International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-6, August, 2019

A New Ranked-Key Structure for Intelligent Pub-Sub Systems in Large Scale

Mohamedou Cheikh Tourad, Abdelmounaim Abdali

Abstract*: The rapid growth of information published on the net and the subsequent increase of the number of it users was the reason behind the discovery of Web syndication. The Pub/sub (Publish/Subscribe) system is the first and the most used system for web syndication. It allows the user to keep an eye on the development of the information that interested him. This information is managed by data collectors (RSS : Really Simple Syndication) in real time. Today, the development of new notifications systems is essential for the retrieval of specific information from the Big Data. In the present work, a new algorithm based on Ranked key is proposed. Such indexation allows the development of an efficient and an intelligent system.

***Keywords* : Pub-Sub, Ranked-key, Inverted-list, Large scale**

I. INTRODUCTION

For the web syndication, the Pub/sub system, based on Web content, is most popular to facilitate the notification. In this system, the number of subscriptions is more important and the margin for growth is huge. The goal of these systems is to re-establish a connection ”publishers→←subscribers”, to assist in the management of the system. In addition, the flow of arrivals items can reach, at some events, thousands of articles per second. Items and subscriptions are defined by sets of keywords. A subscription satisfies an item, on the sole condition that its keywords are included in all of the item’s keywords (broad match semantics)[1],[2]. This article presents the development of a new ICRIL (Inverted Concept of Ranked-key Inverted List) algorithm for indexing structures to implement an efficient notification system. In order to avoid the recurrent problems related to the expensive access to the disk, the index subscription need to be performed in the main memory. The goal of our approach is to find the proper indexing structure of subscriptions that is compact and effective in the search memory. The problem corresponding to a Pub/sub system is to find all subscriptions associated with an article [1]: a subscription responds to an article, if and only if its terms values satisfy the conjunction of terms and conditions of the subscription [3]. Web subscription uses terms of a textual type (key terms) and of a structured type (attribute-value pairs).

Lemma 1: Item $i \in I$ (all items) satisfies a subscription $s \in S$ (all subscriptions) on the sole condition that $\forall t_j \in s \Rightarrow t_j \in i$ (t_1 , t_2 being the elements forming the subscription and items list). Given an item i, the problem is to find all subscriptions S_M (Subscriptions Matched) $\subseteq S$ that satisfy. The remainder of this article is organized as follows: Sec- tion II presents the existing works. Section III. provides a description of our proposed algorithm; Section IV. presents analytical models of RIL and our approach ICRIL (Inverted Concept Ranked-key Inverted List); Section V. illustrates simulation results from our approach in comparison to the RIL algorithm; Section VI. concludes this work and presents future prospects for our work.

Revised Manuscript Received on October 30, 2019.
* Correspondence Author
Mohamedou Cheikh Tourad*, Applied Mathematics and Computer Science Laboratory, Cadi Ayyad University, Marrakech, Morocco. Email: cheikhtouradmohamedou@gmail.com
Abdelmounaim Abdali, Applied Mathematics and Computer Science Laboratory, Cadi Ayyad University, Marrakech, Morocco. Email: a.abdali@uca.ma

II. RELATED WORKS

The indexing structure of Pub/sub systems must index into the main memory to enable the search for all satisfied subscriptions to be simplified and a large number of subscriptions to be handled. So to satisfy subscriptions in real-time and to support high-margin items, that’s why the system must perform a new fast and efficient search algorithm of indexing structure. The literature presents a number of indexing techniques for main memory subscriptions, including:

A. Approaches based on clusters

These approaches [4] are often used in communications applications. They are considered as channels of communication in which each user specifies topics of interest. With a large number of topics on the web and the high flow events, dissemination events need huge time for transmission of events and maintenance of indexes. In these systems, subjects are separated and each is individually owned, although many topics have a significant number of public subscriptions. In this context we cite the following works: [4] proposes to group almost similar subjects of the same subscribers but treated separately, his approach is also used to index subscriptions based systems content, [5] proposes to group subscriptions on their predicate equality: two subscriptions with equality predicates for the same attributes are placed in the same group. A subscription is placed in one of these groups. Inequality predicates are stored in the same manner as in an algorithm based on the counter.

B. Counter and Inverted-lists

In this type of indexing [6] terms are indexed in inverted lists, where each term has a list allocated to it which contains all subscriptions including the said term. In this context, we cite the following works: The Ranked Key is a concept proposed by Yan, T.W. et al. [7] in 1994, based on the inverted lists, in which user profiles are represented by sets of key terms. The subscription identifier is added to the lists of all terms in conventional inverted lists, whereas this solution only sees it added to the list of the subscription's least frequent terms and, in turn, the list of remaining terms. In the work of Fabret, F. et al. [5] and Pereira, J. et al. [8] a count-based index provides the terms; here, the processing of subscriptions involves a count of the terms within an item. The work of Irmak, U. et al. [9] in 2006 focuses on a counting algorithm that takes into consideration how occurrences of terms are distributed, with the aim of optimizing subscription processing using clustering techniques. The new BRCQ system (Boolean Range Continuous Queries), proposed by Chen, L. et al. [10] in 2013, makes use of IQ-Tree (Inverted File Quad-tree), with the goal being to satisfy significant flows of incoming publication queries and objects. This hybrid index approach uses a tree to store geographical information requests, by means of the Ranked Key method. The W-IQ-TREE proposed by Trifinopoulos, J. et al. [11] in 2016, can be presented as an evolution IQ Tree. Only subscription is new Ranked-key index proposed by Tourad, M.C. et al. [12], [13] in 2017. With regard to CILs (Count-based Inverted List) and RILs (Ranked key Inverted List), which are based on inverted lists proposed by Hmedeh, Z. et al. [14] in 2016, the CIL algorithm is used to count key terms of the subscription item (Count-based), whereas the aim of the RIL algorithm is to identify the least frequent term within a particular subscription (Ranked-key).

C. Correspondence Trees

Correspondence Trees [1] are indexing structures that are commonly used to perform rapid searches. Despite their significantly reduced search space, their corresponding memory space has a high financial cost. Performing a search from the root, while respecting the search criteria, enables unnecessary browsing of sub-tree subsets to be eliminated. Kale, S. et al. [3] in 2005 proposed a tree-based RAPIDMatch algorithm, working on the principle that a large number of events have a small number of relevant terms according to which they could be processed. Using dual score levels, subscriptions are indexed within a tree structure. The BE-Tree (Boolean Expression Tree) proposed by Sadoghi, M. et al. [15] in 2011 is an index based on the key attribute which enables expandable with a huge number of Boolean-expressions in a multidimensional space. W-IQ-TREE proposed by Trifinopoulos, J. et al. [11] in 2016. In 2013, Sadoghi, M. et al. [16] proposed an improved BE*-Tree version, analytically studying memory consumption and the times for adding subscriptions and processing. In 2016, Constantin, C. et al. [17] put forth an innovative tree-like index stand on an algebraic coding AS-Index. ROT (Regular Ordered Sorts) was proposed by Hmedeh, Z. et al. [14] in 2016 featuring an alternative for inverted lists involving a type of Ordered Tree index, which is also capable of constructing a hierarchical search space using common prefixes between subscriptions.

D. Graph-based approaches

Graph-based approaches [18] make use of the subscriptions hedging relationship by increasing the efficiency for reporting systems with a distributed architecture. Of relevance to this context is the work by Liu, H. et al. [19], who proposed a graph-based Pub/sub system. Each subscription takes the form:

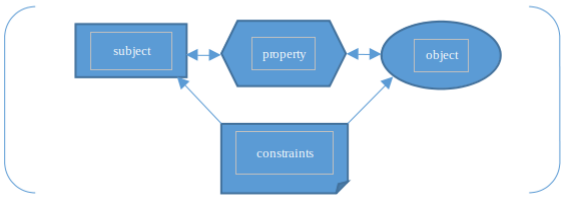

Fig. 1.structure subscription of graphs index

The table below shows the state of indexing techniques for the following approaches (cluster, inverted lists and counter, trees, graphs) with respect to their processing in the Hard Disk and the main memory:

- ⊗Indicates that the treatment is very expensive.
- ⊕Indicates that the treatment is less expensive.

Table- I: THE DIFFERENT INDEXING TECHNIQUES

Indexing techniques	Disc	Main memory
Approaches based on clusters	⊕	⊗
Inverted lists and counter	⊕	⊕
Approaches based on trees	⊕	⊗
Approaches based on graphs	⊕	⊗

A valuable study was undertaken in 2016 by Hmedeh, Z. et al. [14] comparing the various indexing techniques for subscriptions in memory (inverted lists and counters, trees, graphs). This comparative study focuses on structures capable of being stored in the random access main-memory, this condition eliminates Trees and Graphs structures. This comparative study give a way that Ranked key is the best choice to large scale.

III. PROPOSED INVERTED CONCEPT OF RANKED-KEY INVERTED LIST APPROACH

The RIL algorithm will first be presented before our Inverted Concept of Ranked-key Inverted List approach. Web syndication Pub/sub systems involve subscribers submitting long-term (continuous) queries as subscriptions based on key terms. Each new item published is assessed with respect to all subscriptions submitted to the system, and the subscriber receives a notification for each subscription.

Given that $\delta S = \{t_1, t_2, ..., t_n\}$ the length of S referred to by |S| is the total number of distinct terms it contains, where $I = \{i_1, i_2, ..., i_m\}$ refers to the new incoming item flow. A new item for $i \in I$ also comprises a set of distinct terms. As in [21], a frequent assumption is made that presents the condition $\delta S \subseteq \delta I$ (δS, δI are vocabularies of distinct items and subscriptions). In reality, however, it is noteworthy that there may a significant difference between δS and δI. Table 2 below presents the example that we will use to demonstrate the functioning of the two algorithms:

A. Ranked-key Inverted List

This algorithm [7] adds a subscription identifier to the inverted list of the subscription key. All remaining subscription

Table- II: Terms and Frequency

Terms	Tf
t_1	{4}
t_2	{4}
t_3	{3}
t_4	{2}
t_5	{2}
t_6	{1}
t_7	{1}

Table- III: All Subscriptions S

Subscriptions	Terms
S_1	$\{t_1 \wedge t_2 \wedge t_3 \wedge t_4\}$
S_2	$\{t_1 \wedge t_2 \wedge t_3 \wedge t_5\}$
S_3	$\{t_1 \wedge t_2 \wedge t_4 \wedge t_5\}$
S_4	$\{t_1 \wedge t_2 \wedge t_6 \wedge t_7\}$

terms, with the exception of the subscription identifier, are added to the inverted list. A check will be performed on subscriptions indexed in the inverted list for each of a published item's terms. In inverted lists, of course, the most frequent words are minimized, while terms with average frequency are amplified. Fig. 2 presents the RIL index for our example and the ranking of terms (t_1 is the most frequent δS term). It should be noted that the most frequent terms do not appear in the index dictionary. Thus, t_1 and t_2 are absent from the vocabulary, in all cases existing in a subscription of S together with a term of low frequency. Posting(t_4) includes two subscriptions ($s_1 \wedge s_3$), the remaining words of which are recorded in the list. Processing of $i=\{t_1, t_2, t_3, t_4\}$ begins by testing the inclusion of subscriptions. Posting(t_4) for s_3 : i does not include t_5, and as a result s_3 is not satisfied. With respect to s_1, terms t_1, t_2 and t_3 appear in $i' = i - t_4 = \{t_1, t_2, t_3\}$ and s1 contains no other terms, and will from that point onwards be subject to notifications.

RIL-Matching: The processing of an item i first requires its terms to be sorted by rank: $Sort_{ascending}(i)$ followed by the processing of the inverted-list of the least-frequent term, Posting(tj). For each Posting(t4)subscription, item i will subsequently be checked for the presence of each subscription s term:

B. Proposed Approach Inverted Concept of Ranked-key Inverted List (ICRIL)

In the context of large scale, what we are finding is that subscriptions and terms are increased in a huge way. Our approach adds the subscription identifier in the inverted list of the most frequent term. All remaining subscription terms, with the exception of the subscription identifier, are added to the inverted list. A check will be performed on subscriptions indexed in the inverted list for each of a published item's terms. In inverted lists, of course, the least frequent words are minimized, while terms with average frequency are amplified. The least frequent terms are the largest, while the most frequent terms are shorter. Compared with the Ranked-key concept, where subscriptions have common keys. We propose a new keys concept, where the key is the most frequent term in a subscription, this is the inverted concept of Ranked-key Inverted List. If the current subscription identifier exists in the previous inverted list, it will not be added to the new inverted list, which will enable the dictionary and postings to be reduced in terms of size. The difference will be greater when passing to a big scale in the memory and in processing. Fig. 3 presents the ICRIL index for our example

ICRIL-Matching :The same processing is undertaken by ICRIL Matching as by Ranked-key Inverted List, but there is a difference with regard to Posting(t_j) and a new descending sort for terms of i:$Sort_{descending}(i)$ Our approach to a new $Posting^{ICRIL}(t_j)$, where the $Posting^{ICRIL}(t_j)$ is not to contain a subscription s such that s is associated with $Posting^{ICRIL}(t_{j-1})$.

Algorithm 1 : Inverted Concept Ranked-key Inverted List

```
Require: an item i
sorted terms ← sorted descending(i)

while sorted_terms ≠ {} do

    tj ← mostFrequent(sorted terms)

    post_set ← Posting^ICRIL(tj)

    for all s ∈ post_set do

        if s remaining ⊆ sorted_terms then

            SMatched = SMatched ∪{s}

        end if

    end for

end while
```

The follows: function $Posting^{ICRIL}(t_j)$ is defined as

```
post_set = []
 for all s ∈S do
    if s ∉ Posting^ICRIL(tj−1) then
         if tj ∈s then
              post_set = post_set ∪ {s}
         end if
    end if
end for
return post_set
```

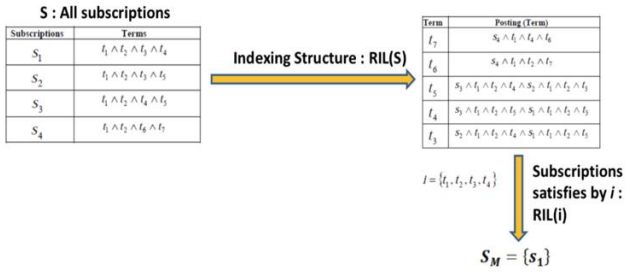

Fig. 2.Application example with RIL

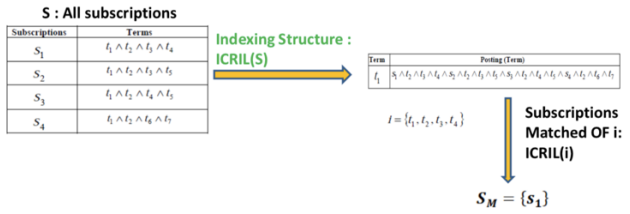

Fig. 3.Application example with ICRIL

IV. ANALYTICAL MODELS

Presented in this section are the new analytical models developed for subscription indexing, with these models able to utilise the memory space and matching time of both these structures (RIL and ICRIL). A summary of all parameters and notations is presented in Table 4 below

Table- IV. PARAMETERS

Parameters	Descriptions
El	|subscription posting-entry|
Ed	|dictionary-entry|
Fr(t_j)	tf distribution in dI
δI	distinct items
δS	distinct subscriptions
α	$\|subscription\|_{avg}$
β	$\|subscription\|_{max}$
Y	|Item|
wi	Probability that a term has a rank ≥i
φi	Probability that a term has a rank ≤i
η(k)	Probability that a subscription has a size k
X	Len(subscriptions)

A. Indexing time

The average subscription size is provided by α.The time required (excluding sorting) for a subscription s to be added to the inverted list (IL) is provided by O(α). Conversely, the indexing types (RIL and ICRIL) require the subscription terms to be sorted, in order to determine the least frequent term for RIL and the most frequent term for ICRIL, where the indexing time is provided by: O(α × log(O(α))). α is defined by:

$$\alpha = \sum_{k=1}^{\beta} \eta(k) \times k \qquad (1)$$

B. Memory space

The estimated memory used by both these structures can be calculated using a probability number, with Fr(tj) representing the occurrence frequency of tj ∈ δI. A random selection of subscription terms is made from each |δI| depending on their frequency distribution. δS subscriptions have a vocabulary size equal to:

$$|\delta S| = \sum_{j=1}^{|\delta I|} P(t_j \in S) \qquad (2)$$

RIL memory space : The RIL index [7] comprises subscription lists and a dictionary representing the least frequent terms for each subscription s∈S. Γ(RIL) = Γ(Dic) + Γ(Postings) The number of least frequent words largely determines subscription list size. Subscription s is added to the t_j term list on the sole condition that $t_j \in s$ and on the condition that $t_{ji} \in s$ does not exist with rank(t_i) ≤ rank(t_j). The rank represents the terms ordered by ascending frequency. The probability of a subscription being contained within the t_j list is:

$$\Phi(s, t_j) = \alpha \times Fr(t_j) \times \sum_{k=1}^{\beta} \varphi_{j-1}^{k-1} \qquad (3)$$

φ_i, with the probability of having a rank(term) ≤ i.

For a subscription size k, there are k possibilities that t_j is chosen, and other terms will be selected from within the most frequent terms. For a term t j to feature in the dictionary, there must be a subscription as a minimum associated with t j in its inverted list, as given by Equation (4):

$$\left(1 - \left(1 - \Phi(s, t_j)^x\right)\right) \qquad (4)$$

Here, the probability of the term t j being the least frequent in a subscription s with a size k is presented: from (4), we deduce that the dictionary size is given s ∈S by:

$$\Gamma(Dic) = \sum_{j=1}^{|\delta S|} \left(1 - \left(1 - \Phi(s, t_j)\right)^x\right) \times E_d \qquad (5)$$

In comparison to [7], it is proposed that the memory space used by RIL is:

$$\Gamma(RIL)=\sum_{j=1}^{|\delta S|}\left(1-\left(1-\Phi(s,t_j)\right)^{X}\right)\times E_d+\sum_{j=1}^{|\delta S|}\alpha\times\sum_{s\in S}\Phi(s,t_j)\times E_l \quad (6)$$

With the knowledge that the probability of the term being the least frequent in a subscription s with a size k, the Posting(t_j) size is presented by:

$$\Gamma\left(Postings(t_j)\right)=\alpha\times\sum_{s\in S}\Phi(s,t_j)\times E_l \quad (7)$$

ICRIL memory space: The ICRIL index comprises subscription lists and a dictionary representing the most frequent terms for each subscription s∈S. Γ(ICRIL) = Γ(Dic) + Γ(Postings) The dictionary and inverted lists are smaller in size than as defined in RIL. Subscription s is added to the list of terms t_j on the condition that $t_j \in s$ and only on the condition that there is no term $t_i \in s$ with rank(t_i) ≥ rank(t_j). Compared with RIL in (3), the probability is given by:

$$\lambda(s,t_j)=\alpha\times Fr(t_j)\times\sum_{k=1}^{\beta}\omega_{j-1}^{k-1} \quad (8)$$

ω_i , with the probability of having a rank(term)≥ i. If {$list_1$,$list_2$,$list_3$,..., $list_{j-1}$ } the probability is given by:

$s \in \Omega(s, t_j) = \lambda(s, t_j) \times (1 - \lambda(s, t_1)) \times (1 - \lambda(s, t_2)) \times \ldots \times (1 - \lambda(s, t_{j-1}))$

This is written as:

$$\Omega(s,t_j)=\lambda(s,t_j)\times\prod_{i=1}^{j-1}(1-\lambda(s,t_i)) \quad (9)$$

The dictionary size is defined by:

$$\Gamma(Dic)=\sum_{j=1}^{|\delta S|}\left(1-\left(1-\Omega(s,t_j)\right)^{X}\right)\times E_d \quad (10)$$

The size of the Posting(tj) is defined by:

$$\Gamma\left(Posting(t_j)\right)=\alpha\times\sum_{s\in S}\Omega(s,t_j)\times E_l \quad (11)$$

Compared with G(RIL), the following is obtained:

$$\Gamma(ICRIL)=\sum_{j=1}^{|\delta S|}\left(1-\left(1-\Omega(s,t_j)\right)^{X}\right)\times E_d+\sum_{j=1}^{|\delta S|}\alpha\times\sum_{s\in S}\Omega(s,t_j)\times E_l \quad (12)$$

C. Matching time

RIL matching time :The necessary processing time for an item i is a function of its number of terms and the size of the subscription list. The terms of item i must first be sorted, then the lists browsed in order to test the inclusion of subscriptions for item i. An estimated cost for processing item i can be calculated by: Δ(RIL) = Δ(Tri) + Δ(Postings)

$$\Delta(RIL)=Y\times\log(Y)+\sum_{j=1}^{Y}\alpha\times\sum_{s\in S}\Phi(s,t_j)\times\tau_1 \quad (13)$$

Where τ_1 represents the time required to test whether a term is included in item i. ICRIL matching time : While the necessary sorting time for item i is the same as in RIL, ICRIL requires less time to match subscription lists. The estimated cost for processing item i can be calculated by: Δ(ICRIL) = Δ(Tri) + Δ(Postings)

$$\Delta(ICRIL)=Y\times\log(Y)+\sum_{j=1}^{Y}\alpha\times\sum_{s\in S}\Omega(s,t_j)\times\tau_2 \quad (14)$$

Where τ_2 represents the time required to test whether a term is included in item i.

V. ASSESSMENT OF PERFORMANCE

The standard Python.3.6 was used to implement the indices. All simulations were carried out using a 3.60 GHz quad- core processor and a 16 GB memory for the JVM. Here, we present how data structures, in addition to the associated parameters, were selected and implemented. The index constituted on the basis of inverted lists (ICRIL, RIL) comprises the dictionary representing the vocabulary terms for subscriptions, and lists of indexed subscription identifiers. Simulations were performed with our real dataset of RSS, with the total number of analyzed items amounting to 17 million. The items acquired were stored in a MongoDB JSON file. Their textual content was a vocabulary set formed of 3 million individual terms, a figure that was extracted and used to generate subscriptions. This subscription generation made use of the Alias [21] method based on term occurrence distribution by selecting actual use. The characteristics defining the generation of subscriptions are:

- Len(subscriptions) = X.
- | subscription | = k.

The vocabulary size δI used and the term occurrence distribution with regard to subscriptions.

A. Memory space

Number of subscriptions: Fig. 4 compares RIL and ICRIL results, and indicates that RIL requires more memory space than ICRIL. This statement remains true for any number of subscriptions.

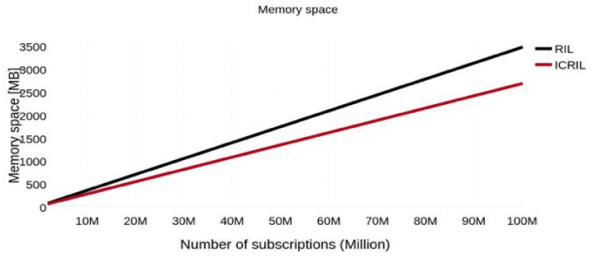

Fig. 4.Memory space as a function of the number of subscriptions

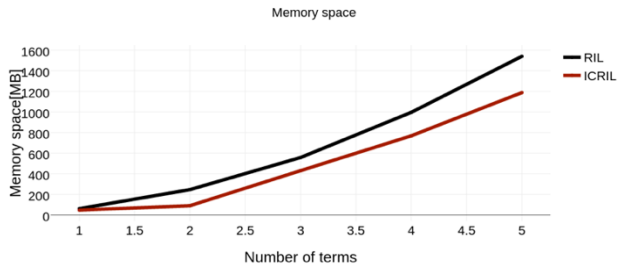

Fig. 5.Memory space as a function of the number of terms

Number of terms: Fig. 5 compares RIL and ICRIL results, and indicates that RIL requires more memory space than ICRIL. This statement remains true for any number of subscriptions.

B. Matching time

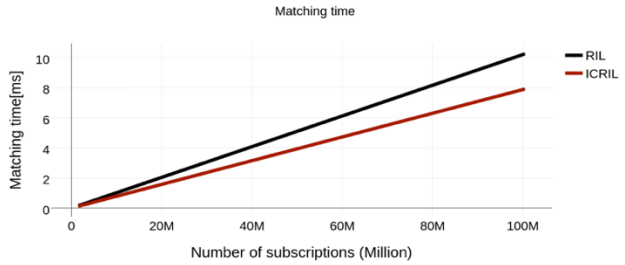

Fig. 6.Matching time as a function of the number of subscriptions

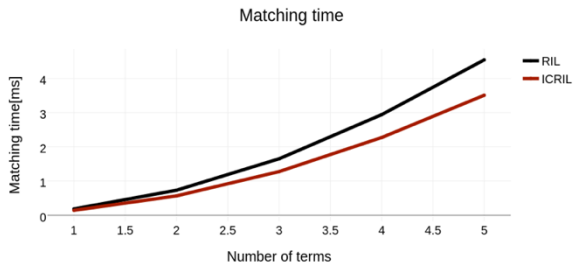

Fig. 7. Matching time as a function of the number of terms

Number of terms: Fig. 7 presents the evolution of RIL and ICRIL as a function of the number of terms, and indicates that RIL requires a longer matching time than ICRIL.

C. Indexing time

The average indexing time was measured for 10 M subscriptions generated using 15 M terms. The index based on inverted lists was faster to construct, requiring only 0.5 µs to insert a subscription into ICRIL (in comparison to 0.8 µs for RIL). We offer a simpler approach to inserting a new subscription into ICRIL indexing. Once the subscription is added to the first list of most frequent terms, it does not appear in any other lists, whereas in RIL subscriptions may appear in all lists of least frequent words.

Table- V: Comparing RIL and ICRIL

	RIL	ICRIL
Memory espace	63%	37%
Matching time	63%	37%
Indexing time	60%	40%

Table V explains that ICRIL have more advantages compared to the RIL algorithm at all levels.

VI. CONCLUSION

In conclusion, we present two subscription indexing algorithms in Pub/sub systems. We also came out with a comparison of the performance of the proposed algorithms for notification subscription. ICRIL indexing structure consumes less memory space by comparison with RIL. The calculations showed that RIL is slower compared to the ICRIL algorithm for scaling the number of indexed subscriptions. with the flow of publication high online, a user risks finding the volume of information received excessive or unmanageable. We will work on the partial satisfaction between the items and the subscriptions based on the similarity measures [22].

REFERENCES

1. Aguilera, M.K., Strom, R.E., Sturman, D.C., Astley, M. and Chandra, T.D., 1999, May. Matching events in a content-based Subscription system. In Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing (pp. 53-61). ACM.
2. Zhou, D., Lawless, S. and Wade, V., 2012. Improving search via personalized query expansion using social media. Information retrieval, 15(3-4), pp.218-242.
3. Kale, S., Hazan, E., Cao, F. and Singh, J.P., 2005, June. Analysis and algorithms for content-based event matching. In 25th IEEE International Conference on Distributed Computing Systems Workshops (pp. 363- 369). IEEE.
4. Milo, T., Zur, T. and Verbin, E., 2007, June. Boosting topic-based publish-subscribe systems with dynamic clustering. In Proceedings of the 2007 ACM SIGMOD international conference on Management of data (pp. 749-760). ACM.
5. Fabret, F., Llirbat, F., Pereira, J. and Shasha, D., 2000, September. Efficient matching for content-based publish/subscribe systems. In Proc. CoopIS.
6. Zobel, J. and Moffat, A., 2006. Inverted files for text search engines. ACM computing surveys (CSUR), 38(2), p.6.
7. Yan, T.W. and Garc ıa-Molina, H., 1994. Index structures for selective dissemination of information under the boolean model. ACM Transac- tions on Database Systems (TODS), 19(2), pp.332-364.
8. Pereira, J., Fabret, F., Llirbat, F., Preotiuc-Pietro, R., Ross, K.A. and Shasha, D., 2000, September. Publish/subscribe on the web at extreme speed. In VLDB (pp. 627-630).
9. Irmak, U., Mihaylov, S., Suel, T., Ganguly, S. and Izmailov, R., 2006, June. Efficient Query Subscription Processing for Prospective Search Engines. In USENIX Annual Technical Conference, General Track (pp. 375-380).
10. Chen, L., Cong, G. and Cao, X., 2013, June. An efficient query indexing mechanism for filtering geo-textual data. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (pp. 749-760). ACM.
11. Trifinopoulos, J., Nguyen, L.T., von Haeseler, A. and Minh, B.Q., 2016. W-IQ-TREE: a fast online phylogenetic tool for maximum likelihood analysis. Nucleic acids research, 44(W1), pp.W232-W235.

12. Tourad, M.C., Abdali, A. and Outfarouin, A., 2016, November. On a new index of Publish/Subscribe System in the context of Big Data. In 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA) (pp. 1-2). IEEE.
13. Tourad, M.C. and Abdali, A., 2017, March. Toward efficient ranked-key algorithm for the web notification of big data systems. In Proceedings of the 2nd international Conference on Big Data, Cloud and Applications (p. 31). ACM.
14. Hmedeh, Z., Kourdounakis, H., Christophides, V., Du Mouza, C., Scholl, M. and Travers, N., 2016. Content-based publish/subscribe system for web syndication. Journal of Computer Science and Technology, 31(2), pp.359-380.
15. Sadoghi, M. and Jacobsen, H.A., 2011, June. Be-tree: an index structure to efficiently match boolean expressions over high-dimensional discrete space. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (pp. 637-648). ACM.
16. Sadoghi, M. and Jacobsen, H.A., 2013. Analysis and optimization for boolean expression indexing. ACM Transactions on Database Systems (TODS), 38(2), p.8.
17. Constantin, C., Du Mouza, C., Litwin, W., Rigaux, P. and Schwarz, T., 2016. As-index: A structure for string search using n-grams and algebraic signatures. Journal of Computer Science and Technology, 31(1), pp.147- 166.
18. Shen, Z., Aluru, S. and Tirthapura, S., 2005. Indexing for Subscription Covering in Publish-Subscribe Systems. ISCA PDCS, 5, pp.328-333.
19. Liu, H., Petrovic, M. and Jacobsen, H.A., 2007. Efficient filtering of RSS documents on computer cluster. Technical Report, MSRG, University of Toronto.
20. Busch, M., Gade, K., Larson, B., Lok, P., Luckenbill, S. and Lin, J., 2012, April. Earlybird: Real-time search at twitter. In 2012 ieee 28th international conference on data engineering (pp. 1360-1369). IEEE.
21. Walker, A.J., 1977. An efficient method for generating discrete random variables with general distributions. ACM Transactions on Mathematical Software (TOMS), 3(3), pp.253-256.
22. Tourad, M.C. and Abdali, A., 2018. An Intelligent Similarity Model between Generalized Trapezoidal Fuzzy Numbers in Large Scale. International Journal of Fuzzy Logic and Intelligent Systems, 18(4), pp.303- 315.

AUTHORS PROFILE

Mohamedou Cheikh Tourad received the degree of specialized Engineer from the Faculty of Sciences and Technology, Marrakesh, Morocco, in 2014, where he is currently pursuing a Ph.D. degree with the Computer Science Department of the Faculty of Sciences and Technology. His research interests are in Intelligent Publish/Subscribe System, Big Data and Fuzzy logic, Data science, Machine learning, Deep learning.

Abdelmoumaim Abdali received a Ph.D. in Solid Mechanics and Structures from the University of Amiens, France, in 1996. He is a professor of Computer Science at the University Cadi Ayyad, Faculty of Sciences and Technology, Marrakech, Morocco, and a member of the Laboratory of Applied Mathematics and Computer Science (LAMAI) Marrakesh, Morocco. His research interests are: Publish/Subscribe System, software engineering, Big Data, Fuzzy logic, computer science, DTN net- work, business intelligence, design and implementation in data warehouse, ubiquitous systems, modeling & design, metamodel design, model transformation, model verification & validation methods, numerical simulation, smart cities, biomechanics, and bone remodeling.