A Study on Keyword Search over Structured and Semi-structured Data Streams (構造データおよび半構造データストリームに対するキーワード検索に関する研究)

Savong Bou[♡]

With the rapid growth of real-time information from various sources, continuous keyword search over streams has become increasingly important. In this dissertation, we propose scalable and efficient ways to enable keyword search over XML and relational streams by addressing the following major problems: 1) quality of search results in keyword search over XML streams and 2) scalability against the number of query keywords and the result size in keyword search over relational streams. The experimental results prove that the proposed methods perform much better than the comparative approaches.

1. Introduction

With the current trends of Cyber Physical Systems [5], Internet of Things [4], etc., the number of real-time information sources has been explosively increasing. Data streams are of various forms ranging from pure texts to structured data streams. XML and relational data are two of the most popular data-representations that have been extensively used for the last decade. Therefore, retrieving information from XML and relational streams is quite important and useful for many applications, such as social activity analysis over social streams, stock market analysis, disaster detection over sensor streams, intelligent supplies management, useful information retrieval from social networks, etc.

To retrieve information from streams, standard query languages, like CQL, XPath, etc., are commonly used. However, these query languages are not suitable for naive users because users have to know the specification of query language and the detail about the schema of streams. Compared to such conventional query languages, keyword search is considered to be a better solution due to its simplicity and its user-friendliness. Therefore, a great deal of efforts have been put on this search paradigm recently.

This dissertation presents two approaches to improve the performance of keyword search over XML streams or relational streams by addressing the following problems:

♡ Student Member Graduate School of Systems and Information Engineering, University of Tsukuba
savong.bou@kde.cs.tsukuba.ac.jp

Keyword search over XML streams It is worth to mention that, in many application scenarios, there is a strong need to make keyword search against some specific parts of XML data whose structures constantly change and are unknown or little known. Taking a bibliographic XML data for example, one may want to retrieve the abstract containing some keyword. However, since query keywords in keyword search can appear either as label (XML element) or textual value, and can carry multiple and different meanings, it is hard to express the exact search intention only with keywords. In particular, it is hard to specify which parts of XML data to which the keyword search should be applied. This problem becomes worse when the XML data is rich in textual contents.

The above problem can be solved by combining XPath-based query with keyword search. The XPath-based query will be used as a mean to specify which part of XML data the keyword search should be applied to, and the keywords are used to specify the user's demand for the query results. It should be noticed that the combination of XPath- and keyword-based queries is beneficial to the users, because they can exploit the benefits from both query styles, that is, one does not need to fully understand the structure of the documents being queried, while having the freedom to limit the parts of the documents to be retrieved in terms of XPath expression.

To the best of authors' knowledge, no research has been made on this type of query in a streaming setting so far. To address this problem, we propose a scheme to process XPath queries combined with keyword search over XML streams. More precisely, we extend NFA model to support XPath-based keyword search. We also extend NFA-based YFilter [3] with the method used in CKStream [8]. The XML data whose structure is simple and understandable, while rich in textual contents are focused on because such data are not in consideration of many XML keyword search works.

Keyword search over relational streams It should be noted that the performance of the comparative approaches [9, 10] considerably degrades when the number of query keywords and/or network size (T_{max}) are increased. The increase of these two parameters causes rapid increase in the number of CNs, which results in a lot of common partial networks remain unintegrated. To exemplify the problem, let us take TPC-H dataset [1] as an example. When the number of keywords and T_{max} are increased from four to five, the number of CNs increases from 3,600 to 85,803 [10]. Likewise, the total number of edges in the query execution plans exponentially increases from 4,276 to 73,596 in S-KWS [9] and from 7,486 to 222,040 in SS-KWS [10]. Thus the performance of S-KWS and SS-KWS would deteriorate in particular when dealing with a lot of query keywords and/or large relational streams consisting of many relations. As reported in [6], the average query length to the search engines has been increasing. For example, the ratio of queries containing more than five words has increased by 10% over the years, while that of single keyword queries has decreased by 3%.

How can we cope with such exponential blowup of CNs and the complication of query plans? If we consider the edges in CNs, each of them can be associated to one of the primary/foreign-key relationship between two tables, whose number is in general small. In other words, we can consider that the edges in CNs are intensively duplicated from the primary/foreign-key relationships in the schema. With the same example above when the number of keywords and T_{max} are increased from four to five, the total number of unique edges in all CNs grows linearly from 1,088 to 3,536. Under this observation, to cope with the problem of CN's exponential blowup, it is possible to consolidate the edges sharing the same primary/foreign-key relationship into one edge when generating a query plan, which leads to great performance improvement. Our algorithm takes into account the above idea. Specifically, a new query plan, called Maximal Sharing structure (*MX-structure*), is proposed to consolidate common edges in different CNs as much as possible.

We evaluate both proposed schemes by extensive experiments on both synthetic and real datasets. The results show that the proposed schemes work well with acceptable throughputs, less memory usage, and good efficiency and utility.

2. XPath-based keyword search over XML streams

Keyword search over XML streams is a searching technique where the inputs are a set of queries that contain keywords and XML streams, and the output is XML sub-trees that contain all keywords. Specifically, the input queries are evaluated against XML streams where XML elements and their textual values continuously arrive. During the entire filtering, all sub-trees of XML streams are kept, and when any sub-tree is detected as containing all keywords of any query, such sub-tree is returned as the query's result.

The main problem of pure keyword search over XML streams is how to efficiently return only the results that are really needed by the users. The existing algorithms of keyword search over XML streams return all results to the users even though they are not what users want. This problem happens because the existing algorithms do not understand the real search intentions of users through queries that consist of sets of pure keywords. Specifying real search intension with just pure keyword search is difficult because keywords can appear in any parts of data streams and can carry multiple meanings. This problem can be partly solved by adopting the ranking mechanisms [7] that have been extensively studied for keyword search over static data. However, such rankings are sometimes not effective because they exclusively use the whole static data and query (a set of pure keywords) to rank the results without taking into account what the users really want to get. To the best of our knowledge, existing algorithms of keyword search over XML streams do not adopt such ranking mechanisms. Nevertheless, we believe that such ranking mechanisms can be adopted to streams' setting, though they may become less effective and involve in very heavy computational cost due to the unavailability of whole data streams at the time of ranking. That will put additional burden to the already poor performance in terms of filtering speed as explained above.

To solve this problem, in stead of adopting the ranking mechanisms of static XML, which is sometimes not efficient and too costly for XML streams, we propose a user-friendly query that allows users to easily and effectively define their real search intentions. For this purpose, we propose XPath-based keyword search that combines XPath with keywords. Specifically, XPath- part is used to specify which part of XML streams that users want to search, and keywords- part is used to define search intention for the query results. Figure 1 shows an example of processing keyword search and XPath-based keyword search for the same search intention. As can be seen, unrelated results are also returned if using keyword search, but only related result is returned by the proposed XPath-based keyword search.

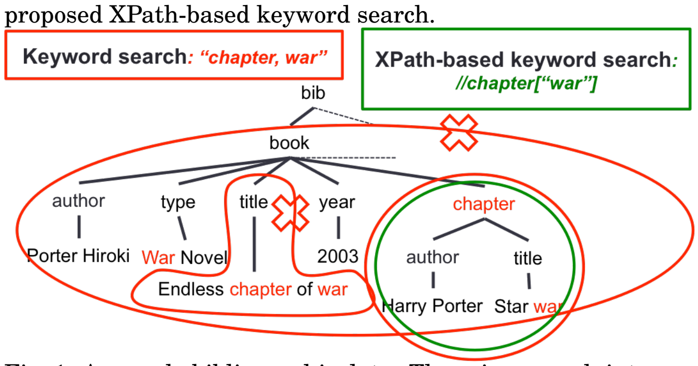
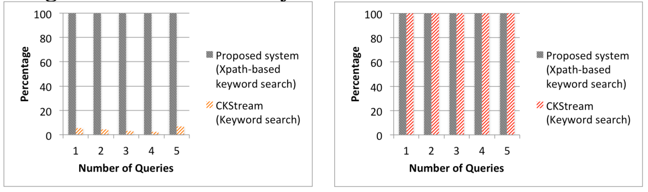


Fig. 1: A sample bibliographic data. There is a search intention (find "chapter" that is about "war") with a corresponding translated-keyword-search in red box, and XPath-based keyword search in blue box. The answers of keyword search are sub-trees in red circle, and that of XPath-based keyword search is sub-tree in blue. Sub-trees in red with crossed mark are unrelated results.

0.1 Experimental evaluation

We evaluate the effectiveness of our proposed method and CKStream [8] when querying for the same search intentions shown in Table 1. The translated keyword searches are processed in CKStream, and the translated XPath-based keyword searches are processed in our proposed method. These queries are chosen randomly from the used XML data generated by XMark [2]. The relevant matches of the search intentions shown in Table 1 are predetermined (e.g., subtrees rooted at elements "*shipping*"), which contain the respective keywords. For the experimental purpose, we modify XML data generated by XMark by adding several keywords into various textual elements randomly.

We evaluated precision and recall of our proposed method and CKStream. As shown in Figure 2(a), CKStream has very low precision on all queries because it returns many unrelated results (any subtrees that contain all keywords). Whereas, our proposed method has high precision (100%) because it only returns the relevant matches (e.g., the subtrees rooted at elements "*shipping*"). This proves the effectiveness of the application of the part "/XPath-" of our proposed method, which can greatly help reduce the vagueness of keyword search. Both methods have high recall (100%) as shown in Figure 2(b) because they both return all desired results.



(a) Precision. (b) Recall.

Fig. 2: Precision and recall.

Table 1: One of the five search intentions and all corresponding queries.

Search intentions	Keyword search	XPath-based keyword search
Find the "*shipping*" with "*fixed*" "*pays*"	*shipping fixed pays*	//shipping[ftcontains(fixed pays)]

2. Keyword Search over Relational Streams by Aggressive Candidate Network Consolidation

Keyword search over relational streams is to find the sets of connected tuples that contain all query's keywords, which is typically called MTJNT(s) [7], as tuples continuously arrive in real-time. An example is shown in Figure 3.

The process of keyword search on relational streams comprises two main steps: *preprocessing* and *filtering* steps.

- ***Preprocessing step***: Given a schema, a set of query keywords, and T_{max}, all *Candidate Networks* (CNs) [9,10] are generated. A CN is a tree, where 1) each node represents a relation and 2) each edge represents a relational *join* operation. Notice that all CNs must conform to the concept of MTJNT [9]. Figure 4 shows an example of all CNs generated from keyword search "k_1, k_2" and the schema in Figure 3(a). Then a query plan is generated from all CNs.
- ***Filtering step***: In this step, the query plan is evaluated over relational streams. When new MTJNTs are detected due to arrivals of new tuples, they are reported. On the other hand, expired tuples are removed by using either eager or lazy approaches [9].

The existing approaches for this search framework create query's plans by integrating all CNs. However, they suffer from the very poor performance when processing longer queries. The reason is that when processing queries that have more number of keywords, the number of partial results (candidate results), which need to be kept and instantly and independently evaluated with the future incoming streams, are exponentially increased. Same problem happens when the maximum size of the search results are set to be big (T_{max}). These two parameters (long query and big T_{max}) are very important for real search scenario. For example, to process keyword search "k1, k2" over relational streams in Figure 3(b), it is required to keep a lot of partial results as shown in Figure 3(c).

As part of the solutions to this problem, we propose an efficient query plan, called *MX-structure* (maximal-sharing structure), that can handle all candidate results more effectively against the future incoming relational streams. The basic idea of the proposed approach is to keep all candidate results together according the query plan in such a way that evaluating them against future incoming relational streams can be done as minimal as possible, which means the number of candidate results that can share processing is maximal. Therefore, longer queries can be handled well. Figure 5 is an example of the proposed MX-structure that is constructed by consolidating all common edges. Notice that, labels (colors) on each edge represent the set of CNs' IDs that it belongs to. Based on the proposed approach, to process keyword search "k1, k2" over relational streams in Figure 3(b), all partial results are kept together by tracking matched results using labels according to MX-structure as shown in Figure 6.

Table 2: Parameters used in the experiments.

Parameter	Range and default
Window size (mn)	10, 20, **30**, 40, 50
Keyword frequency (%)	0.003, **0.007**, 0.01, 0.013
# of keywords	2, **3**, 4, 5
T_{max}	2, 3, **4**, 5

0.2 Experimental evaluation

We compared the performance in terms of CPU running time and memory usage of the proposed approach with those of full mesh (FM) and partial mesh (PM) of S-KWS, and SS-KWS. We used both synthetic and real datasets. Due to lack of space, we only reported the result of synthetic dataset, TPC-H [1], which deals with ad-hoc decision support system in business environment. Notice that this dataset is specially prepared to favor SS-KWS to S-KWS. Parameters used in the experiments are shown in Table 2. The default parameters are written in bold.

First, we measured the CPU running time and the memory usage when varying the number of keywords (Figures 7(a) and 7(b), respectively). As can be seen, CPU running time and the memory usage in FM/PM and SS-KWS were increased exponentially, whereas the proposed scheme was not. This is because the existing approaches keep a lot of partial results independently for evaluating with future incoming streams, which is not efficient. Whereas the proposed approach keep all partial results together and let them share processing when evaluating against future incoming streams. Similar tendency can be observed when varying T_{max} from two to five (Figure 8).

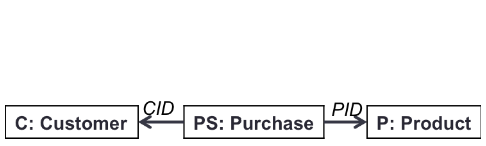
7. Conclusion

In this dissertation we have addressed the problem of keyword search over XML and relational streams. The XPath-based keyword search has proposed to address the problem of vagueness of pure keyword search over XML streams by improving both accuracy and searching performance. In addition, the problem of poor performance of keyword search over relational streams has also been addressed by proposing the efficient query plan, MX-structure, and an efficient mechanism to process MX-structure over relational streams. Performance improvement of both proposals has been proved by extensive experiments on both real and synthetic datasets.

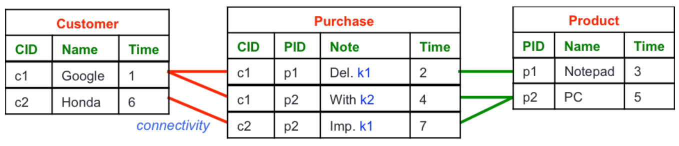
[Bibliography]

[1] Tpc-h benchmark dataset. *http://www.tpc.org/tpch/*, 2015.

[2] R. Busse, M. Carey, D. Florescu, M. Kersten, I. Manolescu, A. Schmidt, and F. Wass. XMark-an XML benchmark project. 2013.



(a) Schema.

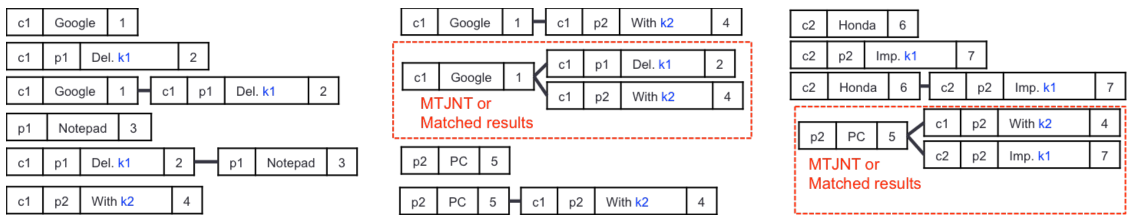

Customer
Purchase
Product
connectivity

Customer		
CID	Name	Time
c1	Google	1
c2	Honda	6

Purchase			
CID	PID	Note	Time
c1	p1	Del. k1	2
c1	p2	With k2	4
c2	p2	Imp. k1	7

Product		
PID	Name	Time
p1	Notepad	3
p2	PC	5

(b) Instances.



(c) Example of MTJNT.

Fig. 3: An example of keyword search "k1, k2" on relational streams in Figure 3(b).

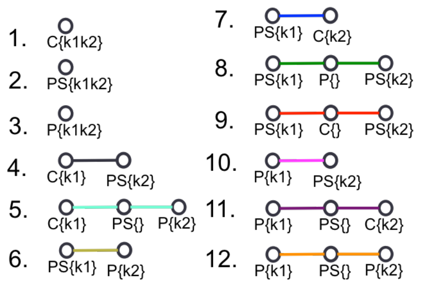


Fig. 4: All CNs created from schema in Figure 3(a) for query "k_1, k_2". Notice that the label under each node is a tuple set, and "*C*" is referred to table "*Customer*", "*PS*" is referred to table "*Purchase*", and "*P*" is referred to table "*Product*". The keyword inside the curly bracket is referred to the keyword of the given query that each node contains. Notice that, for this example, T_{max} is set to 3.

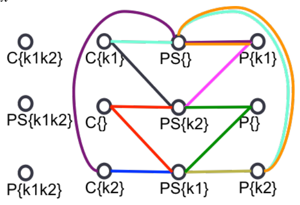


Fig. 5: MX-structure for all CNs in Figure 4

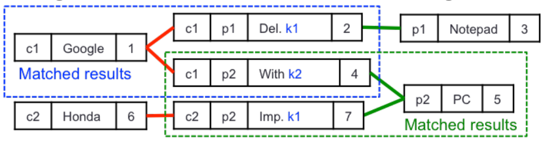
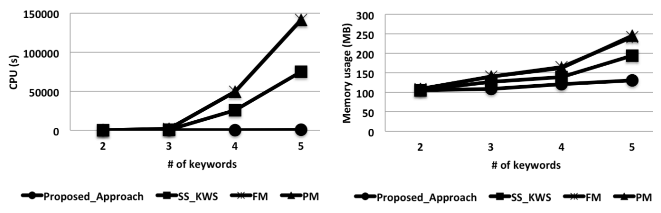
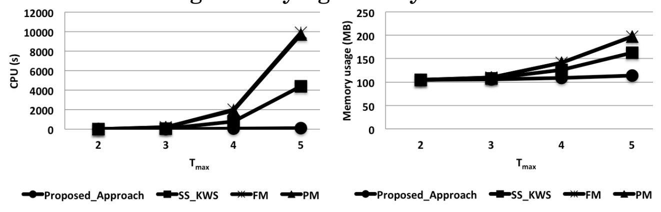


Fig. 6: All partial results using the proposed approach.



(a) CPU running times. (b) Memory usage.

Fig. 7: Varying # of keywords.



(a) CPU running times. (b) Memory usage.

Fig. 8: Varying T_{max}.

[3] Y. Diao and M. J. Franklin. High-performance XML filtering: An overview of YFilter. In *IEEE*, pages 41–48, 2003.

[4] M. Dyk, A. Najgebauer, and D. Pierzchala. Agent-based ms of smart sensors for knowledge acquisition inside the internet of things and sensor networks. *ACIIDS*, 9012:224–234, 2015.

[5] L. Edward. Cyber physical systems: Design challenges. *University of California, Berkeley Technical Report No. UCB/EECS-2008-8. Retrieved 2008-06-07*, 2008.

[6] K. Hogan. Interpreting hitwise statistics on longer queries. *Technical report, Ask.com*, 2009.

[7] V. Hristidis and Y. Papakonstantinou. Discover: Keyword search in relational databases. In *VLDB*, Hong Kong, China, 2002.

[8] F. Hummel, A. Silva, M. Moro, and A. Laender. Multiple keyword-based queries over xml streams. In *CIKM*, Glasgow, Scotland, UK, 2011.

[9] A. Markowetz, Y. Yang, and D. Papadias. Keyword search on relational data streams. In *SIGMOD*, Beijing, China, 2007.

[10] L. Qin, J. Xu Yu, and L. Chang. Scalable keyword search on large data streams. In *VLDB Journal*, 2011.

Savong BOU

is currently a postdoctral researcher at the Center for Computational Science, University of Tsukuba. He received his B.Ed., from Royal University of Phnom Penh (RUPP, Cambodia), and B.Sc. from Norton University (Cambodia) in 2009. He received M.Eng., and Ph.D. degree in Engineering from University of Tsukuba, Japan, in 2014, and 2017, respectively. His interest includes data Integration, mining of data streaming, information retrieval, big data analysis.