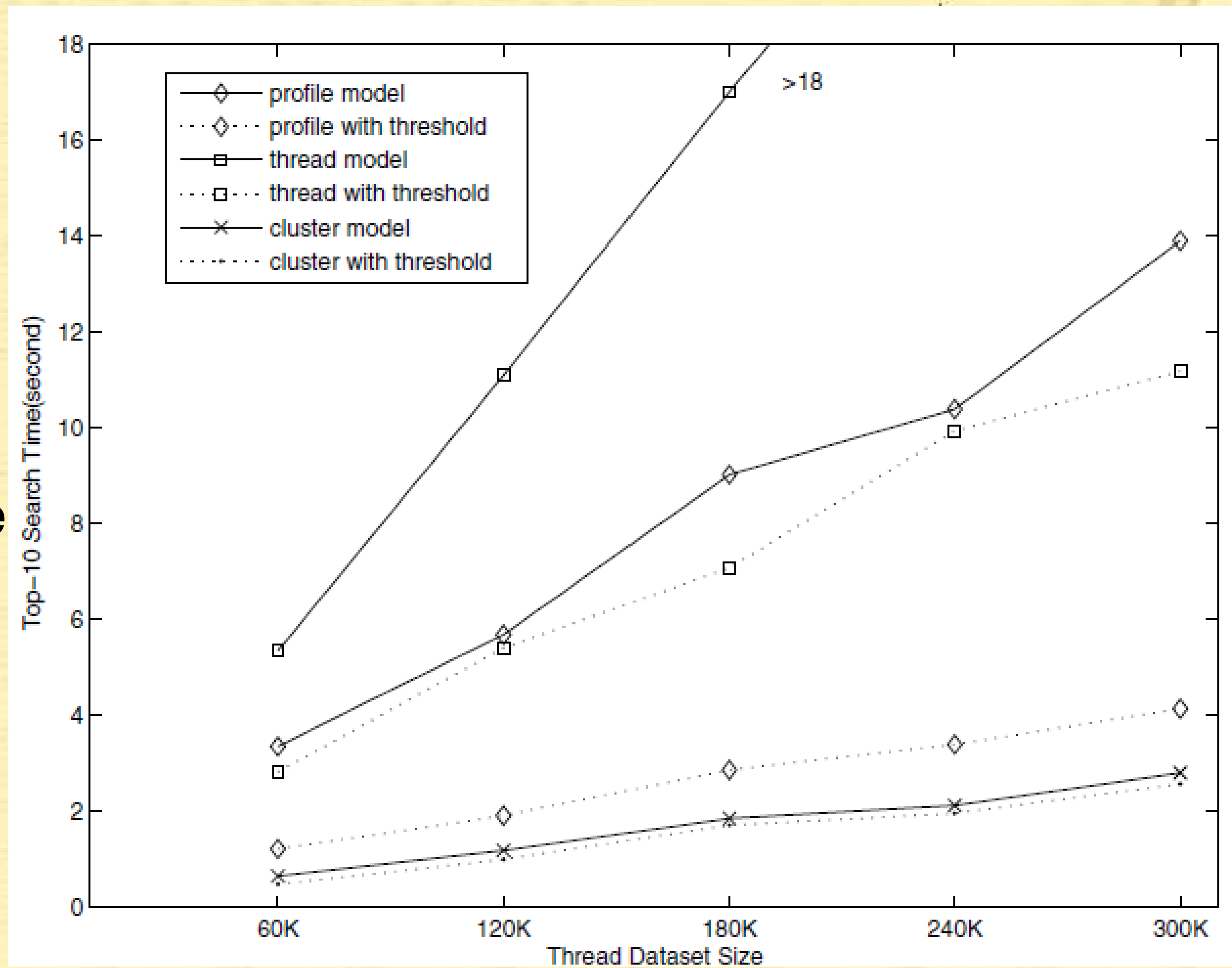
Experiment and Evaluation

- Test efficiency
- On index creation:

Method	List Generation Time	List Sorting Time	Index Size
Profile	153min	145min	490MB
Thread	148min	435min	502+40.2MB
Cluster	142min	0.4min	48.8+0.9MB

Experiment and Evaluation

- Test efficiency
- On Scalability
  - Profile get worse when data set is bigger.
  - Cluster is not such sensitive to the size of data set and always plays best.



Issues of this paper

Issues of this paper (1)

- Based on frequency of words
  - Helpless answer but high contribution for ranking since same words here
    - Question: “What is computer algorithm?”
    - Reply: “Computer algorithm.”
  - Contrarily, helpful answer but low contribution for ranking since it has no relevant word with question
    - http://computer.howstuffworks.com/question717.htm
- If a user with high score has not logged on for years or the user logged on actively but never answered the pushed question?
  - Don’t push question to that user

Issues of this paper (1)

- Feedback from questioner and activation of user

$$p(u \mid q) = \sum_{td} p(q \mid \theta_{td}) con(td, u)$$

$$con(td, u) = \frac{\prod_{w \in q} p(w \mid \theta_{ru})}{\sum_{td'} \prod_{w \in q'} p(w \mid \theta_{r'u})}$$

Questioner gives the feedback after question answered
Donothing =0; satisfied=1

Measures: How often user u logon to forum?
Lower value for longer interval

$$con(td, u) = \alpha \frac{\prod_{w \in q} p(w \mid \theta_{ru})}{\sum_{td'} \prod_{w \in q'} p(w \mid \theta_{r'u})} + \beta \boxed{feedback(td, asker_{td})} + \gamma \boxed{active(u)}$$

$$\alpha, \beta, \gamma \in [0,1] \wedge \alpha + \beta + \gamma = 1$$

Issues of this paper (2)

- What if the forum is new?
  - Little threads data
  - Little authorities information about users
- Computing ranking less precisely.
- Profile may be computed by surveys when user registering.
  - “Which field do you like most?”
  - Can be done by recommending user to fill some html form.

Issues of this paper (3)

- What if the forum may have huge amount of users and question-replies data and they tend to change at any time?
  - Horrible, if so frequently computing new profile
  - Can we compute new ranking by modifying the old one?
- The thing we do is adjusting the order of user in index in models when new user answered a question or even batch updating.

Thank you for your attention!