VOL. 10, NO. 16, SEPTEMBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2015 Asian Research Publishing Network (ARPN). All rights reserved.

www.arpnjournals.com

WEB BASED AGENT AND ASSERTION PASSIVE GRADING FOR INFORMATION RETERVIAL

K.Sridharan[1] and M. Chitra[2]
[1]Department of Information Technology, Panimalar Engineering College, Chennai, Tamilnadu, India
[2]Department of Information Technology, Sona College of Technology, Salem, Tamilnadu, India
E-Mail: sridharank.p@gmail.com

ABSTRACT

We propose a automated solution for ranking query results of Webdatabases in a consumer- and query-dependent environment. We first derive the ranking function for a consumer query by investigating users browsing choices over individual query results. Based on this workload, we propose a similarity model, based on two novel metrics - user- and query similarity, for ranking query results when user browsing choices are not available and query similarity, a metric of similarity between queries, estimated using two independently proposed measures, first one is query-condition similarity, and other one is query-result similarity. We present the results of an experimental study that validates our proposal for user and query-dependent ranking.

Keywords: information retrieval, query mining, data mining.

INTRODUCTION

With the emergence of the deep Web, a large number of Web databases and their related applications (e.g., airline reservations, vehicle search, real estate scouting,) have proliferated. Since these databases support a simple Boolean query retrieval model, it often leads to situations when *too many* results are generated in response to a query.

In order to find the results that match one's interest, the user has to browse through this large result set - a tedious and time-consuming task. Currently, Web databases simplify this task by displaying these results in a *sorted* order on the values of a *single* attribute (e.g., Price, Mileage,etc.). However, most Web users prefer to simultaneously examine multiple attributes and their values (instead of a single attribute) while selecting results relevant to them, and these preferences vary as the user queries change.

The current sorting mechanism adopted by Web databases does not hold the ability to perform such user and query dependent ranking. Some of the proposed extensions to SQL allow *manual* specification of attribute weights (and thus, the ranking function), an approach cumbersome for most Web users.

However, the problem being that domain experts establish a relative ordering between the attributes (e.g., "Price" and "Mileage" are more important than "Color" in the domain of *vehicles*) without estimating the absolute attribute weights and the preferences for attribute values (e.g., one user may prefer „red‟ colored car whereas another may prefer „blue‟ colored car) that can translate to corresponding ranking functions. *Automated* ranking of database results has been studied in the context of relational databases and the most commonly proposed technique is to derive a ranking function that is either *user-independent* or *query-independent* (orboth i.e., a *single* function) for ordering the tuples.

In the context of Web databases, as motivated by the above examples, an ideal ranking model should consider both the dimensions - user and query in conjunction, to form a robust framework for ranking.

We propose such a *user*- and *query*-dependent approach for rankingWeb database query results. We infer the ranking function for a user query via a learning technique that analyses the users *browsing choices* over the query results. Unlike relational databases, the nature of Web database applications allows users to browse and select the results that match their preferences (through an interaction with the Web pages containing the result tuples).

In order to tackle this issue, we propose a novel *similarity*-based technique based onthe intuition that - i) similar users, for the same query, tend to display similar ranking preferences, and ii) the same user tends to display similar ranking preferences over the results of similar queries.

We formalize these notions of similarity into - 1) *consumersimilarity*, a metric of similarity between users, estimated based on their past browsing choices, and 2) *query similarity*, a metric of similaritybetween queries, estimated using two independently proposed measures - i) *query-condition similarity*, and ii) *query-result similarity*. The formerestablishes query similarity by equating the conditions in the respective queries, whereas the latter establishes it by collating their corresponding results.

Based on this setting, we combine the above two metrics into a single *Similarity Model* to determine *the most similar query* asked by *the most similar user* from a workload of past user queries for which functions are inferred based on browsing choices.

Contributions: The contributions are:

a) We propose an automated solution for ranking Web database query results in a *user*-and *query*-dependent environment.
b) Our proposed approach derives the ranking function for an user query by analyzing the users *browsing choices* over thequery results.
c) We propose a *Similarity Model*, based on two novel measures -*query similarity* and *user similarity*, to derive rankingfunctions for user queries whose browsing choices are unknown at query time.

RELATED WORK

This model relies on the availability of a workload of queries spanning all attributes and values to establish a score for a tuple. However, a drawback of such a workload is that in the context of Web databases, user queries are restricted to a subset of the attributes that are displayed in the results. In such a setting, the workload will fail to capture user preferences towards those attributes and values that cannot be specified in the query. In contrast, we capture these preferences via users‟ browsing choices in a query- and user-dependent setting

The work proposed for query-dependent ranking analyzes the relationship between the query results and the tuples in the database. However, a major drawback of this work lies in the fact that it requires the knowledge of the complete underlying database at all times to rank query results, an improbable setting for Web databases that dynamically obtain data from a slew of individual sources.

The use of query similarity has been widely studied in Information Retrieval and Collaborative Filtering. However, database queries involving multiple combinations cannot be directly compared like IR-keyword queries. In this paper, we propose a novel notion of database query similarity that is determined by analyzing the results for individual queries. Although, an intuitive mechanism for establishing user similarity based on profiles has been studied, it involves the use of domain experts in addition to learning models to derive this similarity.

Alternatively, we propose a mechanism to capture user similarity by analyzing the relationship between the users past browsing choices. To the best of our knowledge, *user*- and *query-dependent* ranking in the context of Web database queries has not been studied in literature

PROBLEM DEFINITION AND ARCHITECTURE

We now formally define the problem of ranking in Web databases, and outline a general architecture of our solution.

Problem definition

Consider a Web database table *D* over a set of *m* attributes, *A* = {A1,A2, … Am}. An user *U* asks a query *Q* of the form“SELECT * FROM *D* WHERE X1 = x1 AND …AND Xs = xs”, where each Xi ∈*A* and xi is a value in its domain. Let *N* = {t1, t2, ...,tn} be the set of resulttuples for *Q*.

The ranking problem can be stated as: “*For a query Q given byuser* U*, determine a* ranking functionFUQ *that assigns a score to everytuple from which a ranking order for* N *can be established*”. As spelled out in Section 1, we categorize this problem into two individual sub-problems:

a) Inferring ranking functions using browsing choices

Givena user U, the query Q, and the set of results N, let R (⊂ N) be the set of results generated based on U‟s browsing choices over N. The ranking problem can then be stated as: “*Using* R *and* N*, determine the ranking function* FUQ *that captures* U*'s preferences over* Q*'s results.*”

b) Inferring ranking functions using similarity measures

Consider that a user (U) browsing choices for the results of a query (Q) are not available. In this setting, the ranking problem is stated as: “Assuminga workload of past user queries for which the ranking functions are derived prior to search, determine the ranking function (FUiQj) of the most similar user (Ui) to U, derived for the most similar query (Qj) to Q,for ranking Q’s results.

Ranking architecture

The architecture for our *user*-and *query*-dependent ranking framework (shown in Figure-1) comprises of two components for addressing the sub problems defined above. In the first component *inferring Ranking Functions using Browsing Choices*, the user‟s (U)browsing choices over the query results (N) produces the set of relevant results (R). Both these sets (N and R) are fed to the *LearningModel* that deduces the-i)significance of each attribute to establish the set of *attribute-weights*, and ii) emphasis given by users to particular values of an attribute. The attribute-weights are then integrated into the ranking function F that assigns a score to every tuple t in N.

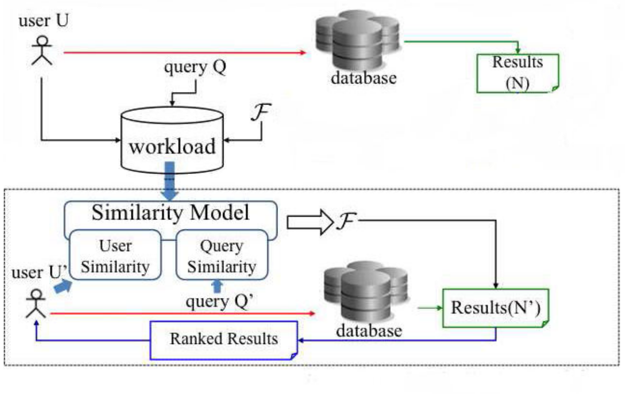


Figure-1. Ranking architecture.

SIMILARITY MODEL FOR RANKING FUNCTION

For a Web database, it is impossible to procure the browsing choices for every user across the results of every query. In order to derive the ranking functions for such user queries, we are proposing the use of a *Similarity Model* that is based on the proposed notions of *query*- and *user-similarity*.

Query similarity

For a single user (U1), let N1 be the results for a query (Q1) for which no ranking function exists.

Consider U1s workload that contains a set of queries - {Q2, Q3, ...,Qr,}, and let {F12, F13, ..., F1r} be the ranking functions derived for each of these individual queries. Based on this information, it is useful to infer if any of the previously derived ranking functions can be used for ranking the results of Q1. The same user may have different ranking functions for different queries. Consequently, we cannot randomly choose a ranking function from U1s workload and use it to rank N1. Hence, in order to determine an appropriate function, we introduce the notion of *querysimilarity*. We advance the theory thatif Q1 is most similar to a past query Qj, then the ranking function (F1j) derived for Qj can be used to rank the results of Q1. In order to translate this proposal into a principled approach, we introduce two independent metrics for establishing similarity between queries - i) *query-condition* similarity, and ii) *query-result* similarity.

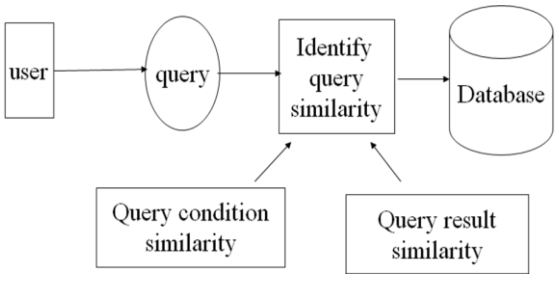


Figure-2. Data flow for Query similarity.

Query-condition similarity

We estimate the *similarity* between two queries by analyzing the relationship between the attribute values in the respective query conditions.

Consider two queries - Q and Qj, each with the conjunctive selection conditions of the form
“WHERE X1=x1 AND ….AND
Xs=xs” and “WHERE X1=x1 AND

… AND Xs=xs” respectively. The *query-condition* similarity between Qand Q‟ is represented as the product of the similarities between the values xi and x‟i for every attribute Xi

Query-result similarity

In this metric, the *similarity* between two queries is determined by comparing their *results*. In the previous Section, we established similarity between values by computing the similarity of the results generated in response to queries containing these values.

Consider two queries Q and Qj, each with the conjunctive selection conditions of the form “WHERE X1=x1 AND …AND Xs=xs” and “WHERE X1=x1 AND ·…AND Xs=xs” respectively.

User similarity

In order to determine *user similarity*, we establish a similarity between their browsing choices i.e., their ranking functions. In order to establish the similarity, we analyse only the former set of queries i.e., for which ranking functions exist for both users, and the similarity is then expressed as the combined similarity between the ranking functions derived for these queries.Consider two users U1 and U2 asking the same set of queries - {Q1, Q2,...,Qr} for which ranking functions ({F12, F13, ..., F1r} and {F22, F23, ..., F2r}) have been derived based on their browsing choices. Then, the *user similarity* between U1 and U2 is expressed as the average similarity between the individual ranking functions derived for U1 and U2 for each query Qj. In other words, in this hypothesis, two users who may not be very similar to each other over the entire workload comprising of similar and dissimilar queries, may in fact, be very similar to each other over a smaller set of similar queries. We formalize this hypothesis using two different models – i) clustered and ii) top-K – for determining user similarity. To determining the top-K for user similarity, Ranking is done using Ranking Algorithm. Ranking Algorithm determines ranking function for most similar query given by most similar user to rank the results of user. The algorithm begins by determining the query similarity between input query and every query in the workload. Based on these selected queries, the algorithm determines the user similarity between current user and every user in the workload. Finally it generates a list of all the user-query pairs and laniaries these pairs by assigning a rank (which is the sum of query and use similarity ranks) to each pair. Define the above process by

- The user Ui submits the query Qj
- The query-similarity model determines the set of queries ({Qj, Q1, Q2, ...,Qp}) most similar to Qj.
- the user-similarity model determines the set of users ({Ui, U1, U2, ...Ur}) most similar to Ui.
- Using these two ordered sets of similar queries and users, search the workload to identify the function FUxQy such that the combination of Ux and Qy is most similar to Ui and Qj.
- FUxQy is then used to rank Qj’s results for Ui.
- In order to rank result (Nj) the corresponding value weight and attribute weight for Fxy will be individually applied to each tuple of Nj.
- Ordering of tuple is achieved which is then displayed to user Ui.

THE SIMILARITY MODEL

In order to derived user's (U1) ranking function for a query (Q1), we have proposed two independent approaches based on *user-* and *query-*similarity. In ageneral context of Web database applications, it is highly unlikely that there exists a query Qj in U1"s workload that is very similar to Q1, thus hampering the applicability of only *query similarity*. Similarly, the likelihood of finding a user Ui, very similar to U1, and for whom a ranking function exists for Q1 is rare, thus, reducing the potential of achieving good ranking via only *user similarity*. In a more generic setting, the best a workload can achieve would be to contain the ranking function for a similar query (to Q1) derived for a similar user (to U1). In order to determine such a function, we combine the two measures into a single *Similarity Model*. The goal of this model is to determine a ranking function (Fij) derived for the most similar query (Qj) to Q1 given by the most similar user (Ui) to U1 to rank Q1"s results. The process for finding such an appropriate ranking function is represented by Algorithm 1.

```
Input: Query Q1, User U1, Workload W
Output: Ranking Function Fij
for each Ui(∈Uset= {U1, ...Up})in W do
        Calculate user-similarity (U1, Ui)
end
for each Qj(∈Qset= {Q1, ...Qr}) in W do
        Calculate query-similarity (Q1, Qj)
end
sort (Uset) // descending order
sort (Qset) // descending order
Initialize matrix F -Uset as rows and Qset as columns
for each Ui and Qj do
        ifFij∈ W then
                F[i][j] = Fij
        else
                F[i][j] = null
        end
end
        Fij = Get-RankFn(F)
return Fij
```

Algorithm 1: Inferring ranking functions using similarity

The input to the algorithm is the user (U1) and the query (Q1) along with the workload (W) containing ranking functions for past user queries. The algorithm begins by determining a *user similarity* between U1 and every user Ui (Step 1), and a *query similarity* between Q1 andevery query Qj (Step 2) from the workload. Based on these similarity calculations, we assign a rank to every query and user based on their similarity with Q1 and U1 respectively, such that a query very similar to Q1 gets a higher rank than the one less similar to Q1 (Steps 3 and 4)From this populated matrix, the *similarity model* needs to derive aranking function that can be used to rank the results of Q1 for U1. The „Get-RankFn" function (Step 8) aids this process by determining a cell (F[i][j]) in the matrix that satisfies both of the following two conditions:

a) *Condition-1*: F[i][j] 6= „null, and
b) *Condition-2*: rank(Ui) + rank(Qj) isminimum

The former condition is self-explanatory since we need to find a cell for which a function has been derived apriori. The second condition ensures that the *Similarity Model* determines the function that belongs to the most similar query (to Q1) derived for the most similar user (to U1).

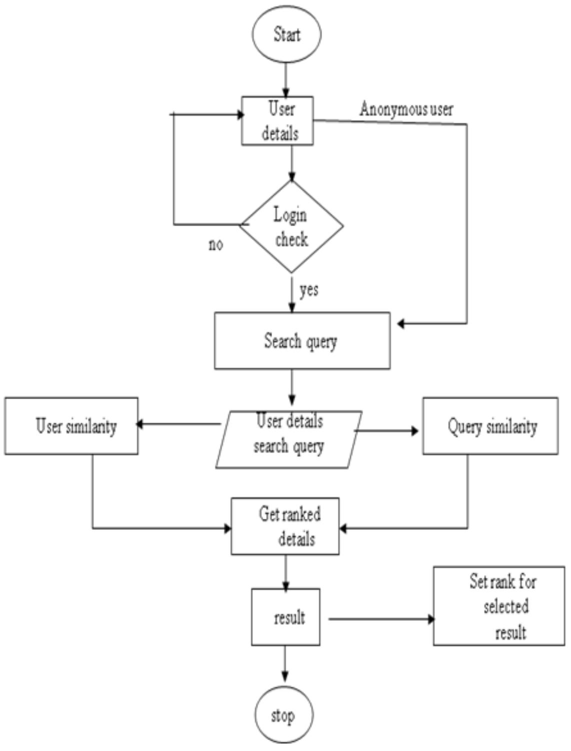
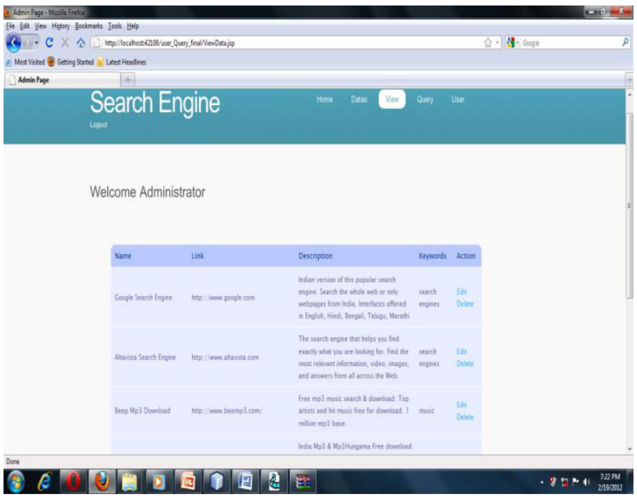


Figure-3. Overall data flow architecture.

The Data Flow Diagram for the overall system is shown in Figure-2. Here the user logs in and the login is validated and the query is being searched. The user details and the query that is being searched are segregated and the corresponding query similarity and user similarity is identified. The results are thereby ranked by implementing the Ranking Algorithm and the result is displayed. The rank must be set for the selected result. Initially the user logins and then searches the query, based on the user query pairs in the workload which is similar to the current user and query the ranking is assigned. User query pair is chosen in such a way that it is most similar to the current user query.

Experimental result

We use to evaluate the data extraction system through the user similarity and query similarity. We are constructed databases which contain the search result of web database. Normally these databases are accessed by user submit a query on web. For example, to find particular word "music", web server retrieved the requested information and generates the response web page. The experimental results are compared by based on user similarity and query similarity.


Admin Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:42108/user_Query_final/ViewData.jsp
Most Visited Getting Started Latest Headlines
Admin Page
Search Engine
Logout
Home
Datas
View
Query
User
Done
7:22 PM
2/19/2012

Welcome Administrator

Name	Link	Description	Keywords	Action
Google Search Engine	http://www.google.com	Indian version of this popular search engine. Search the whole web or only webpages from India. Interfaces offered in English, Hindi, Bengali, Telugu, Marathi	search engines	Edit Delete
Altavista Search Engine	http://www.altavista.com	The search engine that helps you find exactly what you are looking for. Find the most relevant information, video, images, and answers from all across the Web.	search engines	Edit Delete
Beep Mp3 Download	http://www.beemp3.com/	Free mp3 music search & download. Top artists and hit music free for download. 1 million mp3 base.	music	Edit Delete
		India Mp3 & Mp3Hungama Free download		

Figure-4. Sample database view.

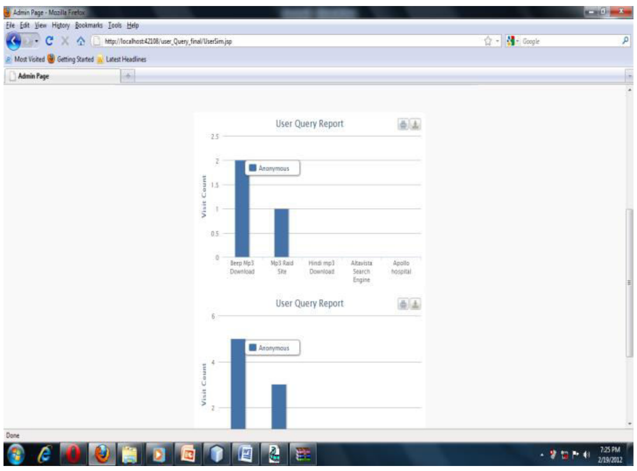
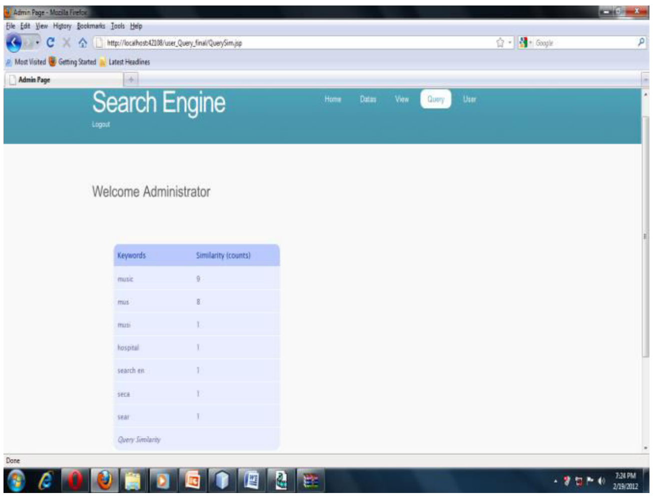


Figure-5. Sample user query report.


Admin Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:42108/user_Query_final/QuerySim.jsp
Most Visited Getting Started Latest Headlines
Admin Page
Search Engine
Logout
Home
Datas
View
Query
User
Done
7:24 PM
2/19/2012

Welcome Administrator

Keywords	Similarity (counts)
music	9
mus	8
musi	1
hospital	1
search en	1
seca	1
sear	1
Query Similarity	

Figure-6. Sample screen for similarity count.

CONCLUSIONS

We proposed a computerized solution for ranking query results of Web databases in an consumer- and query-dependent environment. We first derived the ranking function for an consumer query by investigating users" browsing choices over individual query results. Based on this workload, we propose a similarity model, based on two novel metrics – user- and query similarity, for ranking query results when user browsing choices are not available. We present the results of an experimental study that validates our proposal for user and query-dependent ranking. This is very efficient and more accuracy for producing accurate result for the given statement based on experts and statements.

REFERENCES

[1] Basu.C, Hirsh.H, and Cohen.W.W. (1998) 'Recommendation as classification: Using social and content-based information in recommendation', In AAAI/IAAI, pages 714–720.

[2] Hofmann.T. (2003) 'Collaborative filtering via Gaussian probabilistic latent semantic analysis', In SIGIR, pages 259–266.

[3] Huang. 2008. Similarity measures for text document clustering. In Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008), Christchurch, New Zealand. pp. 49-56.

[4] Liu.S M. X. Zhou, S. Pan, Y. Song, W. Qian, W. Cai, and X. Lian. 2012. TIARA: Interactive, Topic-Based Visual Text Summarization and Analysis. ACM Transactions on Intelligent Systems and Technology (TIST).3: 25.

[5] Naim and D. Gildea, 2012. Convergence of the EM Algorithm for Gaussian Mixtures with Unbalanced Mixing Coefficients.arXiv preprint arXiv: 1206.6427.

[6] P.Shamsinejadbabki and M. Saraee. 2012. A new unsupervised feature selection method for text clustering based on genetic algorithms. Journal of Intelligent Information Systems.38: 669-684.

[7] Pembe. 2010. Automated Query-Biased and Structure-Preserving Document Summarization for Web Search Tasks.

[8] Sun.L and A. Korhonen. 2009. Improving verb clustering with automatically acquired selectional preferences. In:Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: 2: 638-647.

[9] Shehata,S. F. Karray and M. S. Kamel. 2010. An efficient concept-based mining model for enhancing text clustering. Knowledge and Data Engineering, IEEE Transactions on.22: 1360-1371.

[10] Shi.X and Yang.C.C. (2007) 'Mining related queries from web search engine query logs using an improved association rule mining model', J. Am. Soc. Inf. Sci. Technol., 58:1871–1883.

[11] Silva and N. Marques. 2010. Feature clustering with self organizing maps and an application to financial time-series portfolio selection.In: International Conference on Neural Computation.

[12] Su.W, Wang.J, Huang.Q, and Lochovsky.F. (2006) 'Query result ranking over e-commerce web databases', In Conference on Information and Knowledge Management (CIKM), pages 575–584.

[13] Telang.A, Chakravarthy.S, and Li.C. (2010) 'Establishing a workload for ranking in web databases', Technical report, UT Arlington.

[14] Xu.Z, Luo.X, and Lu.W. (2010) 'Association link network: An incremental semantic data model on organizing web resources', In Intl. Conf. on Parallel and Distributed Systems (ICPADS), pages 793–798.

[15] Zhou.T.C, Ma.H, Lyu.M.R, and King.I. (2010) 'Userrec: A user recommendation framework in social tagging systems', In AAAI