IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-6, June 2018 E-ISSN: 2347-2693

Approximate Top-k Queries Monitoring on Document Streams

B. Mounika[1*], R. Raja Sekher[2]

[1,2,]Computer Science and Engineering, JNTU College of Engineering, Ananthapur, India

[*]*Corresponding Author: babburappamounika@gmail.com, Tel.: +91-9441577450*

Available online at: www.ijcseonline.org

Accepted: 08/Jun/2018, Published: 30/Jun/2018

Abstract— Document stream is the stream where documents are flows continuously. By monitoring these documents it is possible to have different applications of the real world like demand presentation, contextual advertisements, filtering of news updates, and general filtering of information to meet the needs of users. User preferences are used to process the top-k monitoring of documents streams continuously. However, it is a tedious and challenging task to fulfil the aspirations of various users and their preferences. In the literature many solutions are found. However an adaptive approach is essential to achieve better results. In this paper we proposed a framework and implemented to have continuous monitoring and approximation of document streams to Top-k queries of different users. Thus the proposed system yields more utility to end users than existing system. Top-k queries instead of preferences can provide the intent of users more clearly. Thus the filtered documents can reveal the user intention in making such queries. An algorithm named Adaptive Identifier Ordering (AIO) is implemented to achieve this. AIO adapts to the runtime dynamics of streaming besides using top-k queries to reports users with most appropriate documents. We build a prototype application to demonstrate proof of the concept.

Keywords— Document streams, top-k queries, continuous monitoring, adaptive identifier ordering.

I. INTRODUCTION

Data in the form of document is growing exponentially. It is evident in the era of big data where data is voluminous, streamed continuously with variety of data. Document streaming is the concept in which documents arrive to server from different sources. Making queries on the streams can help in obtaining information that is latest with high coverage. Social networking web sites like Twitter are generating text documents continuously. Processing such continuously streaming data is challenging but it bestows plethora of benefits. In order to understand the dynamics of document streaming, literature review is made which revealed different existing methods. Various filtering algorithms are explored in [1]-[4] and top-k queries are studied in [5]- [6].

The top-k queries on the document streaming is made in [7]-[8] Where revere ordering techniques like RIO and MRIO are explored. However, we considered continuous monitoring and making top-k queries on document streaming an optimization problem and proposed a method

known as Adaptive Identifier Ordering (AIO) which is adaptive in nature and suitable for continuous monitoring of document streams. Our contributions in this paper are as follows.

1. We built a new algorithm known as Adaptive Identifier Ordering (AIO) for continuous monitoring of top-k queries on document streams. This algorithm is adaptive in nature and found to be effective in making top-k queries on document streams.
2. We built a prototype application to demonstrate proof of the concept. The application has web based intuitive interface while the business logic is built in the server which provides response to user queries. The application is a web client from which user makes queries.
3. We evaluated the proposed algorithm and found it to have better performs when compared with other state-of-the-art algorithms.

The remainder of the paper is structured as follows. Section 2 provides review of literature. Section 3 presents the proposed system. Section 4 presents implementation details while section 5 covers the experimental results. Section 6 concludes the paper besides providing directions for future work.

II. RELATED WORK

This section provides review of literature on document streams and making top-k queries in such streams. For effective document retrieval different filtering techniques are

explored in [1], [2] and [3]. Content matching approach for document retrieval in the domain of publisher/subscriber is investigated in [4]. On the other hand top-k matching in the systems that are based on publisher/subscriber is explored in [5] Finding k most relevant publications in the publisher/subscriber model is studied in [6]. With respect to social annotation based news, in the context of publisher/subscriber, a system is proposed in [7] for top-k publishing. Many top-k query processing techniques are discussed in [8]. Sometimes, it is better to aggregate results so as to make it more meaningful. Such work is done in [9] for making middleware software more useful. Working with multiple aggregations with respect to streaming documents is made in [21].

Text mining and performing inverted search operations is the main focus of the study made in [10]. There are two-level procedures followed in [11] for evaluating queries processing. With respect to top-k queries, evaluation procedures are discussed in [12]. With respect to sliding windows, the problem of continuous top-k monitoring is explored in [13]. Top-k queries with user preferences is investigated in [14] for continuous performance with respect to top-k queries. Evaluation of such continuous top-k queries is made in [15] and [16]. With respect to document streaming and personalized query processing is investigated in [17] by using web 2.0 streams.

Monitoring of personalized hot news in the context of web 2.0 document streams is made in [18]. Documents with non-homogenous scoring functions are studied for processing continuous queries is made in [19] while approximate top-n queries on the documents streams are the study in [20]. In [22] a technique known as reverse identifier ordering is employed and they improved it to have efficient top-k queries and monitoring on document streams. From the review of literature it is understood that there has been considerable research on top-k queries on document streams. In this paper we found the suitability of adaptive approach to improve it further by considering it as an optimization problem.

III. METHODOLOGY

This section provides the problem formulation, the purpose of the proposed system, methodology followed to solve the problem besides an algorithm that is used to achieve the solution.

A) Problem Definition

Document streams are the streams of documents that continuously flow into a system. Unlike normal documents, the document streams are dynamic in nature and efficient processing of such documents is challenging. Defining similarity metric that is required by documents and queries is another important aspect to be considered. Provided set of documents D and set of queries Q, continuous top-k queries and monitoring them is the problem is to be addressed.

B) Purpose of the System

In the proposed system, a framework is designed and implemented to have continuous monitoring and approximation of document streams to Top-k queries of different users. Thus the proposed system yields more utility to end users than existing system. Top-k queries instead of preferences can provide the intent of users more clearly. Thus the filtered documents can reveal the user intention in making such queries. An algorithm named Adaptive Identifier Ordering (AIO) is implemented to achieve this. AIO adapts to the runtime dynamics of streaming besides using top-k queries to reports users with most appropriate documents. We build a prototype application to demonstrate proof of the concept.

C) Methodology

The methodology followed in the proposed system is as follows. Set of documents that arrive as a stream is denoted as $D = \{d1, d2, ..., d_n\}$. Each document has number of terms. The terms are denoted as $T=\{t1, t2, ..., t_n\}$. Each term is associated with a weight denoted as $\boldsymbol{f}$. Adaptive Identifier Ordering (AIO) is the algorithm proposed to have continuous monitoring of document streams with top-k queries. Each document has its ID denoted as d_{ID}. A set of words in dictionary is used to make a set of lists. Each term has its own list Li containing (d_{ID}, f_i). A set of queries given by users is denoted as $Q=\{q1, q2, ..., q_n\}$.

D) Adaptive Identifier Ordering

This algorithm is meant for achieving top-k results from document streams. It takes set of documents, set of queries and dictionary word collection as input and produces top-k results for each query in the given set of queries.

Adaptive Identifier Ordering (AIO)

Input: Set of documents D streamed at server, set of queries Q, Dictionary W

Output: Top k results for each query

01 Initialize vector for list L

Preparing fore ID ordering

2 For each word w in W
3 For each document d in D
4 For each query q in Q
5 Compute TF-IDF f_i for w
6 Update list L with d_{ID} and f_i for adapting
7 Save L
8 End For
9 End For
10
11 End For

Finding Top-K Results

11 For each query q in Q
12 Sort all lists based on ID
13 Find the average weights
14 Display top k results for query q
15 End For

The AIO algorithm takes the streamed documents, set of user queries and set of dictionary words. For each dictionary word, it computes set of lists and finally finds top k documents based on the relevancy. The relevancy is computed using TF-IDF approach of Okapi BM25. Similarity between a query q and the document d is computed as in Eq. 1 according to cosine similarity measure.

$$C(q,d) = \frac{q.d}{|q||d|} = \sum 1 \leq i \leq |T| \quad \text{wifi} \qquad (1)$$

With the help of similarity measure, it is easier to find out similarity of documents based on given user queries. Moreover the proposed approach is adaptive in nature which continuously adapts to the new dynamics of documents and queries.

IV. IMPLEMENTATION DETAILS

Implementation in the form of client application is made with web based interface. The application demonstrates proof of the concept. The classes involved in the application are
administrator, user, and web server. The outline of these roles is as presented in Figure 1.

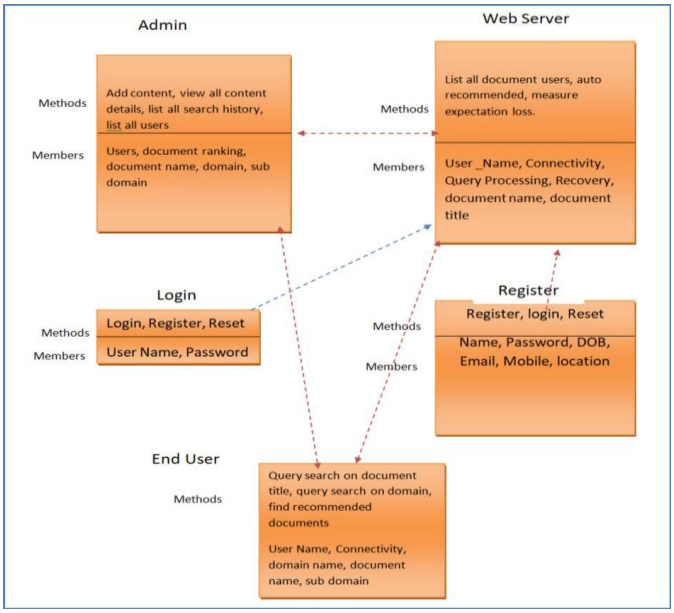


Figure 1: The classes and the outline of relationship among them.

As presented in Figure 1, it is evident that the admin role has activities like adding content, viewing content, viewing search history, and finding all users associated. The end user role has support for operations like query search on document title, query search on domain and finding recommendations or results of queries. These two roles do have interaction with login and registration objects. The data flow among the objects is as shown in Figure 2.

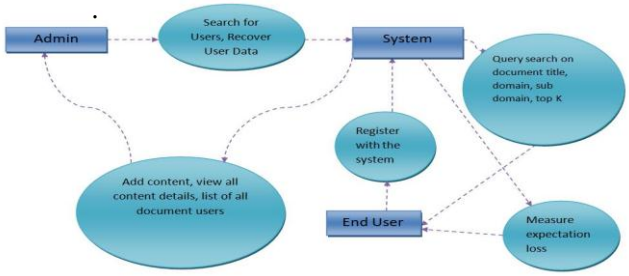


Figure 2: Shows communication flow among the objects.

As presented in Figure 2, it is evident that the system support query search on documents while users perform actual search operations. The admin can perform admin activities and controlling the document streams and users. The users and admin users can make use of the system with respective operations. It is role based and permit only intended operations to the roles involved in the system. It is intuitive in nature and users who login can see only the related operations required by them. Thus the system is able to provide functional requirements and also non functional requirements like security and usability.

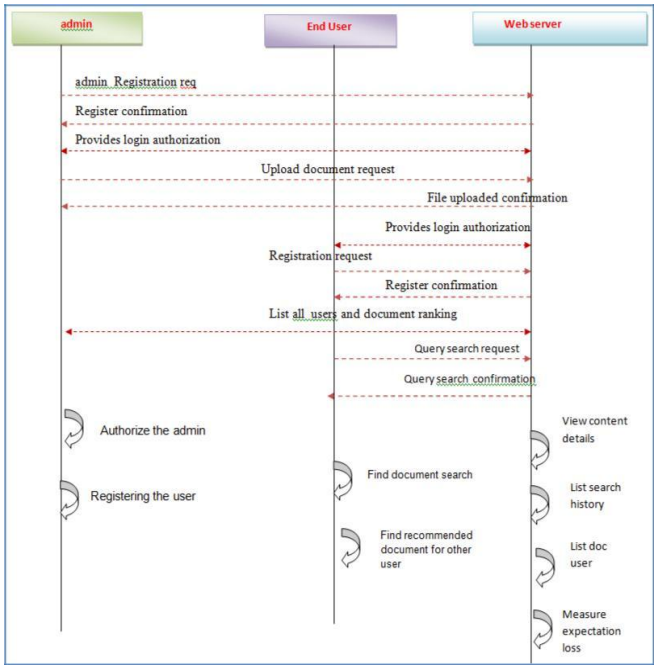


Figure 3: Interaction among the objects involve

As shown in Figure 3, it is evident that there is interaction among the admin, end user and web server. Since it is a web based application and the business logic is implemented in the server, there is interaction among the objects and the server has to give the results. The implementation is actually made with three tier architecture. The three tiers are known as client tier (browser where web client application runs), web tier (web server like Tomcat Server/Glassfish Server) and data tier like the document base. The algorithm proposed in this paper is implemented as part of the server side functionality.

V. RESULTS AND DISCUSSION

Experiments are made with different datasets like Wiki-Uniform and Wiki-Connected. More details on these datasets can be found in [22]. The observations made in the experiments include number of queries versus response time of the system and length of the queries versus response time.

A) Response Time Based on Number of Queries

Response time is observed based on number of queries made. As the number of queries is increased, the response time taken for RIO, MRIO and proposed AIO are observed and compared.

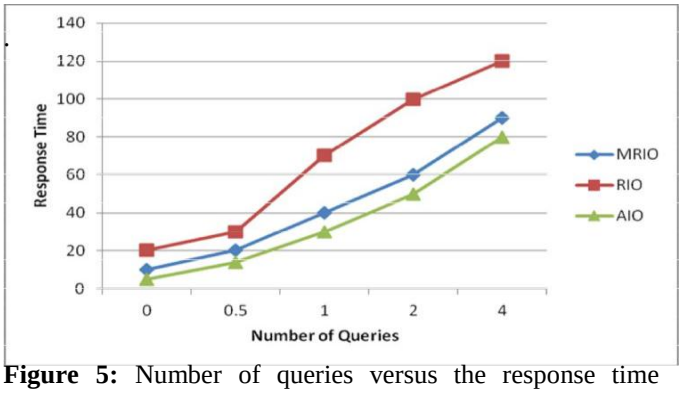
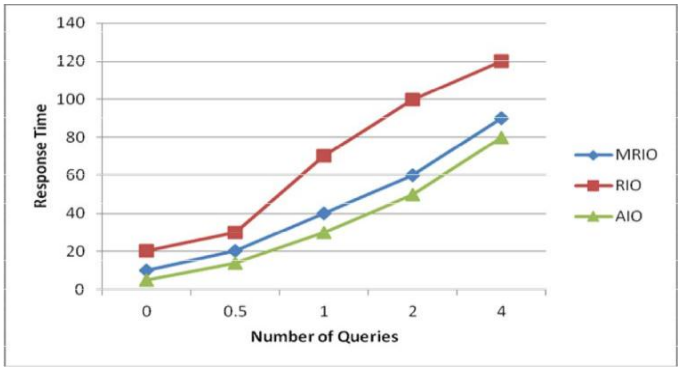


Figure 5: Number of queries versus the response time (Wiki-Connected)



As shown in Figure 5, it is evident that the number of queries is taken in horizontal axis while the vertical axis shows the response time observed against the number of queries. The response time of the RIO is less than that of MRIO. The proposed algorithm that is AIO outperforms both ROI and MRIO. As the number of queries is increased, the response time is also increased with linear relationship. These results are captured with the dataset Wiki-Connected.

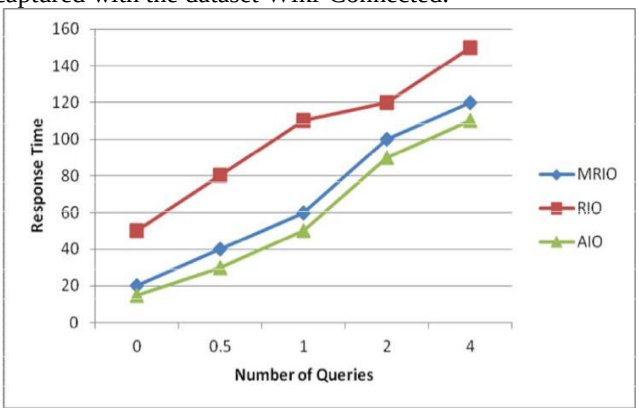


Figure 5: Number of queries versus the response time (Wiki-Connected)

As shown in Figure 5, it is evident that the number of queries is taken in horizontal axis while the vertical axis shows the response time observed against the number of queries. The response time of the RIO is less than that of MRIO. The proposed algorithm that is AIO outperforms both ROI and MRIO. As the number of queries is increased, the response time is also increased with linear relationship. These results are captured with the dataset Wiki-Connected.

B) Response Time Based on Length of Queries

Response time is observed based on length of queries. As the length of queries is increased, the response time taken for RIO, MRIO and proposed AIO are observed and compared for both the datasets.

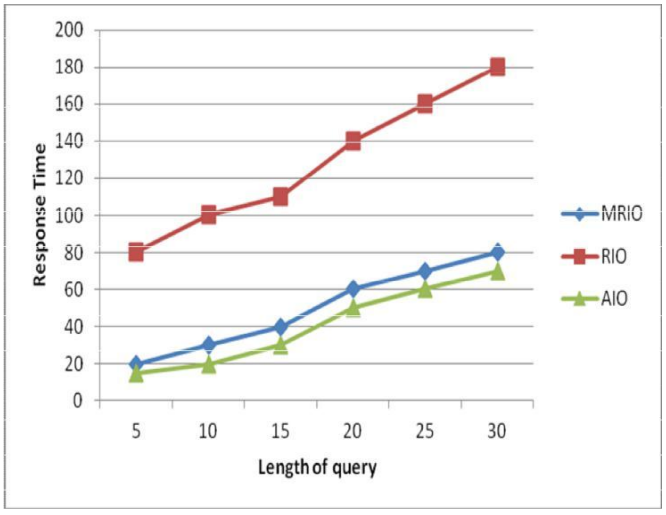


Figure 6: Length of queries versus the response time (Wiki-Uniform)

As shown in Figure 6, it is evident that the length of queries is taken in horizontal axis while the vertical axis shows the response time observed against the length of queries. The response time of the RIO is less than that of MRIO. The proposed algorithm that is AIO outperforms both ROI and MRIO. As the number of queries is increased, the response time is also increased with linear relationship. These results are captured with the dataset Wiki-Uniform.

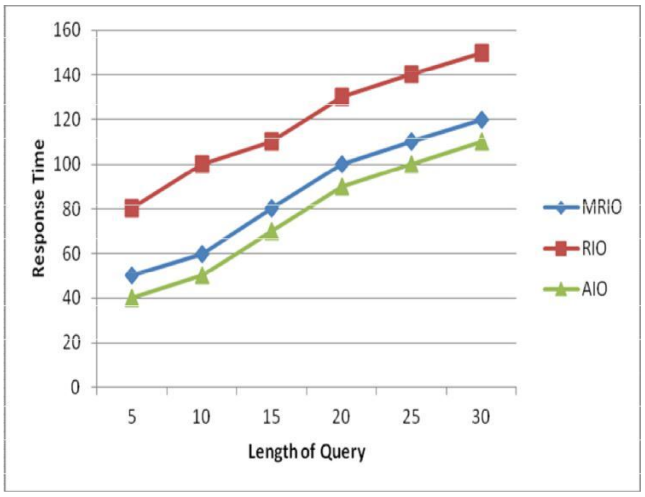


Figure 7: Length of queries versus the response time (WikiUniform)

As shown in Figure 7, it is evident that the length of queries is taken in horizontal axis while the vertical axis shows the response time observed against the length of queries. The response time of the RIO is less than that of MRIO. The proposed algorithm that is AIO outperforms both ROI and MRIO. As the number of queries is increased, the response time is also increased with linear relationship. These results are captured with the dataset Wiki-Uniform.

V. Conclusion and Future Scope

In this paper, document streams and the process of top-k queries and continuous monitoring of such system are explored. It is found that making top-k queries on continuous document streams is very challenging. In the literature different approaches are found to deal with document streams. However, an adaptive approach that continuously monitors document steams with queries is proposed in this paper. We studied RIO and MRIO methods presented in[22]. Inspired by the work, we proposed an algorithm known as AIO that follows an adaptive approach. We built a [23]prototype application and implemented the algorithm as part of a three-tier web application. Web interface is used to make experiments. However, the algorithm lies in server and the streaming takes place in server. End users can make queries from the web based interface in order to observe the response time in presence of number of queries and different length in the queries involved. The experimental results revealed the significance of AIO and its performance improvement over other state-of-the-art algorithms. In future we intent to improve the AIO method with new programming paradigm such as Map Reduce with distributed programming frameworks like Hadoop.

Acknowledgment

I thank to project guide Dr. R. Raja Sekhar, his timely advice and guidance have helped me complete this project in time.

References

[1] S. E. Robertson and D. A. Hull, "The TREC-9 Filtering Track Final Report", in 9 th Text Retrieval Conference (TREC-9), NIST SP 500-236, pp. 25– 40, 2000.

[2] Y. Zhang and J. Callan, "Maximum Likelihood estimation for Filtering Thresholds", in theProceedings of the 24th Annual International Conference on Research and Development in Information Retrieval (SIGIR 2001), Louisiana, USA, pp. 294-302, 2001.

[3] F. Fabret, H. Jacobsen, F. Llirbat, J. L. M. Pereira, K.A. Ross, and D. Shasha, "Filtering algorithms and implementation for very fast publish/subscribe", in the proceedings of the 2001 InternationalConference on Management of Data (SIGMOD 2001), California, USA, pp. 115-126, 2001.

[4] W. Rao, L. Chen, A. W.-C. Fu, H. Chen, and F. Zou, "On efficient content matching in distributed pub/sub systems", in the proceedings of the 2009 IEEE International conference on INFOCOM, pp. 756–764, 2009.

[5] M. Sadoghi and H. A. Jacobsen, "Relevance matters: Capitalizing on less (top-k matching in publish/subscribe)", in the proceedings of the 2012 IEEE 28th International conference on Date Engineering (ICDE 2012), pp. 127–138, 2008.

[6] K. Pripuzic, I. P. Zarko, and K. Aberer, "Top-k/w publish/subscribe: finding k most relevant publications in sliding time window w.", in the proceedings of the 2008 IEEE 28th International conference on DEBS, 2008, pp. 127–138, 2008.

[7] A. Shraer, M. Gurevich, M. Fontoura, and V. Josifovski, "Top-k publish-subscribe for social annotation of news," Journal Proceedings of the VLDB Endowment (PVLDB) , vol. 6, no. 6, pp. 385–396, 2013.

[8] I. F. Ilyas, G. Beskales, and M. A. Soliman, "A survey of top k query processing techniques in relational database systems", Journal of ACM Computing Surveys, vol. 40, no. 4, pp. 11:1-11:58, 2008.

[9] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," Journal of computer and system sciences, vol. 66, no. 4, pp.614–656, 2003.

[10] J. Zobel and A. Moffat, "Inverted files for text search engines", Journal of ACM Computing Surveys, vol. 38, no. 2, pp.1-56, 2006.

[11] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Y. Zien, "Efficient query evaluation using a two-level retrieval process." in proceedings of the

twelfth international conference on Information and knowledge management (CIKM 2003), New Orleans, LA, USA, pp. 426–434, 2003.
[12] M. Fontoura, V. Josifovski, J. Liu, S. Venkatesan, X. Zhu, and J. Y. Zien, "Evaluation strategies for top-k queries over memory resident inverted indexes" , Journal Proceedings of the VLDB Endowment (PVLDB), pp. 1213–1224, 2011.
[13] K. Mouratidis, S. Bakiras, and D. Papadias, "Continuous monitoring of top-k queries over sliding windows", in the 2006 ACM SIGMOD international conference on Management of data (SIGMOD 2006), pp. 635–646, 2006.
[14] A. Yu, P. K. Agarwal, and J. Yang, "Processing a large number of continuous preference top-k queries." the 2006 ACM SIGMOD international conference on Management of data (SIGMOD 2012), pp. 397–408, 2012.
[15] W. Rao, L. Chen, S. Chen, and S. Tarkoma, "Evaluating continuous top-k queries over document streams." World Wide Web, vol. 17, no. 01, pp. 59– 83, 2014.
[16] K. Mouratidis and H. Pang, "Efficient evaluation of continuous text search queries," IEEE Transaction on Knowledge and Data Engineering., vol. 23, no. 10, pp. 1469–1482, 2011.
[17] P. Haghani, S. Michel, and K. Aberer, "The gist of everything new: personalized top-k processing over web 2.0 streams", in proceedings of the 19 th ACM conference on Information and Knowledge Management (CIKM 2010), Ontario, Canada, pp. 489–498, 2010.
[18] P. Haghani, S. Michel, and K. Aberer, "Efficient monitoring of personalized hot news over web 2.0 streams," Computer Science – Research and Development, vol. 27, no. 1, pp. 81–92, 2012.
[19] N. Vouzoukidou, B. Amann, and V. Christophides, "Processing continuous text queries featuring non-homogeneous scoring functions." in proceedings of the 19 th ACM conference on Information and Knowledge Management (CIKM 2012), Hawaii, USA, pp. 1065–1074, 2012.
[20] N. Koudas, B. C. Ooi, K. Tan, and R. Zhang, "Approximate NN queries on streams with guaranteed error/performance bounds" in proceeding of the 30 th International Conference on Very Large scale Data bases Volume (VLDB 2004), Toronto, Canada, pp. 804–815, 2004.
[21] R. Zhang, N. Koudas, B. C. Ooi, D. Srivastava, and P. Zhou, "Streaming multiple aggregations using phantoms" , The VLDB Journal, vol. 19, no. 4, pp. 557–583, 2010.
[22] L. Hou U, J. Zhang, K. Mouratidis, and Y. Li. "Continuous Top-k Monitoring on Document Streams", IEEE Transaction on Knowledge and Data Engineering, vol. 29, no. 5, pp. 991-1003, 2017.

Author Profile

B. Mounika received B.Tech in Information Technology from Sri Padmavathi Mahila viswavidyalayam Tirupathi, in 2014. Currently, she is pursuing M.Tech Computer Science and Engineering from JNTUA College of Engineering, Ananthapuramu, Andhra Pradesh, India. Her areas of interests include Big data.

Dr R Raja Sekhar received the Ph.D. degree from JNTU College of Engineering, Ananatahapuramu, Andhra Pradesh, India in 2018. He is currently an Associate Professor with the Computer science and Engineering, JNTU College of Engineering. His current research interests Mobile Ad-hock Networks, Compilers and algorithms include digital multimedia forensics.

