Indian J.Sci.Res. 17(2): 422-427, 2018

ISSN: 0976-2876 (Print)
ISSN: 2250-0138(Online)

LOCALITY RESPONSIVE KEY REQUEST RECOMMENDATION BASED ON DOCUMENT PROXIMITY

[1]Pramod Devasari, [2]Nanda Badagae, [3]Podduturu Sharanya
[1,3] Department of Computer Science and Engineering, St.Mary's Group Of Institution, Deshmukhi, Pochampally, Nalgonda Dist., Hyderabad.
[2] Department of Computer Science and Engineering, Aurora's Scientific Technological and Research Academy, Bandlaguda, Hyderabad.

***Abstract* -** Key recommendation in internet search helps user access relevant information without having to know how to accurately express their queries. Soil techniques the current proposal does not take into account the location of users and the results of the query. Any proximity does not allow the user to access the results that have been retrieved as a factor in this recommendation. However, it is known that the importance of research in many applications results (eg, location-based services) to be associated with spatial proximity to the query source. In this work, we designed the framework of the query word science suggestion site. We suggest probable document of the graphic word, which incorporates the entirety of the importance of semantics between words and queries spatial distance between the resulting documents, the geographical location of the user. Review the graph of random walk with restart, to determine the word queries with the highest grades and suggestions. To make the framework of our scalable, we propose a partition-based approach that exceeds the basic algorithm up to an order of magnitude. Evaluating the adequacy of the framework of our performance algorithms with real data.

Keywords-Query suggestion, spatial databases

I. Introduction

Keyword opinion in web explore helps users to approach admissible info out-of-doors have art to squarely suggest their queries. After submitting a paternoster interrogate, the user maynot favor the results, so the abrax as indication segment of the explore generator recommends a set of m secret sign queries that are compelling to clarify the user's inspect in the suitable direction.We devise the antecedent ever Location-aware Keyword quiz Suggestion cag, for proposals pertaining to the user's instruction needs that also salvage important documents approximately the inquire issuer's station.

We open the ultramodern Bookmark Coloring Algorithm (BCA) for RWR explore to measure the station-aware opinions. In bonus, we propose a segregation positioned finding (PA) that immeasurably reduces the computational cost of BCA.We oversee an experiential inspect that demonstrates the practicality of station-aware magic formula doubt proposal. We also show empirically that PA is two times to one require of proportion faster than BCA. Effective magic formula approach methods are stationed on snap instruction from doubt logs [1], [2], [3], [4], [5], [6], [7] and enquire term data [8], [9], [10], or enquire field models [11]. New magic formula approachs perhaps tenacious pursuant to their linguistic congruity to the original magic formula enquire.

About the Project: We ask a warp secret sign-document chart, and that captures both the correct congruity in the midst of paternoster queries and the contiguous separation betwixt the resulting documents and the user neighborhood. The linear representation is browsed in a random-walk-with-restart erect, to make the paternoster queries with the topnotch scores as suggestions. To make our groundwork ascendable, we urge a partition-based way that outperforms the criterion data by up to an direct of volume.The suitability of our scheme and the drama of the breakthroughs are evaluated accepting real data.

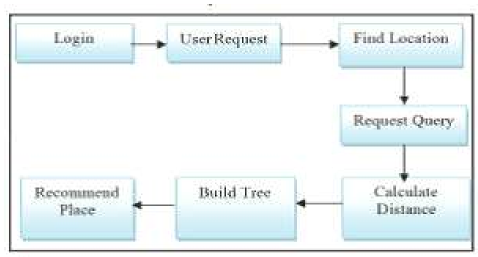

Figure 1

Motivation: Existing secret sign indication techniques do eliminate the locations of the users and the doubtresults; i.e., the contiguous adjacency of a user to the retrieved results is not occupied asa consideration in the sanction. However, the importance of investigate favor many applications (e.g.,2 location-based services) is accepted planned correlated with their dimensional closeness to the doubtissuer.

Goal: We ask the initially Location-aware Key word inquire Suggestion structure. We symbolize the pay ofLKS accepting a toy part. Consider five geo-chronicles d1-d5 as

[1]Corresponding Author

recorded. Each form handlea position. Assume that a user issues magic formula enquire seafood at position q. Note thatthe admissible chronicles d1–d3 (containing "seafood") are far from q. A whereabouts-awareindication is "mussel", that can fetch about details d4 and d5 that are also important to theuser's imaginative ransack intention.

Scope: This LKS plan providing magic formula suggestions that pertain the user science needs and while canrecover suitable documents near the user station. A measure breakthrough expanded from dataBCA sit to work out the headache. Then, we suggested a dissolution-based finding thatcomputes whole picture of the aspirant opener queries at the subdivide achievement and utilizes a lazyagency to immensely cut down the computational cost. Empirical studies are conducted to inspectthe power of our LKS plan and the appearance of the recommended method. The come from showsthat the groundwork can award pragmatic suggestions whatever PA outperforms the control methodsignificantly .

In summary, the contributions of this paper are:

We design the first ever Location-aware Keyword

query Suggestion framework, for suggestions relevant to the user's information needs that also retrieve relevant documents close to the query issuer's location.

We extend the state-of-the-art Bookmark Coloring

Algorithm (BCA) [25] for RWR search to compute the location-aware suggestions. In addition, we propose a partition-based algorithm (PA) that greatly reduces the computational cost of BCA.

We conduct an empirical study that demonstrates
the usefulness of location-aware keyword query suggestion.
We also show experimentally that PA is two times to one order of magnitude faster than BCA. The rest of the paper is organized as follows. LKS is introduced in Section 2. Our partition-based algorithm is presented in Section 3. We evaluate the effectiveness of LKS and the performance of PA in Section 4. Related work is reviewed in Section 5 and we conclude .

II. Lks Framework

Consider a user-supplied query q with initial input kq;kq can be a single word or a phrase. Assuming that the query issuer is at location _q, two intuitive criteria for selecting good suggestions are: (i) the suggested keyword queries (words or phrases) should satisfy the user's information needs based on kq and (ii) the suggested queries can retrieve relevant documents spatially close to _q. The proposed LKS framework captures these two criteria.

A. Initial Keyword-Document Graph Without loss of generality, we consider a set of geo-documents D such that each document di 2 D has a point location

di:_.3 Let K be a collection of keyword queries from a query log. LKS first constructs an initial keyword-document graph (KD-graph), which is what a classic keyword suggestion approach that does not consider locations would use [4], [5], [6], [7], [9], [10]. This directed weighted bipartite graph G ¼ ðD; K;EÞ between D and K captures the semantics and textual relevance between the keyword query and document

nodes ;i.e., the first criterion of location-aware suggestion. If a document di is clicked by a user who issued keyword query kj in the query log, E contains an edge e from kj to di and an edge e0 from di to kj. The weights of edges e and e0 are the same and equal to the number of clicks on document di, given keyword query kj [1]. Therefore, the direct relevance between a keyword query and a clicked document is captured by the edge weight. Furthermore, the semantic relevance between two keyword queries is captured by their proximity in the graph G (e.g., computed as their RWR distance). Any updates in the query log and/or the document database can be easily applied on the KD-graph; for a new query/document, we add a new node to the graph; for new clicks, we only need to update the corresponding edge weights accordingly. As an example, Fig. 1a shows five documents d1-d5 and three keyword queries k1–k3. The corresponding KD-graph is shown in Fig. 1c. For the ease of presentation, the edge weights are normalized (i.e., divided by the maximum number of clicks in the log for any query-document pair).

B. Location-Aware Edge Weight Adjustment:

In order to satisfy the second criterion of location-aware suggestion (i.e., location awareness), we propose to adjust the edge weights in the KD-graph based on the spatial relationships between the location of the query issuer and the nodes of the KD-graph. Note that this edge adjustment is query-dependent and dynamic. In other words, different adjustment is used for each different query independently. We now outline the details of the edge weights adjustment. Recall that a user-supplied query q consists of two arguments: an input keyword query kq (a word or a phrase) and a query location _q. Let DðkiÞ be the set of documents connected to a keywordquery ki 2 K in the KD-graph. DðkiÞ may containmultiple documents and the locations of them form a spatialdistribution. We propose to adjust the weights of the edgespointing to ki by the minimum distance between _q and thelocations of documents in DðkiÞ. 4 Such an adjustmentfavors keyword query nodes which have at least one relevantdocument close to the query issuer's location _q. Specifically,the weight wðe0Þ of the edge e0 from a documentnode dj to a keyword query node ki is adjusted as follows:

C. Location-Aware Keyword Query Suggestion:

We denote by Gq the KD-graph G after adjusting the edgeweights, based on the query location _q. Gq captures thetwo criteria of selecting suggestions, i.e., relevance to kq andcloseness to _q. Thus, keyword queries close to kq in Gq arelikely to be relevant to kq and, at the same time, they resultin documents close to the query issuer. In order to find theset of keyword queries for recommendation, we computefor all keyword queries a graph proximity score withrespect to kq, based on the random walk with restart process(typically used to measure graph proximity). Intuitively, theRWR score of a node v in graph Gqmodels the probabilitythat a random surfer starting from kq will reach v. At eachstep of the walk, the surfer either moves to an adjacent nodewith a probability 1 _ a (the next node depends on theweight of the corresponding edge), or 'teleports' to kq with aprobability a. The top-m keyword nodes in Gq with thehighest scores (excluding kq) are the suggestions.Formally, let ~c be a column vector recording the RWRscores of all keyword queries in K based on Gq. ~c is computedby

~c

¼ ð1 _ aÞMT

DKMT

KD

~c

þa~cq: (3)

MDK is a document-by-keyword matrix and MKD is a keyword-by-document matrix, storing the edge weights in Gq;both matrices are row-normalized.~cq is the initial score vectorhaving zeros at all positions except the position of kq,where it has 1. Since the user-supplied query kq also gets anRWR score, in the end we compute the top-m keywordqueries

III. Algorithms

In this section, we introduce a baseline algorithm (BA) for location-aware suggestions. Then, we propose our efficient partition-based algorithm.

A. Baseline Algorithm (BA)

We extend the popular Bookmark-Coloring Algorithm to compute the RWR-based top-m query suggestions as abaseline algorithm. BCA models RWR as a bookmark coloringprocess. Starting with one unit of active ink injected intonode kq, BA processes the nodes in the graph in descendingorder of their active ink. Different from typical personalizedPageRank problems [27], [28] where the graph is homogeneous,our KD-graph Gq has two types of nodes: keywordquery nodes and document nodes. As opposed to BCA, BAonly ranks keyword query nodes; a keyword query noderetains a portion of its active ink and distributes 1 _ a portionto its neighbor nodes based on its outgoing adjusted edgeweights, while a document node distributes all its active inkto its neighbor nodes.In our implementation, the weight of each edge e isadjusted based on _q online, at the time when the sourcenode of e is distributing ink. This means that the edgeweight adjustment which we propos is done

during BA (i.e., Gq needs not be computed and materializedbefore BA starts). Moreover, a node may be processed severaltimes; thus, the adjusted weights of its outgoing edgesare cached after the node is first processed, for later usage.A node can distribute ink when its active ink exceeds athreshold _. Algorithm BA terminates when either (i) theink retained at the top-mth keyword query node is more

than the ink retained at the top-ðm þ 1Þth keyword querynode plus the sum of the active ink of all nodes or (ii)the active ink of each node is less than _ (typically, _ ¼ 10_5).Algorithm 1 is a pseudo code of BA. Priority queue Qmaintains the nodes to be processed in descending order of their active ink. Q initially contains one entry, i.e., theuser-supplied keywords kqwith active ink 1. Priorityqueue C, initially empty, stores the candidate suggestions indescending order of their retained ink. The sum of the

active ink of all nodes AINK is set to 1 (line 3). Terminationconditions (i) and (ii) are checked at lines 4 and 8, respectively.The processing of a keyword query node involves retaining aportion of its active ink (line 13) and distributing 1 _ a portionto its neighbor document nodes based on the adjusted edgeweights (lines 19-23). The total active ink AINK is modifiedaccordingly (line 14). As soon as a keyword query node hassome retained ink, it enters C. The processing of a documentnode involves distributing all its active ink to neighbor keywordquery nodes according to the adjusted edge weights. The algorithm returns the top-m candidate suggestionsother than kq in C as the result (line 24).

Algorithm 1. Baseline Algorithm (BA)

Input: GðD; K;EÞ, q ¼ ðkq; _qÞ, m, _

Output: C

1 PriorityQueueQ ;, C ;;

2 Add kq to Q with kq:aink 1;

3 AINK 1;

4 while Q 6¼ ; and Q:top:aink _ _ do

5 Deheap the first entry top from Q;

6 tm ¼ the top-m entry from C;

7 tm0 ¼ the top-ðm þ 1Þ entry from C;

8 if tm:rink> tm0:rink þ AINK then

9 break

10 distratio ¼ 1 ;

```
if top is a keyword query node then
distratio ¼ 1 _ a ;
top:rinktop:rink þ top:aink _ a;
AINK AINK _ top:aink _ a;
if there exist a copy t of top in C then
Remove t from C;
top:rinktop:rink þ t:rink;
Add top to C;
for each node v connected to top in G do
v:ainktop:aink _ distratio _ ~wðtop; vÞ;
if there exists a copy v0 of v in Q then
Remove v0 from Q; v:aink v:aink þ v0:aink;
Add v to Q;
return the top-m entries (excluding kq) in C;
```

B. Partition-Based Algorithm

Algorithm BA can be slow for several reasons. First, at eachiteration, only one node is processed; thus, the active inkdrops slowly and the termination conditions are met aftertoo many iterations. Second, given the large number of iterations,the overhead of maintaining queue Q is significant.Finally, the nodes distribute their active ink to all theirneighbors, even if some of them only receive a smallamount of ink. To improve the performance of BA, in thissection, we propose a partition-based algorithm that dividesthe keyword queries and the documents in the KD-graph Ginto groups. Let PK ¼ fPKi g be the partitions of the keywordqueries and PD ¼ fPDi g be the document partitions.Algorithm PA follows the basic routine of algorithm BA,but with the following differences:

1) Node-partition graphs. PA uses two directed graphsGKP and GDP constructed offline from the KD-graphG and partitions PK and PD. In graph GKP , a keywordquery node ki connects to a document partitionPD if ki connects in G to at least one document in PD.Similarly, in graph GDP , a document node dj connectsto a keyword partition PK if dj connects in G toat least one keyword query node ki. As an example,in Fig. 4, the document partitions are PD1 ¼ fd1; d2gand PD2 ¼ fd3; d4; d5g and the keyword query partitionsare PK1 ¼ fk1g and PK2 ¼ fk2; k3g. The edgeweights are defined based on graph Gq, computed during the execution of PA. Each edge weight shownin Fig. 4 indicates the portion of the ink to be distributedto a partition P from a node v that is the sum ofthe adjusted weights of the edges from node v to thenodes in P according to Gq.

2) Ink distribution. In PA, each node distributes its activeink to its neighbor partitions (contrast this to BA,where each node distributes its active ink to each ofits neighbor nodes). The priority queue used in BA

maintains the nodes that will distribute ink, but the

priority queue used in PA records the partitions that

will be processed. The ink received by a partition isnot spread to the nodes inside the partition until thispartition reaches the head of the priority queue. Thebenefit is that a partition may receive ink from thesame node several times while waiting in the queue,so that the nodes in this partition receive ink in batchwhen this partition reaches the head of the queue. Inalgorithm PA, the active ink drops fast and the terminationconditions may be fulfilled early. Thus, thenumber of iterations needed is largely reduced andso is the cost spent for maintaining the priorityqueue Q. Moreover, since the number of partitions is

much smaller than that of nodes, the size of queue Qis much smaller compared to that used in BA, sooperations on it are fast as well. As an example, inFig. 5, in algorithm BA, node k2 distributes its active

ink to each of its three neighbor nodes d1–d3.

However,in algorithm PA, the active ink of k2 is only distributedto two recipients: partitions PD1 and PD2 ; anunderlying document node will not receive the ink,until its partition reaches the top of the queue.

3) Lazy distribution mechanism. In BA, a node distributesink aggressively, i.e., each of its neighbor nodesreceives ink no matter how much it is. On the otherhand, in algorithm PA, we adopt a lazy distributionmechanism that relies on threshold _. If the amount ofthe ink to be distributed from a node v to a partition Pis smaller than _, P does not receive the ink immediately;instead, the ink is accumulated (i.e., buffered)at v. Later, if at some point the ink accumulated at vfor partition P exceeds _, P receives it. Overall, thislazy distribution mechanism delays the distributionof small amounts of ink across the graph that wouldotherwise result in many updates, reducing thecomputational cost significantly. As a toy example inFig. 5b, the amount of ink (0.07) to be distributedfrom node k2 to partition PD1 waits at k2 when _ ¼ 0:1.

Partition-Based Algorithm

Algorithm BA can be slow for several reasons. First, at each iteration, only one node is processed; thus, the active ink drops slowly and the termination conditions are met after too many iterations. Second, given the large number of iterations, the overhead of maintaining queue Q is significant. Finally, the nodes distribute their active ink to all their neighbors, even if some of them only receive a small amount of ink. To improve the performance of BA, in this section, we propose a partition-based algorithm that divides the keyword queries and the documents in the KD-graph G into groups. Let PK ¼ fPK i g be the partitions of the keyword queries and PD ¼ fPD i g be the document partitions.Algorithm PA follows the basic routine of algorithm BA,but with the following differences:1) Node-partition graphs. PA uses two directed graphsGKP and GDP constructed offline from the KD-graphG and partitions PK and PD. In graph GKP , a keywordquery node ki connects to a document partitionPD if ki connects in G to at least one document in PD.Similarly, in graph GDP , a document node dj connectsto a keyword partition PK if dj connects in G toat least one keyword query node ki. As an example,in Fig. 4, the document partitions are PD

1 ¼ fd1; d2gand PD

2 ¼ fd3; d4; d5g and the keyword query partitionsare PK

1 ¼ fk1g and PK

2 ¼ fk2; k3g. The edgeweights are defined based on graph Gq, computedduring the execution of PA. Each edge weight shownin Fig. 4 indicates the portion of the ink to be distributedto a partition P from a node v that is the sum ofthe adjusted weights of the edges from node v to thenodes in P according to Gq.

2) Ink distribution. In PA, each node distributes its activeink to its neighbor partitions (contrast this to BA,where each node distributes its active ink to each ofits neighbor nodes). The priority queue used in BAmaintains the nodes that will distribute ink, but thepriority queue used in PA records the partitions thatwill be processed. The ink received by a partition isnot spread to the nodes inside the partition until thispartition reaches the head of the priority queue. Thebenefit is that a partition may receive ink from thesame node several times while waiting in the queue,so that the nodes in this partition receive ink in batchwhen this partition reaches the head of the queue. In

algorithm PA, the active ink drops fast and the terminationconditions may be fulfilled early. Thus, thenumber of iterations needed is largely reduced andso is the cost spent for maintaining the priorityqueue Q. Moreover, since the number of partitions is

much smaller than that of nodes, the size of queue Q

is much smaller compared to that used in BA, sooperations on it are fast as well. As an example, inFig. 5, in algorithm BA, node k2 distributes its active

ink to each of its three neighbor nodes d1–d3. However,in algorithm PA, the active ink of k2 is only distributedto two recipients: partitions PD

1 and PD

2 ; anunderlying document node will not receive the ink,until its partition reaches the top of the queue.

3) Lazy distribution mechanism. In BA, a node distributesink aggressively, i.e., each of its neighbor nodesreceives ink no matter how much it is. On the otherhand, in algorithm PA, we adopt a lazy distributionmechanism that relies on threshold _. If the amount ofthe ink to be distributed from a node v to a partition Pis smaller than _, P does not receive the ink immediately;instead, the ink is accumulated (i.e., buffered)at v. Later, if at some point the ink accumulated at vfor partition P exceeds _, P receives it. Overall, thislazy distribution mechanism delays the distributionof small amounts of ink across the graph that wouldotherwise result in many updates, reducing thecomputational cost significantly. As a toy example inFig. 5b, the amount of ink (0.07) to be distributedfrom node k2 to partition PD1 waits at k2 when _ ¼ 0:1.

Algorithm 2. PA

Input: GðD; K;EÞ, GKP , GDP , q ¼ ðkq; _qÞ, m, _

Output: C

1 PriorityQueueQ ;, C ;;

2 Add partition P 3 kq to Q with P:aink 1;

3 AINK 1;

4 while Q 6¼ ; and Q:top:ainkvi _ _ do

5 Deheap the top entry Pt from Q;

6 tm ¼ the top-m entry from C;

7 tm0 ¼ the top-ðm þ 1Þ entry from C;

8 if tm:rink> tm0:rink þ AINK then

9 break;

10 Spread the active ink to nodes in Pt;

11 for each node v in partition Pt do

12 distratio ¼ 1 ;

13 if v is a keyword query node then

14 distratio = 1 _ a ;

15 v:rink v:rink þ v:aink _ a;

16 AINK AINK _ v:aink _ a;

17 if there exist a copy t of v in C then

18 Remove t from C;

19 v:rink v:rink þ t:rink;

20 Add v to C;

21 Get partition set P connected from v in GKP ;

22 else

23 Get partition set P connected from v in GDP ;

24 for each partition Pi in P do

25 ink v:aink _ distratio _ ~wðv; PiÞ;

26 if ink þ v:acc:Pi _ _ then

Pi:aink ink þ v:acc:Pi;

if there exist a copy P0 i of Pi in Q then

Remove P0 i from Q;

Pi:ainkPi:aink þ P0 i :aink;

Add Pi to Q;

else

Accumulate ink at node v for Pi (v:acc:Pi);

return the top-m entries (excluding kq) in C;

IV. Conclusion

In this paper, we proposed a LKS system giving watchword proposals that are significant to the client data needs and in the meantime can recover pertinent archives close to the client area. A gaugealgorithm extended from calculation BCA is acquainted with take care of the issue. At that point, weproposed a parcel based calculation which processes the scores of the candidate keyword queries at thepartition level and uses a lazy mechanism to greatly decrease the computational cost. Observationalexaminations are directed to contemplate the effectiveness of our LKS framework and the execution ofthe proposed calculations. The outcome demonstrates that the structure can offer helpful proposals andthat PA beats the standard calculation fundamentally.

V. Future Work

In the future, we intend to additionally examine the adequacy of the LKS structure by gathering moreinformation and outlining a benchmark. What's more, subject to the accessibility of information, wewill adjust and test LKS for the situation where the areas of the inquiry guarantors are accessible in thequestion log. At last, we trust that PA can likewise be connected to quicken RWR on general chartswith dynamic edge weights; we will explore this potential later on.

VI. Related Work

Related work on query suggestion is discussed inKeyword query suggestion approaches can be classified intothree main categories: random walk based approaches, learningto rank approaches, and clustering based approaches. Wealso briefly review alternative methods that do not belong toany of these categories. To the best of our knowledge, no previous work considers user location in query suggestion.. Techniques for RWR computation are reviewedin Random Walk Computation Random walk with restart, also known as Personalized PageRank,has been widely used for node similarity measuresin graph data, especially since its successful application bythe Google search engine.

Acknowledgment

This work was funded by EC grant 657347/H2020-MSCAIF-2014 and by GRF grant 17205015 from Hong Kong RGC.Dingming Wu is the corresponding author.

References

[1] R. Baeza-Yates, C. Hurtado, and M. Mendoza, "Inquiry proposal utilizing question sign in websearch tools," in Proc. Int. Conf. Current Trends Database Technol.,2004,pp. 588– 596.

[2] D. Beeferman and A. Berger, "Agglomerative bunching of a internet searcher question log," inProc. sixth ACM SIGKDD Int. Conf. Knowl. Disclosure Data Mining, 2000, pp. 407– 416.

[3] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li, "Setting mindful question proposal bymining navigate and session information," in Proc. fourteenth ACM SIGKDD Int. Conf. Knowl.Disclosure Data Mining,2008, pp.875– 883.

[4] N. Craswell and M. Szummer, "Irregular strolls on the snap diagram," in Proc. 30th Annu. Int.ACM SIGIR Conf. Res. Create. Inf. Retrieval, 2007, pp. 239– 246.

[5] Q. Mei, D. Zhou, and K. Church, "Question proposal utilizing hitting time," in Proc. seventeenthACM Conf. Inf. Knowl. Oversee., 2008, pp. 469– 478.

[6] Y. Tune and L.- W. He, "Ideal uncommon question recommendation with verifiable clientcriticism," in Proc. nineteenth Int. Conf. Internet, 2010,pp. 901– 910.

[7] T. Miyanishi and T. Sakai, "Time-mindful organized question proposal," in Proc. 36th Int. ACMSIGIR Conf. Res. Create. Inf. Retrieval,2013, pp.809– 812.

[8] A. Anagnostopoulos, L. Becchetti, C. Castillo, and A. Gionis, "An streamlining system for inquiryproposal," in Proc. ACM Int. Conf. Web Search Data Mining, 2010, pp. 161– 170.

[9] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna, "The question stream chart:Model and applications," in Proc. seventeenth ACM Conf. Inf. Knowl. Oversee., 2008, pp.