Finding Local Experts for Dynamic Recommendations Using Lazy Random Walk

Diyah Puspitaningrum, Julio Fernando, Edo Afriando, Ferzha PU, Rina Rahmadini, and Y. Pinata

Department of Computer Science

University of Bengkulu

Bengkulu, Indonesia

E-mail: diyahpuspitaningrum@gmail.com

https://www.diyahpuspitaningrum.net

The 7th International Conference on Cyber and IT Service Management
(*CITSM 2019*)

Finding Local Experts for Dynamic Recommendations Using Lazy Random Walk

- Statistics based privacy-aware recommender systems make suggestions more powerful by extracting knowledge from the log of social contacts interactions, but unfortunately, they are static — moreover, advice from local experts effective in finding specific business categories in a particular area.
- We propose a dynamic recommender algorithm based on a lazy random walk that recommends top-rank shopping places to potentially interested visitors.
- We consider local authority and topical authority.
- The algorithm tested on FourSquare shopping data sets of 5 cities in Indonesia with k-steps={5,7,9} (lazy) random walks and compared the results with other state-of-the-art ranking techniques.
- The results show that it can reach high score precisions (0.5, 0.37, and 0.26 respectively on p@1, p@3, and p@5 for k=5).
- The algorithm also shows scalability concerning execution time.
- The advantage of dynamicity is the database used to power the recommender system; no need to be very frequently updated to produce a good recommendation.

Finding Local Experts for Dynamic Recommendations Using Lazy Random Walk

- There have been studies that show that people are likely to prefer learning from local experts who know the neighborhood well and have first-hand experience [1]. In the case of shopping place recommender systems, to predict a user u's preference for place p, the recommender model should be built from an appropriate local expert given a database of local shopping places with positive reviews. Using statistics based model has a drawback that it leads to choosing the same most popular spot by visitors reviews. In the case of our proposed lazy random walk recommender model, a recommender system recommend a place because the user listens either to recommendations given by his social contacts about a specific region or to suggestions given by local experts from the global network, with both of them must meet the local authority and topical authority criteria.
- Furthermore, finding local experts is not a trivial task. Since FourSquare data only comes from visitors' reviews of some shopping places, the data is sparse for both users and shopping places. Typical shopping place on FourSquare has few visitors' reviews.
- This work aims to come up with a system that can find dynamically local experts, to produce dynamic recommendations, given a query about shopping in certain cities in which suggestions from a local expert from a friend's graph is more likely than the one from global networks.

Problem Definition

- In privacy-aware recommender systems we have a set of users U = {u1, ..., uN} assuming that N is the total number of users in the shopping dataset and a set of shopping places P = {p1, ..., pM} and M is the total number of shopping places by local expert's query category c(q). Each user ui can be associated with a location li and a set of categories Ci. The objective is to find the collection of shopping places from users U(q) such that for each uj in U(q), c(q) is in Cj, and l(q) is local to lj, dynamically using Lazy Random Walks. By privacy-aware, we prefer recommendations from social contacts rather than from global users.

Problem Definition (2)

- Several main definitions as follows:
- 1) Local authority: A user is said a candidate of a local expert if the user's location, and the query location, are in the same city.
- 2) Topic authority: A user is said a candidate of a local expert if the user has considerable knowledge about the query category to be recognized as an expert or not. Since the dataset is sparse, a user is a local expert candidate if he has reviews about category c(q).
- 3) Places: Recommendations are generated from shopping places with the majority of positive reviews and suggested by a local expert that meets both the local authority and the topical authority criteria. In this work, we do sentiment analysis first to categorize shopping places dataset into two bins: positive and negative bins.

Related Works

- In this section, we review related work on recommender systems related to the use of local experts and random walk. Tanvi Jindal (2015) [2] solves the problem of finding local experts from the Yelp dataset. He proves that the random forest algorithm works best to classify expert and non-expert, so the recommender system gives the best prediction accuracy. Once we have the set of local experts for a category in a location, we can use it to generate a summary for the businesses in that category. Since they are knowledgeable and influential, using just their reviews would give a higher quality summary about a business.
- Torres et al. (2012) [3] use random walks as a query suggestion method to help children find keywords that are more likely to be relevant for them. Abbasssi and Mirrokni (2007) [16] apply the local partitioning algorithms based on refined random walks to approximate the personalized PageRank vector for the use of a recommender system for weblogs based on the link structure among them. In (Bogers, 2010) [14], random walks over the contextual graph models the browsing process of a user on a movie database website. Gori and Pucci (2006) [12] use a random-walk based scoring algorithm to rank products according to expected user preferences. In (Jamali and Ester, 2009), [15] a random walk algorithm is proposed to recommend items in a trusted network. The algorithm recommends items based on ratings expressed by trusted friends, using random walk and probabilistic item selection.

Algorithms

Our proposed recommendation system as follows:

1. Query identification: query q (or category such as: "shopping mall", "department store", "supermarket", "bookstore", "market"), user id, city, and algorithm.

2. Filter the visitor review data only to a graph that meets the query criteria. Select only reviews that originated from user id that located in the city stated in point 1.

3. If a user is a person without social contacts, then use a global graph search local expert. Otherwise, use a graph derived from his social contacts to do the local expert finding.

4. As we use network properties and random walk variants to test our system, then if method="PageRank" (or other network centrality measures, e.g., "betweenness," "closeness," and "degree"), for the highest local expert score, find his recommendation of places that meet the query. Else: If method="Lazy Random Walk" or "Random Walk," for source node of all users in the graph (see point 3) with walking-steps = {5, 7, 9} use the last step of the walk as a local expert candidate and then find his recommendation of places that meet q.

5. Sort recommendations by the number of positive reviews in its descending order.

Remark 1. The "Global" Graph Search Procedure

1. Build a global graph(s) of reviewers (users who have comments on a shopping place that is meet the user query q) by finding each reviewer's contact until two intermediate nodes. A reviewer in this graph can be connected or isolated, so it is possible to have more than one global graph.

2. Since a local expert should influence his local network (in his city), find the highest degree graph as the origin of candidates of local experts.

3. Do lazy random walk or PageRank on the graph with the number of walking steps = {5,7,9}.

4. The last visited node is the local expert.

Remark 2. The "Privacy-Aware" Graph Search Procedure

1. Build a global graph until two intermediate nodes ahead of a user who request the query.

2. Do lazy random walk on the graph.

3. The last visited node is the local expert.

Algorithms (2)

Our proposed recommendation system as follows:

1. Query identification: query q (or category such as: "shopping mall", "department store", "supermarket", "bookstore", "market"), user id, city, and algorithm.

2. Filter the visitor review data only to a graph that meets the query criteria. Select only reviews that originated from user id that located in the city stated in point 1.

3. If a user is a person without social contacts, then use a global graph search local expert. Otherwise, use a graph derived from his social contacts to do the local expert finding.

4. As we use network properties and random walk variants to test our system, then if method="PageRank" (or other network centrality measures, e.g., "betweenness," "closeness," and "degree"), for the highest local expert score, find his recommendation of places that meet the query. Else: If method="Lazy Random Walk" or "Random Walk," for source node of all users in the graph (see point 3) with walking-steps = {5, 7, 9} use the last step of the walk as a local expert candidate and then find his recommendation of places that meet q.

5. Sort recommendations by the number of positive reviews in its descending order.

Remark 1. The "Global" Graph Search Procedure

1. Build a global graph(s) of reviewers (users who have comments on a shopping place that is meet the user query q) by finding each reviewer's contact until two intermediate nodes. A reviewer in this graph can be connected or isolated, so it is possible to have more than one global graph.

2. Since a local expert should influence his local network (in his city), find the highest degree graph as the origin of candidates of local experts.

3. Do lazy random walk or PageRank on the graph with the number of walking steps = {5,7,9}.

4. The last visited node is the local expert.

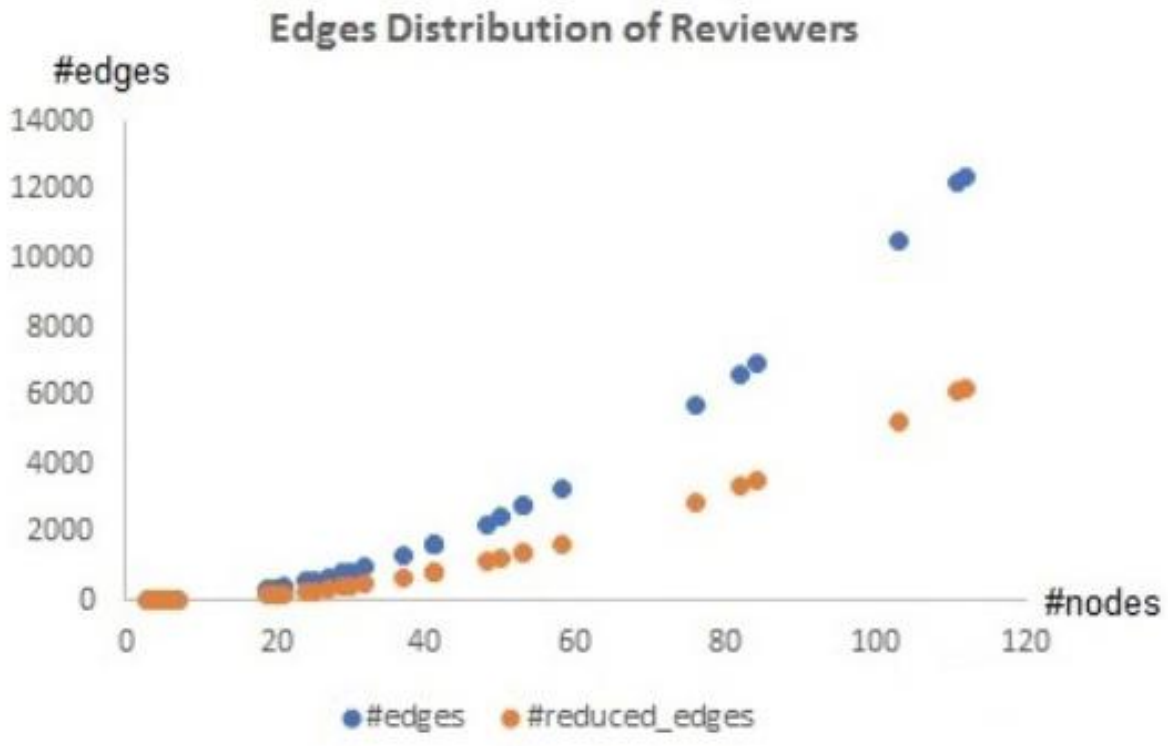
Remark 2. The "Privacy-Aware" Graph Search Procedure

1. Build a global graph until two intermediate nodes ahead of a user who request the query.

2. Do lazy random walk on the graph.

3. The last visited node is the local expert.

Statistics of Dataset



2(a). Edge Distribution of User Review Counts

2(b). Places Distribution of Local Expert Candidates

Parameter Set Up

- We have 220 queries qi consists of combinations of: query+city+mode+k-walking steps. query = {"mall", "inexpensive market", "discount books", "dress shop", "comfortable shopping", "complete and inexpensive shopping"}. city loc= {Jakarta, Bandung, Yogyakarta, Palembang, Bengkulu}. mode = {"global", "pa"}. k-walking steps = {5,7,9,11,13,15,17,19,21,23} walking steps. The term “global” refers to global graph search procedure; the “pa” stands for privacy aware graph search procedure. Due to we have less data for privacy aware networks, the pa mode is replaced by the global networks.

Parameter Set Up (2)

- To be able to generate a recommendation, the recommender system must be trained to classify positive and negative reviews of shopping places dataset correctly. We test our system to do this sentiment analysis task using five different classifiers: backpropagation neural nets that run individually (bp_1, bp_2, bp_3), and an ensemble backpropagation which runs bp_1, bp_2, bp_3 simultaneously in ensemble mode, and an SVM classifier. We use ensemble since combining multiple classifiers leads to better performance than using only an individual one.
- The bp_1 has learning rate=0.00005, 100 hidden layers, maximum iterations of 100 iterations. The bp_2 has learning rate=0.0001, 200 hidden layers, maximum iterations=500. The bp_3 has a learning rate=0.00015 and 300 hidden layers with maximum iterations of 1000 iterations. We use the SVM classifier with nonlinear degree 3 of RBF kernel and C-support vector classification parameter equal to 3 with a maximum 1000 iteration. A large C gives the system low bias and high variance.

Evaluation Metrics

In our experiments, we compare the results for different methods. Since we can view generating recommendations as a kind of link prediction task, we use R-score (Breese et al., 1998) [18] and average over links to handle the extreme case of scarce local experts and top-rated local experts (Singh et al., 2007)[10]. By a user's likelihood:

$$p(u,t) = 2^{-\frac{(j_t - 1)}{(\lambda - 1)}} \tag{1}$$

the jt is the position of a shopping place t on a ranked list of statistic-based shopping places, which has the most positive reviews in decrease order. The is equal to the number of walking steps (or walking steps for short) divide by 2 (a position in the list with 50-50 stopping chance). As one of the evaluation tools, we compare two parameters of R-score: the R] that each, respectively, is an expected utility and the maximum possible R-score.

Evaluation Metrics (2)

- Given recommendation places from a local expert, R-score computes:

$$R_u = \sum_{i=1}^{n} p(u, g_i) \bigcup (u_i, g_i) \quad (2)$$

where n = number of places with is a maximum p(u,gi)∪(ui,gi).

- A utility function to predict whether a local expert candidate has a non-popular shopping place defined as

$$U(u_i, g_i) = -\log\left(\frac{1}{len(nodes)} * walking\_steps\right) \quad (3)$$

where len(nodes) is the number of local expert candidates.

- For validity, the maximum possible R-score is achieved by

$$E[R_u] = \frac{1}{m} \sum \frac{R_u}{R_u^{\max}} \quad (4)$$

m=10, as we generate ten times of recommendation for each user query.

Evaluation Metrics (3)

- An excellent recommendation system must have low MSE.
- We also compute precision at x, or p@x, as the second metric where x={1,3,5,10,15,20,30} with value range of [0; 1].
- p@x defined as the number of correct recommendations over x gold standard recommendation.
- A gold standard dataset generated by comments or user reviews about shopping places that meet user query criteria and sort in decrease order based on the most positive reviews.
- Further, to measure scalability, we compute the computational cost of time for building graph (tgraph), time for running random walker (talgo), time for others required for producing output from recommender systems (tother), and total execution time ($t_{total} = t_{graph} + t_{algo} + t_{other}$).
- An online system needs a slightly fast running time.

Results
Sentiment Analysis on User Reviews in Shopping Places Dataset

Classifier	*System Validation (%)*	*System Testing (%)*
bp_1	100	96.45
bp_2	100	96.45
bp_3	100	96.58
Ensemble_bp	100	96.45
SVM RBF degree 3	100	100

Results (2)
Sentiment Analysis on User Reviews in Shopping Places Dataset

p@1	*1*	*2*	*3*	*4*	*5*	*6*
betweenness	0.4	0.6	0.4	0.4	0.2	0.2
closeness	0.4	0.6	0.4	0.4	0.2	0.2
degree	0.4	0.6	0.4	0.4	0.2	0.2
pagerank	0.2	0.6	0.2	0.2	0.2	0.2
rw k=5	0.34	0.36	0.46	0.38	0.28	0.38
rw k=7	0.13	0.15	0.15	0.23	0.08	0.15
rw k=9	0.08	0.25	0.15	0.25	0.16	0.1
lrw k=5	0.5	0.32	0.5	0.44	0.3	0.26
lrw k=7	0.35	0.2	0.13	0.53	0.1	0.13
lrw k=9	0.35	0.25	0.1	0.55	0.1	0.06

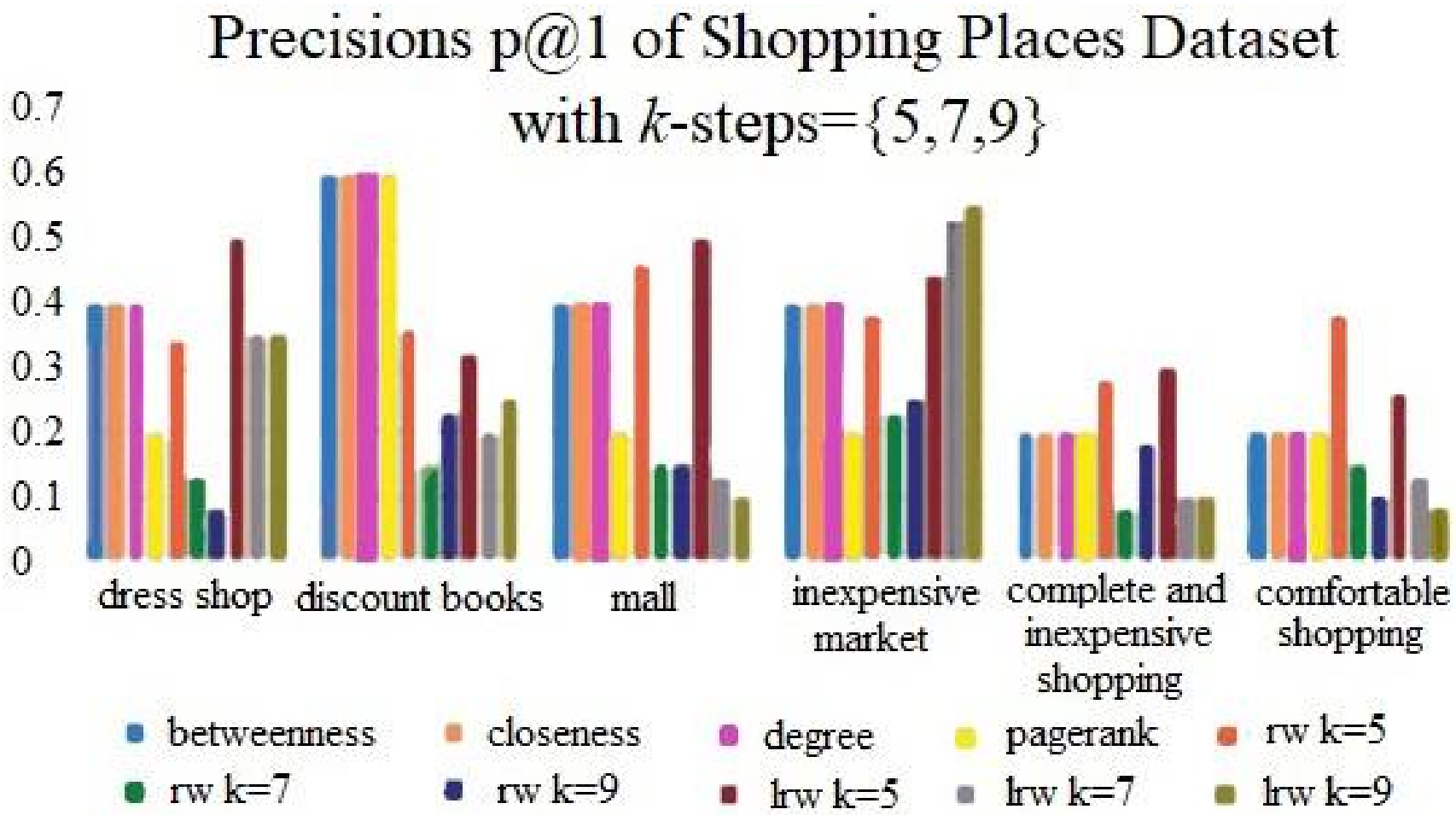
p@3	*1*	*2*	*3*	*4*	*5*	*6*
betweenness	***0.25***	***0.25***	***0.5***	***0.33***	***0.2***	***0***
closeness	***0.25***	***0.33***	***0.5***	***0.33***	***0.2***	***0***
degree	***0.25***	***0.33***	***0.5***	***0.33***	***0.2***	***0***
pagerank	***0.17***	***0.25***	***0.33***	***0.33***	***0.27***	***0***
rw k=5	***0.18***	***0.26***	***0.26***	***0.36***	***0.24***	***0.19***
rw k=7	***0.16***	***0.26***	***0.29***	***0.18***	***0.09***	***0.19***
rw k=9	***0.18***	***0.28***	***0.26***	***0.2***	***0.13***	***0.17***
lrw k=5	***0.17***	***0.27***	***0.27***	***0.37***	***0.21***	***0.12***
lrw k=7	***0.19***	***0.28***	***0.24***	***0.24***	***0.08***	***0.12***
lrw k=9	***0.2***	***0.27***	***0.23***	***0.28***	***0.08***	***0.12***

p@5	*1*	*2*	*3*	*4*	*5*	*6*
betweenness	***0.25***	***0.25***	***0.5***	***0.1***	***0.05***	***0.15***
closeness	***0.25***	***0.25***	***0.5***	***0.1***	***0.05***	***0.15***
degree	***0.25***	***0.25***	***0.45***	***0.1***	***0.05***	***0.15***
pagerank	***0.1***	***0.2***	***0.5***	***0.15***	***0.05***	***0.15***
rw k=5	***0.2***	***0.25***	***0.25***	***0.23***	***0.11***	***0.19***
rw k=7	***0.2***	***0.25***	***0.25***	***0.19***	***0.1***	***0.21***
rw k=9	***0.2***	***0.27***	***0.25***	***0.18***	***0.1***	***0.15***
lrw k=5	***0.21***	***0.2***	***0.26***	***0.22***	***0.07***	***0.16***
lrw k=7	***0.21***	***0.26***	***0.26***	***0.21***	***0.07***	***0.14***
lrw k=9	***0.21***	***0.21***	***0.25***	***0.22***	***0.06***	***0.15***

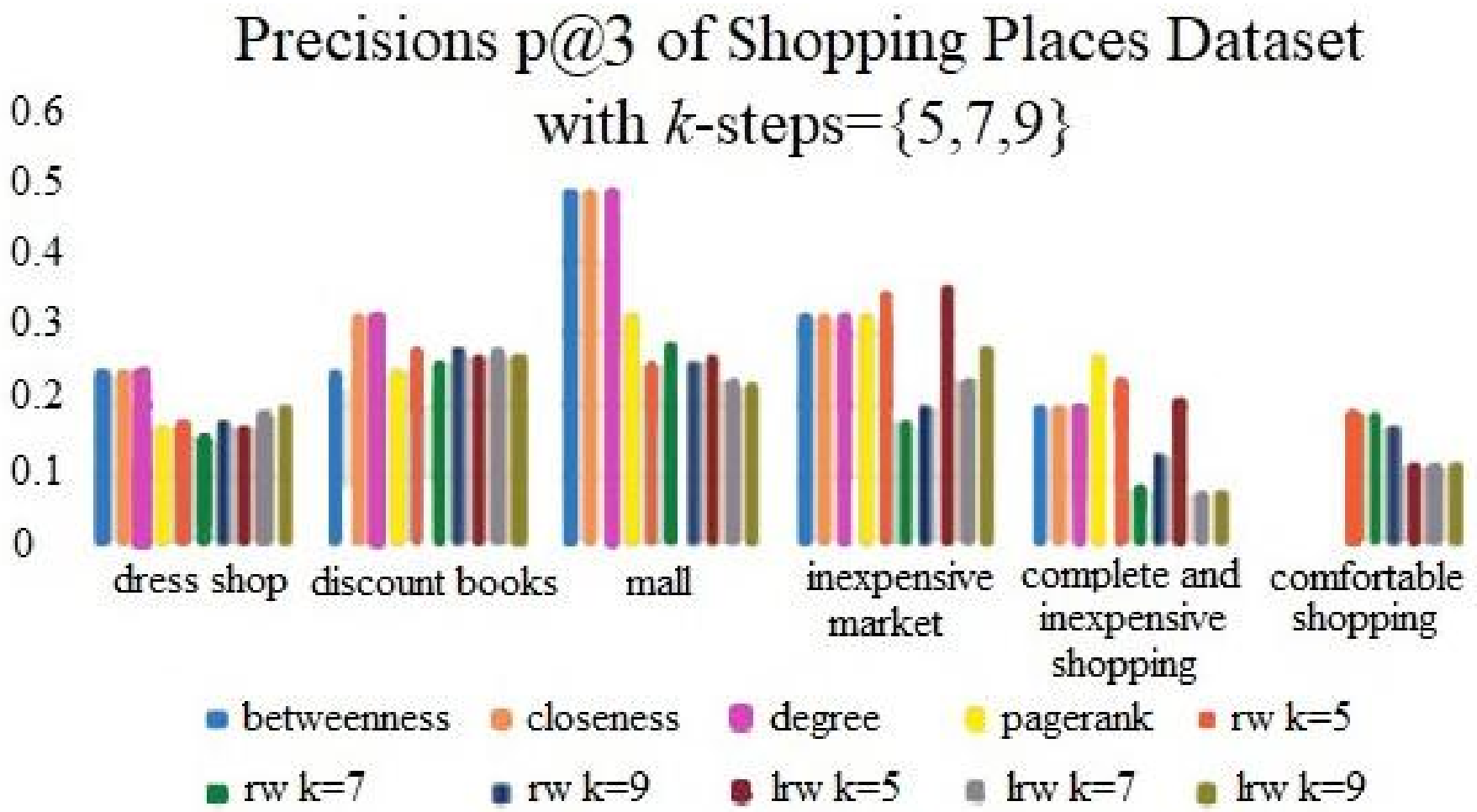
[a.] Queries: 1: dress shop ; 2: discount books; 3: mall; 4: inexpensive market; 5: complete and inexpensive shopping; 6: comfortable shopping.

[b.] Methods: rw: random walk; lrw: lazy random walk.

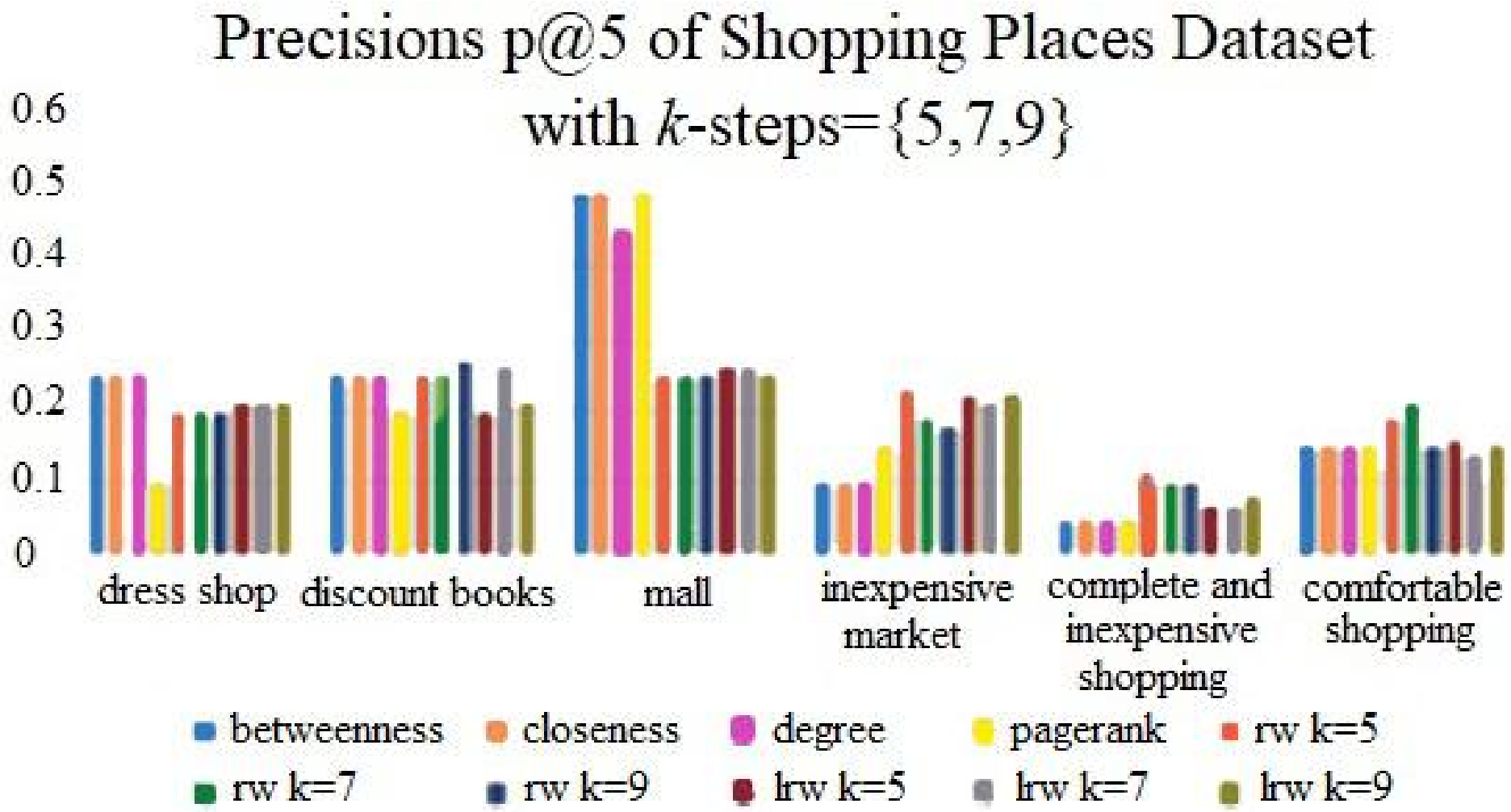
Results (3)


Precisions p@1 of Shopping Places Dataset
with k-steps={5,7,9}
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
dress shop
discount books
mall
inexpensive market
complete and inexpensive shopping
comfortable shopping
betweenness
closeness
degree
pagerank
rw k=5
rw k=7
rw k=9
lrw k=5
lrw k=7
lrw k=9

Results (4)


Precisions p@3 of Shopping Places Dataset
with k-steps={5,7,9}
0.6
0.5
0.4
0.3
0.2
0.1
0
dress shop
discount books
mall
inexpensive market
complete and inexpensive shopping
comfortable shopping
betweenness
closeness
degree
pagerank
rw k=5
rw k=7
rw k=9
lrw k=5
lrw k=7
lrw k=9

Results (5)


Precisions p@5 of Shopping Places Dataset
with k-steps={5,7,9}
0.6
0.5
0.4
0.3
0.2
0.1
0
dress shop
discount books
mall
inexpensive market
complete and inexpensive shopping
comfortable shopping
betweenness
closeness
degree
pagerank
rw k=5
rw k=7
rw k=9
lrw k=5
lrw k=7
lrw k=9

Results (6): An Example

Query=”mall”, City=”Bandung”, Mode=”Global”,
Order By Highest Positive Comments and Lowest Negative Comments

Query	***Mall, city=”Bandung”, mode=”global”***
lrw + SVM	[(‘Istana Plaza (IP)’,24L,5L),(‘LotteMart’,23L,3L), (‘Gramedia’,20L,3L)]
rw + SVM	[(‘Istana Plaza (IP)’,24L,5L),(‘LotteMart’,23L,3L),(‘METRO Department Store’,21L,6L)]
SVM gold standard	[(‘Istana Bandung Electronic Center (BEC)’,26L,1L),(‘Trans Studio Mall (TSM)’,24L,4L), (‘Istana Plaza (IP)’,24L,5L),(‘Cihampelas Walk (CiWalk)’,23L,1L),(‘Lotte Mart’,23L,3L),(‘Braga City Walk’,23L,4L),(‘Bandung Trade Centre - BTC Fashion Mall’,23L,3L),(‘Bandung Indah Plaza (BIP)’,21L,2L),(‘riaujunction’,21L,5L),(‘METRO Department Store’,21L,6L),(‘Gramedia’,20L,3L),(‘Festival Citylink’,19L,5L),(‘Paris Van Java (PVJ)’,18L,3L),(‘Liana swalayan’,5L,0L),(‘Living Plaza Dago’,3L,0L)]

lrw: lazy random walk; *rw*: random walk; *SVM*: Support Vector Machines; *bp*: backpropagation; *ens*: ensemble; *btw*: betweenness, *deg*: degree, *pr*: pagerank; *cls*: closeness graph.
(‘LotteMart’,23L,3L) means the query result has twenty three positive comments and three negative comments.

Results (6): An Example (cont'd)

Query=”mall”, City=”Bandung”, Mode=”Global”,
Order By Highest Positive Comments and Lowest Negative Comments

Query	***Mall, city=”Bandung”, mode=”global”***
btw + ens. bp	[(‘Trans Studio Mall (TSM)’,27L,1L),(‘Istana Plaza (IP)’,25L,4L),(‘Festival Citylink’,21L,3L)]
cls + ens. bp	[(‘Trans Studio Mall (TSM)’,27L,1L),(‘Istana Plaza (IP)’,25L,4L),(‘Festival Citylink’,21L,3L)]
deg + ens. bp	[(‘Trans Studio Mall (TSM)’,27L,1L),(‘Istana Plaza (IP)’,25L,4L),(‘Festival Citylink’,21L,3L)]
pr + ens. bp	[(‘Trans Studio Mall (TSM)’,27L,1L),(‘Istana Plaza (IP)’,25L,4L),(‘Festival Citylink’,21L,3L)]

lrw: lazy random walk; *rw*: random walk; *SVM*: Support Vector Machines; *bp*: backpropagation; *ens*: ensemble;
btw: betweenness, *deg*: degree, *pr*: pagerank; *cls*: closeness graph.
(‘LotteMart’,23L,3L) means the query result has twenty three positive comments and three negative comments.

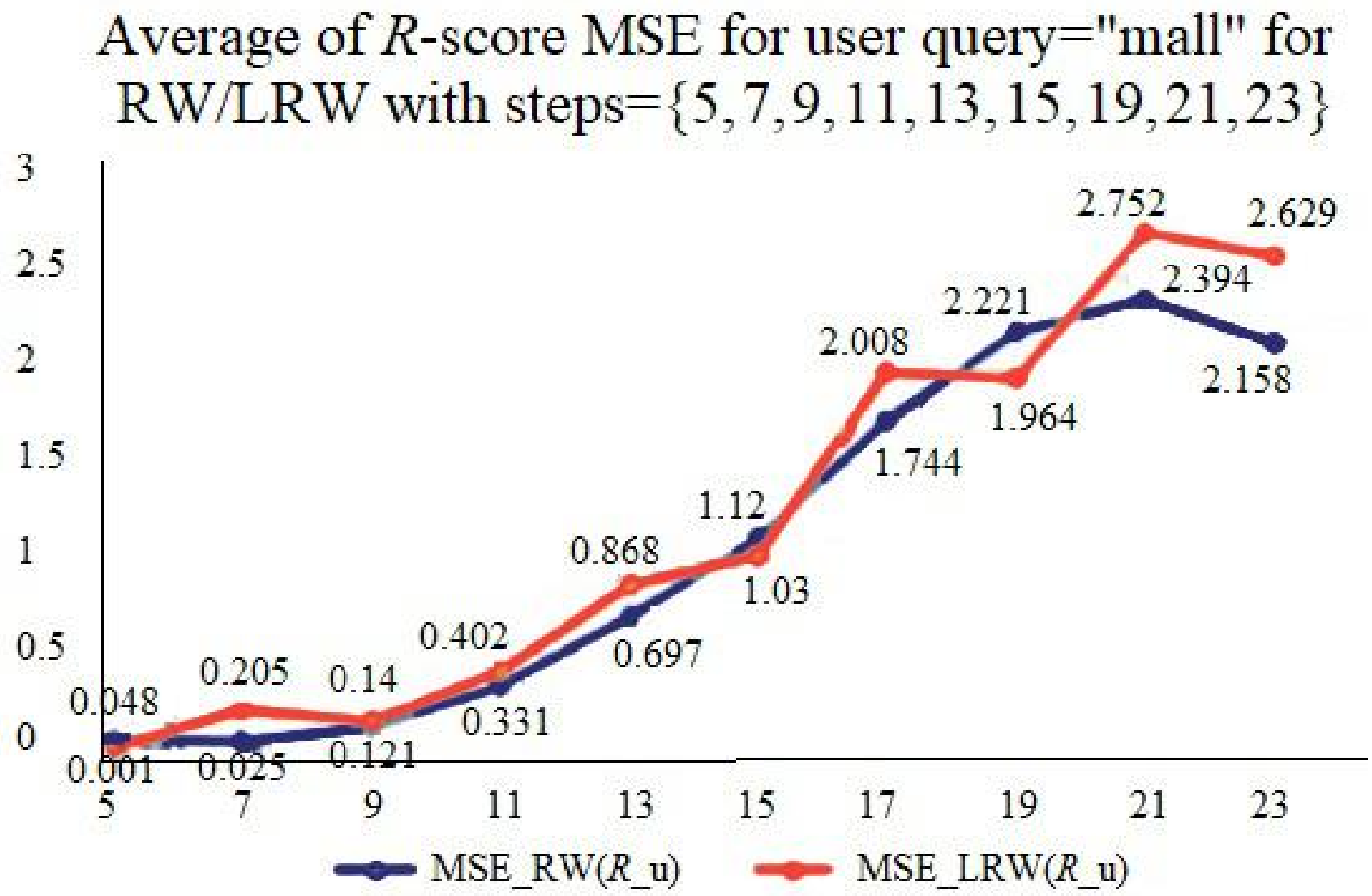
Results (6): An Example (cont'd)

Query=”mall”, City=”Bandung”, Mode=”Global”,
Order By Highest Positive Comments and Lowest Negative Comments

Query	***Mall, city=”Bandung”, mode=”global”***
ens. bp gold standard	[(‘Setiabudhi Supermarket’,29L,1L),(‘Trans Studio Mall (TSM)’,27L,1L),(‘Istana Bandung Electronic Center (BEC)’,26L,1L),(‘Istana Plaza (IP)’,25L,4L),(‘Bandung Trade Centre - BTC Fashion Mall’,25L,1L),(‘Toko Buku Togamas’,24L,4L),(‘riaujunction’,24L,2L),(‘METRO Department Store’,24L,3L),(‘Gramedia’,22L,1L),(‘Braga City Walk’,22L,5L),(‘Cihampelas Walk (CiWalk)’,21L,3L),(‘Festival Citylink’,21L,3L),(‘Bandung Indah Plaza (BIP)’,21L,2L),(‘Paris Van Java (PVJ)’,20L,1L),(‘LotteMart’,19L,7L), (‘Liana swalayan’,5L,0L)]

lrw: lazy random walk; *rw*: random walk; *SVM*: Support Vector Machines; *bp*: backpropagation; *ens*: ensemble; *btw*: betweenness, *deg*: degree, *pr*: pagerank; *cls*: closeness graph.
(‘LotteMart’,23L,3L) means the query result has twenty three positive comments and three negative comments.

Results (7)


Average of R-score MSE for user query="mall" for RW/LRW with steps={5,7,9,11,13,15,19,21,23}
3
2.5
2
1.5
1
0.5
0
0.001
0.048
0.025
0.205
0.121
0.14
0.331
0.402
0.697
0.868
1.12
1.03
1.744
2.008
1.964
2.221
2.394
2.158
2.752
2.629
5
7
9
11
13
15
17
19
21
23
MSE_RW(R_u)
MSE_LRW(R_u)

Analysis

- About systems scalability, for both of Lazy Random Walk (LRW) and Random Walk (RW), most of the time (93%) was consumed in t(graph stage), while the 3% of the time is on talgo graph (the RW/LRW stage).
- The remaining time (4%) was for t (output or graphical user interface systems).
- Our recommendation systems itself is very scalable that it needs only a total of 3.215 seconds for the random walk and 3.222 seconds for the lazy random walk.
- The time difference between the two methods is insignificant due to the small size of our used dataset.

Analysis (2)

- From the results in TABLE II, the precisions of Lazy Random Walk and the Random Walk can reach sufficiently high scores (0.5, 0.37, and 0.26 for p@1, p@3, and p@5 for Lazy Random Walk and 0.46, 0.36, and 0.25 for p@1, p@3, and p@5 for Random Walk).
- Moreover, since they are also slightly fast, then both methods are potential for online implementation, embedded in a recommendation system.
- Another advantage, because it is dynamic, any recommendation systems that utilized either Random Walk or Lazy Random Walk method are natural to be maintained due to their databases no need to be very frequently updated to produce useful recommendations.

Conclusions

- Both Lazy Random Walk and Random Walk methods are potential for online implementation in a recommendation system since they offer slightly fast running time with reliable results.
- From the industry perspective, both techniques provide easiness viz. infrequent updating time but still produce a good recommendation.

References

- [1]Antin, J., de Sa, M., Churchill, E. F.: Local Experts and Online Review Sites. In: ACM 2012 Conference on Computer Supported Cooperative Work Companion, CSCW 12, pp. 55-58. ACM, New York (2012)
- [2]Jindal, T.: Finding Local Experts From Yelp Dataset. Master thesis, Master of Science in Computer Science, Graduate College of the University of Illinois at Urbana Champaign (2015)
- [3]Torres, S.D., Hiemstra, D., Weber, I., Serdyukov, P.: Query Recommendation for Children. CIKM'12 (2012)
- [4]Bauman, K., Tuzhilin, A.: Discovering Contextual Information from User Reviews for Recommendation Purposes. In: CBRecSys 2014, pp. 2{9. CBRecSys 2014, Silicon Valley, CA, USA (2014)
- [5]Bahmani, B., Chowdhury, A., Goel, A.: Fast Incremental and Personalized PageRank. In: Proceedings of the VLDB Endowment, Vol. 4, No. 3, pp. 173{184. VLDB Endowment, Seattle, WA (2010)
- [6]Zhang, L., Li, C.: A Novel Recommending Algorithm Based on Topical PageRank. In: Wobcke, W., Zhang, M. (eds.) AI 2008. LNAI, vol. 5360, pp. 447{453. Springer, Berlin, Heidelberg (2008)
- [7]Nguyen, P. T., Tomeo, P., Di Noia, T., Di Sciascio, E.: An Evaluation of SimRank and Personalized PageRank To Build A Recommender System for the Web of Data. In: WWW 2015 Companion, pp. 1477{1482. ACM, New York, NY, USA (2015)
- [8]Sawant, S.: Stanford CS224w Collaborative Filtering using Weighted Bipartite Graph Projection. A Recommendation System for Yelp, snap.stanford.edu/class/cs224w-2013/projects2013/cs224w-038-final.pdf
- [9]Sulieman, D., Malek, M., Kadima, H., Laurent, D.: Combining Social and Semantic Information for Recommendation: Comparative Study. J. Hermes Science Publication, no y/2013, 1{11 (2013)
- [10]Singh, A. P., Gunawardana, A., Meek, C., Surendran, A.C.: Recommendations using Absorbing Random Walks. North East Student Colloquium on Artificial Intelligence (NESCAI)(2007)