IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

Semantic Search on Applicant Tracking System

Le Quan Ha[1] and Mainur Rahman[2]

School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast BT3 9DT, United Kingdom[1]

Department of Computer Science, University of Calgary, Calgary, Alberta T2N 1N4, Canada[2]

Abstract: Web search engines often federate many user queries to relevant structured databases. For example, a recruitment-related query might be federated to a jobseekers-and-employers database containing their resumes and skills. The relevant structured data items are then returned to the user along with web search results. Though each structured database is searched in isolation, the search often produces empty / incomplete results as the database may not contain the required information to answer the query. Starting from our Applicant Tracking System (ATS), we have 16 development databases of over 650,000 profile documents of resumes / cover letters / skills. There are on average 238 keywords per document. In fact, per minute there can be up to 200,000 transactions within all these databases. Our existing traditional database search technique (by full-text keyword PostgreSQL search) can be frozen or taking very long to respond unless if we cut off / only search from top profiles, we will not have the search results ready by thirty seconds, but this cut-off limitation returned incorrect results; for example for a query "Jet fuel Thermal Oxidation" to request information about job seekers whose resumes contain skills in Oil and Gas industry, in the top ten results there was a conflict in relevance ranking. In order to research a more suitable full-text keyword search technique better than the existing database search, we considered employment of semantic search models. Our semantic search technique has 88% - 91.22% accuracy with very much quicker queries that can help users to make a search of 4 keywords of skills completed from 1 second to 28 seconds. Furthermore, the semantic search engine becomes very strong that users can search by entering a whole text paragraph. We designed a combination of semantic search that look for web pages per search and database search.

Keywords: applicant tracking system, ATS, click model, entity retrieval, entity ranker, keyword extraction, ontology, PostgreSQL, resource description framework, RDF, semantic search, Triplify.

I. INTRODUCTION

Web search engines include the four most commonly used techniques below [47].
(i) In document-based partitioning, the web pages are divided among the nodes; each maintains local inverted lists of the assigned web pages. A query is broadcast to all nodes then each returns the k most highly ranked pages. Its accuracy is 75% - 95% but the total query number/second must be small [4, 34].
(ii) For keyword-based partitioning [38], each node maintains the inverted lists of some keywords. A query with $k \geq 1$ keywords contacts k nodes and requires that the inverted lists of $k-1$ keywords be sent over the network. Its accuracy is 100% but its cost grows linearly with the web document number in the system.
(iii) Hybrid indexing has small communication cost per query which is independent of the total number of web documents that is also very helpful for a large number of queries each second. However, it has the cost of 10%–50% accuracy degradation and it requires significant amount of extra storage. [39]
(iv) Semantic search [15] seeks to improve search accuracy by understanding searcher intent and the contextual meaning of terms as they appear in the searchable data space to generate more relevant results. It can identify those web pages with similar concepts; retrieve documents containing "automobiles" for queries containing "cars". Hence, semantic search is suitable for huge databases/concept-based queries.

Starting from our Applicant Tracking System (ATS), user queries issued against profile search engines did not look for web pages per search as Google or Yahoo, instead ATS requested information from huge structured databases [1] then showed ranking candidate results on web pages. Candidates could be ranked on all information collected on them – resume, skills, cover letters and dynamic form information. In order to provide more informative results for these scenarios, the web query is federated to one or more structured databases. Each structured database is searched individually and the relevant structured data items are returned to the web search engine. Similarly, based on the collected information, jobseekers will be consistently informed of current/future positions that are most like their requirements. The principal function of an ATS is to provide a central location and database for a company's recruitment efforts. ATS are built to better assist management of resumes and applicant information. Data is either collected from internal applications via the ATS front-end, located on the company website or is extracted from

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

applicants on job boards. Functionality of an ATS is not limited to data mining and collection. ATS applications in the recruitment industry include the ability automate the recruitment process. This process requires finding qualified candidates for specific job type within their boarder candidate database. Similarly, the candidates should be able to find suitable jobs from the database. Recently when users can also use mobile browsers, then the needs of ATS profile searches on cell phones also happen daily. Tempo, a Webkit-based browser using semantic search techniques, can improve the mobile browser delay by 1 second (20%) [44]. Semantic search was also applied for Facebook [31] applications on cell phones. We combined semantic search and database search together, now better than only semantic search or database search, user queries can be served by information from a recruitment database combined to semantic search together in our system.

II. RELATED WORK

Candidate resumes and job description contains complex data set and thus requires complex queries involving inference and reasoning for producing meaningful search results. The semantic technology encodes meanings separately from data and content files. It enables machines as well as people to understand, share and reason with them at execution time. In order to start our experimental design, the structures or concepts of entities so called the Ontologies are designed and stored [9]. For example, the ontology of jobseekers in our system has 44 attributes: jobseeker ID, first name, last name, salutation, login and password, etc.
For example, Leo Kant is a jobseeker registered with us, “Leo_Kant” becomes an entity with attribute values (jobseeker ID 80108, first name “Leo”, last name “Kant”, salutation “Mr”, login “lkant”, etc.)
From Leo’s profile documents (resume and cover letter) that are either in HTML or MS Word file format, we have to transform these raw content into semantic metadata in resource description framework (RDF) format; RDF [46, 12, 24] includes entities, events and facts with entities scores [2, 11, 14, 32, 40, 48]. In this Leo Kant’s documents, there are 2 countries “India” occurring 6 times and “Canada” appearing 4 times, many IT skills are detected “ms sql server” and “visual basic”, etc. Based on this frequency, “India” has the score 0.63, “Canada” is scored 0.546 and “ms sql server” is scored 0.519, etc. Finally, his profile can be classified as a “Technology_Internet” with the score 0.948. The emergence of semantic technologies such as RDF has introduced novel models as the RDF Graph Model [7].
The semantic search structure of the Triplify [3] is a typical software architecture of such an RDB2RDF tool that just transforms all the values in the database into RDF format.
The goal of the entity extraction task is to extract entities from the given structured database that are mentioned in each document. Traditionally, these entity extraction techniques are usually not very efficient. We will have to re-investigate algorithms for popular keyword extraction in which ranking of suggestions is based on frequency in a query click log. The most critical part of popular keyword extraction algorithms is the scoring of candidate keywords. Weak scoring will cause the algorithm to get lost in the large search space.
Enhanced semantic search techniques include ranking by click models with the accuracy in relevance estimation by the well-known normalized discounted cumulative gain [17] that measures the divergence of the predicted ranking/human judgments; query recommendation (it means recommending alternative queries with closer search results to the original query); web mining for objects; and spatio-temporal analysis for geographical topic, etc.

Click models [18, 37, 42, 8, 6] making usages of query logs and click logs [23, 45, 19] can be used in many of the semantic search subjects - usually click models are understood as statistical tools using query click log data to adjust the ranking to offer for users better web search results so that users can search more accurately and more efficiently.
Starting from 2015 that click models have been researched for mobiles (cell phones) by PocketTrend system using search logs [30]. Topical (domain) information can be used to improve the relevance of retrieval results by generalizing query classification to identify other relevant results, but topical information is an indirect query classification. Hence, click models introduce a natural definition of query class that stems from the search query results and that can be estimated using click-behavior. The query click log analysis can improve directly ranking relevance. Specifically, query click logs are server-end logs that record user activities on search pages and encode user preferences of search results, typically contains timestamp, query, clicked URLs and user information. The analysis of click-through logs can help to understand the user’s latest preference tendencies. Naturally, many studies have attempted to discover user preferences from click-through logs to improve the relevance of search results. Click logs and/or query logs were also researched for Google and Yahoo search engines on cell phones [36]. Given a large query click log [26], click models provide a principled approach to inferring user-perceived relevance of web documents. There are click models as the click chain model [16] based on a solid Bayesian framework, the ImpressionRanks [5], the contextual advertising [25, 43], the DBN [10], UBM [13], BBM [27] and GCM [49]. Click models provide a principled way of integrating knowledge of user search behaviors to infer user-perceived relevance of web documents.
These click models are typically based on an assumption called the examination hypothesis. The examination hypothesis assumes that, if a document has been examined, the click-through rate of the document for a given query is a constant number whose value is determined by the relevance between the query and the document. But other users

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

usually have different intentions for searching while they still submit the same query to the search engine, it means that the examination hypothesis have to re-check to improve the search. The Unbiased-DBN and Unbiased-UBM [21] added more assumptions into the intent hypothesis of "needed" documents, they improved 2.10%-2.96% the search accuracy. Click models can be applied for image search [22] obtained a 13% improvement over the Bing engine.

III. PROPOSED METHODOLOGY

We created Java software to generate ontologies [28], entities and RDFs in XML from jobseeker profiles. The research includes 6 steps

Step 1 Entity Extraction: The entity extraction [41] component takes a text document (or a document tokenized into a sequence of tokens) as input, analyzes the document and outputs all the mentions of entities from the given structured database mentioned in the document. For each mention, it outputs the entity name, id, and the position in document it occurs at. One of the main tasks is to establish the relationships between the web documents and entities in these structured databases focusing on the "mentions relationship", that is, a web document is related to an entity if the entity occurs in the document.

Step 2 Entity Retrieval: The task of entity retrieval is to lookup the index for a given document identifier and retrieve the entities extracted from the document (at document indexing time) along with their positions and relevance. It happens at query processing time, so its overhead could be excessive. [35]

Step 3 Entity Ranker: The entity ranker ranks the set of all entities returned by the entity retrieval component. The set of highest-ranked entities are then returned along with the web document results.
For a document with an entity e inside, the entity score of e is calculated by

$$r_e = \frac{N_e - N_{min}}{N_{max} - \min} \qquad (1)$$

where N_e is the frequency (number of occurrences) of entity e within the current document, and N_{max} or N_{min} is the maximum or minimum frequency within the whole document collection of the entity e.
If we have 100 web sites including entity "John Scott", and the maximum frequency N_{max}=28 in the web site no. 8, the minimum frequency N_{min}=3 in the web site no. 99. If in the current web site, N_e=15 occurrences of entity "John Scott", then we will have $r_{john\text{-}scott}$=(15-3)/(28-3) = 12/25 = 0.480.

Step 4 Keyword extraction: keyword search over entity databases is an important problem [20]. Current techniques for keyword search on databases may often return incomplete and imprecise results. Although some returned entities contain almost of the query keywords, the intention of the keywords in the query could be different from that in the entities. So, the results could also be imprecise.

Step 5 Ranker: ranking function is suggested by Plinkr [33]

$$\text{score} = \text{avg}(r) * \text{avg}(d) * f_n \qquad (2)$$

When search for a term for example - web,
f_n: the no. of resumes that include 'web' = 200
d: default PostgreSQL search ranking for 'web', … it is 1 for the first resume, it is 1/2=0.5 for the second resume, and 1/3=0.33 for the third resume, etc.
if the 100th resume is Ha Le, then

$$d_{web}(\text{Ha Le})=1/100=0.01$$

r: relevance calculated by equation (1) , if in resume of Ha Le, there are 3 'web' terms, in PostgreSQL database, the minimum found in Sean's resume with 1 'web' to the maximum 21 web's (in Mark's resume)

Therefore the relevance of Ha Le

$$r_{web}(\text{Ha Le})= (3-1)/(21-1) = 2/20 = 0.2$$

The final score of Ha Le's resume on searching for the term 'web' is

$$\text{Score}_{web}(\text{Ha Le})=0.2*0.01*200 = 0.4$$

In case we search for more than one keywords, then we have to apply the average values. For example if we search for two keywords 'web developer', then we will have

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

$avg(r) = 0.5*(r_{web}+r_{developer})$
$avg(d) = 0.5*(d_{web}+d_{developer})$

We now have to simplify to speed-up and to reduce complexity. Because the default PostgreSQL search ranking is relatively based on text index for the frequencies of the search keywords in a document: the resume that has the maximum found 'web' terms will be ranked the first by PostgreSQL, we now replace the PostgreSQL search ranks by the frequency f; in 'Ha Le' resume, there are 3 web terms, f_{web}(Ha Le)=3.

Step 6. Combined Structured Databases Semantic Search Architecture: This is our own design; it overcomes the limitations of database search, it should return relevant search results when database search fails to return any relevant results (superiority over the database search) with the below important tasks.
We make the semantic search similarly to the semantic search structure of the Triplify, Fig. 1 illustrates our high level overview of our new search engine architecture. It has main components
(i) **Web application** (the crawler) crawls documents on the web and store them in our 16 huge databases,

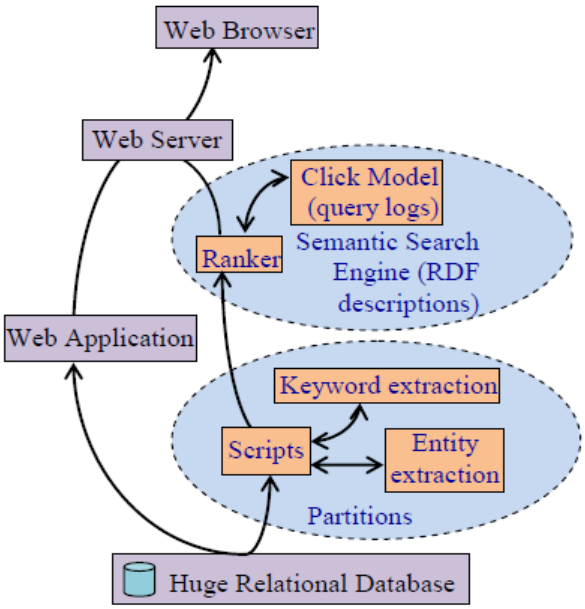

Fig. 1 Our Semantic Search Software Architecture

(ii) **Partition manager** or indexers process each document from the crawled huge databases, and build an inverted index and a document index [29]. When combing into the PostgreSQL database "silo-ed" seach [1], our federated partitions can associate semantic searches to any hashing partitions. Over a hundred entity and keyword partitions for jobseeker profiles speed up the semantic search in 16 huge network databases. The Scripts can deliver full-text keywords' hashing partitions with 2 powerful novel techniques – one that captures unlimited relevant keywords from a profile document and another that captures the top n (= 510 web search results/10 pages) similar documents in partitions upon to these keywords. The indexers using over 100 partitions can be accessed fast as well as taking advantages of over 300 well-setting triggers to maintain conveniently these entities and keywords. Our hashing partitions have scalable capability so that we are able to cover a good storage of full-text document for any enterprises' databases from more than 20 years of accumulation.
(iii) **Ranker** or the semantic search engine uses the inverted index to execute the semantic ranking for the best k document identifiers and then access the document index to obtain the URLs, resumes, cover letters and skills and generate the snippets that are shown to the user. Our semantic ranker is attached by Ontologies and RDFs.
(iv) **Front-end Web Browser** to present the search of a user.
We include novel modules than the Triplify architecture that are the Partition manager and the click model with query click logs. Machine learning by Bayesian frameworks can create for us an intelligent web search engine. Each query is enhanced by the machine intelligence and transparent to users.

In order to complete our architecture, we are training new query click logs with the structure including top-ten query ranking information, satisfaction information, click-through information and user information, etc. We do not limit the number of clicks that a user can make during a search. Our query click logs are trained with basic assumptions that: (a) a click occurs if and only if the user has examined the URL and deemed it relevant, (b) users make a linear transversal

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

through the results and decide whether to click based on the perceived relevance of the document. The user chooses to examine the next URL once unsatisfied with the clicked URL. Because the user is not necessarily satisfied with a clicked URL, we will re-check the perceived relevance and actual relevance.

IV. RESULTS

We tested our semantic search design by our Applicant Tracking System (ATS) developed on Zend Server (PhP MVC language); using PostgreSQL, we have 16 development databases of over 650,000 profile documents of resumes / cover letters / skills. There are on average 238 keywords per document. In fact, per minute there can be up to 200,000 transactions within all these databases. The timing cost of resume search is now at Table 1. On average, a search takes 4 seconds/query, the search accuracy is 88% - 91.22%.

TABLE 1 PERFORMANCE OF SEMANTIC SEARCH ON RESUME DATABASE

Keyword number per search	Timing cost
One	1 second – 10 seconds
Two	1s – 12s
Three	1s – 17s
Four	1s – 28s
Five	1s – 1 minute
Short sentence	2s – 38s
Long sentence	9s – 1m 12s
Paragraph	21s – 5m

Our semantic search included novel features as

- **Federated partitions**: we maintains huge full-text jobseeker databases; its semantic search platform can associate semantic searches to any hashing partitions. Over a hundred entity and keyword partitions for jobseeker profiles speed up the semantic search in a huge network database.
- **Scalable capability**: Our hashing partitions are able to cover a good storage of full-text document for any enterprises' databases from more than 20 accumulation years by over 300 well-setting triggers.
- **Query-cloud web module**: On generating the keywords and the entities, users are in perfect control so that they can increase or decrease the scores of all query keywords. This tends to provide users a chance to affect into the search strategy.
- **Ontology extraction**: we focus on generating OWL semantic ontologies using HTML web forms in conjunction with their associated database schema.
- **Semantic Resource-Description-Frameworks (RDF) ranking**: RDFs are applied to score jobseeker documents; we transform their raw unstructured content into semantic metadata in RDF format.
- **Intelligent web search engine**: Each query is enhanced by the machine intelligence and transparent to show to the users for complete understanding.

V. CONCLUSIONS

Our semantic search system can manage the growth of our resume database for the next 20 years. It is easy to apply in practical, the search results are fast and high accurate. We successfully developed a new semantic search architecture. Our future work aims to develop deeper click models for semantic search software on automated ranking alterations (to adjust/align the top-part of ranking with users' preference); search quality metrics which correlate with user satisfaction; and adaptive search when the meaning of a query changes over time, so do user-click patterns. A good future research can be the organization of our semantic search application onto mobile devices or cell phones.

ACKNOWLEDGMENT

Our deepest heartfelt thanks are given to Hire Ground Software Solutions company in Calgary, Alberta.

REFERENCES

[1] S. Agrawal, K. Chakrabarti, S. Chaudhuri, V. Ganti, A.C. König and D. Xin (2009). Exploiting web search engines to search structured databases. WWW'09.
[2] M. Arenas, S. Conca and J. Pérez (2012). Counting Beyond a Yottabyte, or how SPARQL 1.1 Property Paths will Prevent Adoption of the Standard. WWW'12.

[3] S. Auer, S. Dietzold, J. Lehmann, S. Hellmann and D. Aumueller (2009). Triplify – Light-Weight Linked Data Publication from Relational Databases. WWW'09.
[4] S. Banerjee, B. Bhattacharjee and C. Kommareddy (2002). Scalable Application Layer Multicast. In Proc. of ACM SIGCOMM, pp.205–217.
[5] Z. Bar-Yossef and M. Gurevich (2009). Estimating the ImpressionRank of Web Pages. WWW'09.
[6] N. Barbieri, F. Silvestri and M. Lalmas (2016). Improving Post-Click User Engagement on Native Ads via Survival Analysis. WWW'16.
[7] P. N. Bennett, K. Svore and S. T. Dumais (2010). Classification-Enhanced Ranking. WWW'10.
[8] A. Borisov, I. Markov, M.D. Rijke and P. Serdyukov (2016). A Neural Click Model for Web Search. WWW'16.
[9] S. Chakrabarti, S. Kasturi, B. Balakrishnan, G. Ramakrishnan and R. Saraf (2012). Compressed Data Structures for Annotated Web Search. WWW'12.
[10] O. Chapelle and Y. Zhang (2009). A dynamic bayesian network click model for web search ranking. WWW'09, pages 1-10.
[11] M. Ciglan, K. Nørvåg and L. Hluchy (2012). The SemSets Model for Ad-hoc Semantic List Search. WWW'12.
[12] D. Colazzo, F. Goasdoué, I. Manolescu and A. Roatis (2014). RDF Analytics: Lenses over Semantic Graphs. WWW'14.
[13] G. E. Dupret and B. Piwowarski (2008). A user browsing model to predict search engine click data from past observations. SIGIR'08, pages 331–338.
[14] L. Galárraga, C. Teflioudi, K. Hose and F.M. Suchanek (2013). AMIE: Association Rule Mining under Incomplete Evidence in Ontological Knowledge Bases. WWW'13.
[15] P. Ganesan, B. Yang and H. Garcia-Molina (2004). One torus to rule them all: Multi-dimensional queries in p2p systems. WebDB'04.
[16] F. Guo, C. Liu, A. Kannan, T. Minka, M. Taylor, Y. Wang and C. Faloutsos (2009). Click Chain Model in Web Search. WWW'09.
[17] F. Guo, C. Liu and Y. M. Wang (2009). Efficient multiple-click models in web search. WSDM'09, pages 124–131.
[18] Q. Guo and E. Agichtein (2012). Beyond Dwell Time: Estimating Document Relevance from Cursor Movements and other Post-click Searcher Behavior. WWW'12.
[19] Y. He, K. Chakrabarti, T. Cheng and T. Tylenda (2016). Automatic Discovery of Attribute Synonyms Using Query Logs and Table Corpora. WWW'16.
[20] D.M. Herzig and T. Tran (2012). Heterogeneous Web Data Search Using Relevance-based On The Fly Data Integration. WWW'12.
[21] B. Hu, Y. Zhang, W. Chen, G. Wang and Q. Yang (2011). Characterizing Search Intent Diversity into Click Models. WWW'11.
[22] V. Jain and M. Varma (2011). Learning to Re-Rank: Query-Dependent Image Re-Ranking Using Click Data. WWW'11.
[23] D. Jiang, K.W. Leung and W. Ng (2014). Fast Topic Discovery From Web Search Streams. WWW'14.
[24] M. Kaminski, E.V. Kostylev and B.C. Grau (2016). Semantics and Expressive Power of Subqueries and Aggregates in SPARQL 1.1. WWW'16.
[25] M. Karimzadehgan, W. Li, R. Zhang and J. Mao (2011). A Stochastic Learning-To-Rank Algorithm and its Application to Contextual Advertising. WWW'11.
[26] A. Korolova, K. Kenthapadi, N. Mishra and A. Ntoulas (2009). Releasing Search Queries and Clicks Privately. WWW'09.
[27] C. Liu, F. Guo and C. Faloutsos (2009). Bbm: Bayesian browsing model from petabyte-scale data. KDD'09, pages 537–546.
[28] D. Lonsdale (2003). Ontologically-based searching for jobs in linguistics. DLLS. BYU, Provo, UT.
[29] M. Mochol, H. Wache and B. Nixon, (2007). Improving the accuracy of job search with semantic techniques. BIS'07.
[30] G. Pekhimenko, D. Lymberopoulos, O. Riva, K. Strauss and D. Burger (2015). PocketTrend: Timely Identification and Delivery of Trending Search Content to Mobile Users. WWW'15.
[31] F. Qian, Z. Wang, Y. Gao, J. Huang, A. Gerber, Z.M. Mao, S. Sen and O. Spatscheck (2012). Periodic Transfers in Mobile Applications: Network-wide Origin, Impact, and Optimization. WWW'12.
[32] J.F. Sequeda, M. Arenas and D.P. Miranker (2012). On Directly Mapping Relational Databases to RDF and OWL. WWW'12.
[33] J.L. Scott (2009). Plinkr: an Application of Semantic Search. MSc thesis, supervised by Prof Dennis, S. New York University.
[34] M. Shirakawa, T. Hara & S. Nishio (2015). N-gram IDF: A Global Term Weighting Scheme Based on Information Distance. WWW'15.
[35] W. Song, S. Zhao, C. Zhang, H. Wu, H. Wang, L. Liu and H. Wang (2015). Exploiting Collective Hidden Structures in Webpage Titles for Open Domain Entity Extraction. WWW'15.
[36] Y. Song, H. Ma, H. Wang and K. Wang (2013). Exploring and Exploiting User Search Behavior on Mobile and Tablet Devices to Improve Search Relevance. WWW'13.
[37] Y. Song, X. Shi and X. Fu (2013). Evaluating and Predicting User Engagement Change with Degraded Search Relevance. WWW'13.
[38] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek and H. Balakrishnan (2001). Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. Proc. of ACM SIGCOMM, pp.149-160.
[39] C. Tang, S. Dwarkadas and Z. Xu (2004). Hybrid Global-Local Indexing for Efficient Peer-To-Peer Information Retrieval. 1st USENIX/ACM NSDI.
[40] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga, D. Gerber and P. Cimiano (2012). Template-based Question Answering over RDF Data. WWW'12.
[41] C. Wang, K. Chakrabarti, T. Cheng and S. Chaudhuri (2012). Targeted Disambiguation of Ad-hoc, Homogeneous Sets of Named Entities. WWW'12.
[42] H. Wang, C. Zhai, A. Dong and Y. Chang (2013). Content-Aware Click Modeling. WWW'13.
[43] X. Wang, A. Broder, M. Fontoura and V. Josifovski (2009). A Search-based Method for Forecasting Ad Impression in Contextual Advertising. WWW'09.
[44] Z. Wang, F.X. Lin, L. Zhong and M. Chishtie (2012). How Far Can Client-Only Solutions Go for Mobile Browser Speed? WWW'12.
[45] S. Whiting and J.M. Jose (2014). Recent and Robust Query Auto-Completion. WWW'14.
[46] M. Wylot, P. Cudré-Mauroux and P. Groth (2014). TripleProv: Efficient Processing of Lineage Queries in a Native RDF Store. WWW'14.
[47] M. Zhong, J. Moore, K. Shen and A.L. Murphy (2005). An Evaluation and Comparison of Current Peer-to-Peer Full-Text Keyword Search Techniques. Webdb'05.
[48] Y. Zhou, B.C. Grau and I. Horrocks (2013). Making the Most of your Triple Store: Query Answering in OWL 2 Using an RL Reasoner. WWW'13.
[49] Z. A. Zhu, W. Chen, T. Minka, C. Zhu and Z. Chen (2010). A novel click model and its applications to online advertising. WSDM'10, pages 321–330.

BIOGRAPHIES

Le Quan Ha (MSc First Laureate, PhD) is a Research Assistant of Queen's University Belfast in the United Kingdom and graduated his PhD in Queen's University Belfast, United Kingdom in 2005.

Mainur Rahman (MSc) is a Teaching Assistant of University of Calgary in Alberta, Canada and graduated his MSc of Computer Science in University of Calgary, Alberta, Canada in 2010.