Future Generation Computer Systems 95 (2019) 163–174

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

An efficient event matching system for semantic smart data in the Internet of Things (IoT) environment

Noura Alhakbani [a], Mohammad Mehedi Hassan [a,*], Mourad Ykhlef [a], Giancarlo Fortino [b]

[a] Department of Information Systems, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia
[b] Department of Informatics, Modeling, Electronics, and Systems, University of Calabria, Italy

HIGHLIGHTS

- Utilization of semantic Smart data for efficient event matching in IoT environment.
- A semantic Pub/Sub model is designed for semantic Smart data matching with subscriptions.
- The matching algorithm utilizes a tree-based structure to provide efficient communication to support time-critical Smart data applications.

ARTICLE INFO

Article history:
Received 11 July 2018
Received in revised form 18 September 2018
Accepted 29 December 2018
Available online 9 January 2019

Keywords:
Event matching
IoT
Publish/subscribe
Semantic

ABSTRACT

The publish/subscribe model for communication has proved to be the most suitable in the Internet of things (IoT) environment because of the decoupling provided by this model that supports communication among heterogeneous parties. The standard or common publish/subscribe uses exact model to match events to subscriptions. However, in the IoT environment, an exact match is an extreme requirement because of the diverse and large environment and generation of various forms of Smart data. Therefore, semantically similar events must be considered and returned to subscribers as a possible match. However, matching events approximately to subscriptions is a much more complex task that negatively affects the efficiency of matching. Our proposed algorithm, semantic matching using the tree structure (SMT), provides efficient communication to support time-critical applications. SMT achieved linear time in terms of throughput compared with exponential time achieved in previous work. Combining SMT with taxonomy clustering improved the effectiveness in terms of the F-score, which is an indication of the recall and precision of the results, particularly when 100% of subscriptions were to be semantically matched.

1. Introduction

Internet of things (IoT) sensors are increasing significantly, according to Gartner forecast published in 2017 [1] "20.4 billion connected things will be in use worldwide by 2020". The high rate of growth in the use of IoT in our daily lives demands effective and efficient communication between them to achieve their potential. The continuous increase in the adoption of the IoT around us in smart environments on different sized scales, such as smart homes, smart universities, and smart cities, emphasizes the need for a suitable middleware for communication. The middleware should be extendable in terms of supporting heterogeneous sensing and actuating devices. To date, there has been no such communication standard, which creates a challenge [2].

For the middleware to be usable in communication, it should support scalability, and help in crossing semantic and syntactic boundaries between communicating parties [3]. Moreover, it should be efficient given the real-time requirements of the time critical applications and the large volume of data to be communicated. According to [4], some of the most important areas of key IoT opportunities are fleet management, security, and surveillance. For example, Lufthansa Airlines is using real-time aircraft, airport, and weather sensor data to improve on-time performance and optimize operations. These areas need real-time processing because of their time-critical requirements. Moreover, most of these time-critical applications deal with big data at the same time [2].

The publish/subscribe paradigm is well suited for large-scale distributed systems because of its ability to provide scalable, efficient communication. It supports decoupling in terms of time, space and synchronization with limited resource usage [5]. Furthermore, semantic boundaries should be crossed to facilitate communication between different parties. Agreeing on a rigid unified syntax among many heterogeneous devices and users is unrealistic. Communication should be relaxed by crossing the semantic boundary and achieving approximate matching to facilitate communication between publishers and subscribers.

* Corresponding author.
E-mail address: mmhassan@ksu.edu.sa (M.M. Hassan).

https://doi.org/10.1016/j.future.2018.12.064
0167-739X/

The challenge arises in this type of communication due to its large scale and highly heterogeneous participants [6]. Besides, approximate matching of events in the publish/subscribe paradigm complicates the matching process and negatively affects efficiency. Moreover, the absence of agreement on an event schema or the lack of use of a predefined ontology hinders achieving accurate matching results Yet, time-critical applications such as disaster or emergency related require the communication to be effective and efficient at the same time.

Very few studies have applied approximate semantic event matching using semantic decoupling to the publish/subscribe paradigm [7]. Their proposed approach for approximate semantic event matching is based on explicit semantic analysis (ESA) and the frontier algorithm for semantic matching. In our previous work [8], we proposed approximate semantic event matching publish/subscribe system. We used taxonomy clustering (TC) with the top-k candidates algorithm and ESA to approximately match events to subscriptions, which significantly improved the effectiveness compared with previous results. Clustering achieved a 40% increase in F-score results. However, in [7] and [8], the throughput time increased exponentially with increasing subscriptions. Thus, to improve the efficiency we propose the semantic matching using the tree structure (SMT) algorithm, which also uses ESA to semantically match events to subscriptions. However, we use optimized, limited size tree structure, where the top node can be expanded efficiently. Then, we combine the TC approach proposed in [8] with (SMT) and called it the TC-SMT algorithm to improve effectiveness alongside improved efficiency. Our contributions are outlined below:

- We proposed approximate semantic event matching publish/subscribe system for IoT platform that clusters IoT-related events and subscriptions according to distinct topics using taxonomy clustering (TC) and then perform matching using SMT approach for events and subscriptions that fall within intersected clusters.
- Our proposed SMT reduced the total complexity of the algorithm from O (n.m.log(m) + n.log(n) + k.n^2 + k.log(k)) achieved in [7] and [8] to O (n.m. log (m) + n.k. log (k)) .
- Finding the top-K candidates events for a subscription in our proposed algorithm is increasing linearly O (n.k. log (k)) compared to O (n.log(n) + k.n^2 + k.log(k)) achieved in [7] and [8]. The increase is linear for different values of subscription predicates and number of events, compared with an exponential time increase in previous work [7] and [8].
- The achieved effectiveness of the TC-SMT algorithm maintained similar or slightly higher results than the results achieved in [8].

The remainder of the paper presents related work, followed by our motivation, then the requirements and research question. Next, we present the requirements and our research questions. Then the language model used followed by the proposed architecture explained. Subsequently, SMT algorithm explained, then the experiment, and experimental results. Finally, Discussion and conclusion are presented.

2. Literature review

In this section, we discuss semantic publish/subscribe and semantic coupling, and then present the types of semantic publish/subscribe and existing work for each type.

Semantic Publish/Subscribe and Semantic Coupling

The traditional publish/subscribe paradigm matches subscriptions with precisely matching events. The publish/subscribe paradigm, in general, is based on the idea of decoupling in terms of time, space, and synchronization. However, the traditional publish/subscribe paradigm is based on tight coupling with respect to syntax. Publishers and subscribers assume an agreement regarding event types, properties, and values called type coupling, property coupling, and value coupling respectively:

- Type coupling means agreement on the type or class of event attributes or instances.
- Property coupling means agreement on the set of values for the event attributes or instances.
- Value coupling means agreement on the set of values for event attributes or instances.

Semantic publish/subscribe loosens the above couplings, and matches subscriptions with events that have a similar meaning.

In the following section, we present different categories of semantic publish/subscribe matching algorithms in the literature.

2.1. Concept-Based methods

The concept-based approach is based on a concept-level shared agreement or the use of domain-specific ontology.

S-TOPPS [9] introduced the use of a concept hierarchy, to allow different relationships between the schema and attribute values for publish/subscribe algorithms such as generalization and specialization or mapping functions. In [10], the resource description framework (RDF), graph patterns, and DAML was used to represent events, subscriptions, and subscription language respectively. Ontologies and relational operators to correlate and map events were used in [11]. Fuzzy ontology was used for mapping in [12], and events were represented using attribute–value pairs. In [13], they converted the content-based publish/subscribe model to concept-based applying shared ontology library on multidimensional tree structure, and then perform event-matching using the cover of subscriptions information and matching order to speed matching. Semantic-based clustering and ontologies were used to achieve energy-efficient event routing in [14]. An atom-based container for semantic information was proposed in [15] to be used in a service-oriented system to improve the quality of service for resource-constrained devices. In [16], the authors predicted the subscribers using ontology-based quantitative similarity metric and semantic similarity found in historical events. Complex event services (CES) ontology was used in [17]. We can observe that concept-based methods are mostly based on ontologies or a shared agreement of concepts. Using ontologies does not support terminology decoupling. Moreover, expanding the ontology in use by adding a new term or concept can become cumbersome. Therefore, ontologies are difficult to apply in diverse and dynamic environments.

2.2. Approximate semantic event matching

Approximate semantic event matching is based on a statistical model built over distributional semantics, the results are ordered according to the approximation. One of the first proposals of approximate semantic event matching was A-TOPPS [18]. The authors introduced the use of approximate operators and match degree rather than crisp Boolean. In [19], events were represented as RDF, and subscriptions were queries in SPARQL. The results were ranked according to similarity. In [7] and [20], the authors used the attribute–value model to represent events. The results were ranked according to similarity relatedness. The performance efficiency decreased with an increase in the number of attributes that were to be matched. In [2], the authors mentioned that one of the research gaps is the generation of IoT knowledge before it becomes outdated, and therefore, high processing throughput is a necessity. To speed up the matching process and increase the throughput, the *k*-nearest neighbors (*k*NN) algorithm was used in [21]. However, the *k*NN algorithm need to be manually trained before applying it for different patterns or types of queries to be able to match accurately.

2.3. Thematic event processing

Approximate semantic event matching is very useful for event matching in the IoT; however, the matching process should be more efficient. In [20] and [22], theme tags that represent the domain were added to the payload of the subscription or event to filter the incoming events according to the filter or theme to speed up the matching process. Yet, this requires the user to choose a theme alongside events and subscriptions.

3. Motivation

Publish/ subscribe paradigm is decoupled in terms of space, time and synchronization. However, the communication is based on exact syntax matching of events and subscriptions. Yet, it is necessary that communication is flexible and accept approximate or semantic matching due to the heterogeneity and diversity of data providers and consumers. Tying the communication to an agreed upon terms beforehand limit the communication and result in high cost when change is needed [7,23].

Semantic interoperability significance and the need for semantic publish/subscribe middleware has been emphasized in the report published on March 2015 by the European Research Cluster on the IoT [24]. It suggested loosening the semantic coupling i.e. decoupling to be well suited for the IoT diverse environment [7, 19,20].

Many of the IoT applications would greatly benefit from loosening the semantic coupling so the communication between different parties do not need to be based on beforehand agreement between communicating parties. [7,19,23].

Most work in this area focused on ontology based semantic matching which require agreement on agreed upon ontology between communicators beforehand which is difficult in an open environment which holds on to the tight coupling in a different way. Moreover, changing or adding to the ontology due to the dynamic nature of the environment is costly, cumbersome, and unfeasible [16,17].

The state of the art work in approximate semantic matching is presented in [7,20] and they addressed it in an original way and has managed to achieve low cost of rules management.

Yet, loosening the exact syntax matching complicates the matching process and the results were negatively affected in terms of effectiveness and efficiency when the number of subscriptions and events increase.

Given the exponential increase in the number of things to communicate through the internet and consequently the number of events; the matching algorithm need to be more efficient which means bigger number of events are matched in less time. Moreover, the approximate matching is not based on ontology or agreed on taxonomy beforehand which negatively affects the effectiveness of matching i.e. matching accurately when approximated.

Below we present a sample scenarios that can happen in an IoT environment that emphasizes the need for efficiency given the lack of well-defined agreed on schema.

Assume a person gets in a car accident and needs to find the nearest ambulance or health practitioner available. The device in the car is programmed to ask for help; in case of an accident it sends a request for help with the location. Time is very critical in similar situations; an ambulance nearby could reroute to the designated location as soon as it receives the request. The request should be understood by the ambulance even if it did not use the exact words expected by the ambulance.

4. Requirements and research questions

The following user requirements for an IoT event paradigm are identified after conducting the literature review and realizing the motivation.

R1. Usability and ease of maintenance of the event processing system by nontechnical users

R2. Efficient and effective event processing of IoT events

Those user requirements can be developed into the following high-level research questions.

Q1. How can semantic matching be efficient given the added complexity of loosening the exact matching to approximate semantic matching?

Q2. How can semantic matching between subscribers and publishers be achieved effectively?

Our work is focused on enhancing the efficiency of approximate semantic publish/subscribe paradigm while maintaining effective, simple and easy-to-use communication.

5. Language model

In this paradigm, one of our objectives is to use easy to use language for subscriptions and events, so users with limited technical background could easily communicate.

A subscription uses the attribute–value model to constitute the predicate. The equality operator (=) is used between the attribute and the value. A tilde operator (∼) is appended to the attribute or/and value to denote the acceptance of an approximate match. A subscription requires an exact syntax match to the attribute or value when no tilde operator is appended to it. An example of a subscription is presented below in Eq. (1)

$$\begin{aligned} \textit{Subscription} = \{&\text{Major} \sim \\ &= \text{Information Systems} \sim,\ \text{Degree} \sim \\ &= \text{Bachelor} \sim\} \end{aligned} \tag{1}$$

Subscription in Eq. (1) means the subscriber is asking for someone with bachelor's degree and major Information Systems. The tilde "∼" sign here means that the subscriber would want events that approximately matches the subscription i.e. not necessarily exact match to the syntax of this provided subscription for all its attributes and values.

Possible event would look like the following event presented in Eq. (2):

$$\textit{Event} = \{\text{Department} = \text{IS},\ \text{Certificate} = \text{B.S.}\} \tag{2}$$

5.1. Mapping

Given the loose semantic matching between subscriptions and events, there are different possible mappings σ between an event and a subscription. There are two possible mappings between the proposed subscription above in Eq. (1) and the event proposed in Eq. (2).

$\sigma 1 = \{($
Major $\sim =$ Information Systems $\sim \leftrightarrow$Department $=$ IS),
(Degree $\sim =$ Bachelor $\sim \leftrightarrow$Certificate $=$ B.S.)}
$\sigma 2 = \{($
Major $\sim =$ Information Systems $\sim \leftrightarrow$Certificate $=$ B.S.),
(Degree $\sim =$ Bachelor $\sim \leftrightarrow$Department $=$ IS)}

Each mapping has a different score of the similarity sum that reflects the degree of relatedness of the mapping to the subscription.

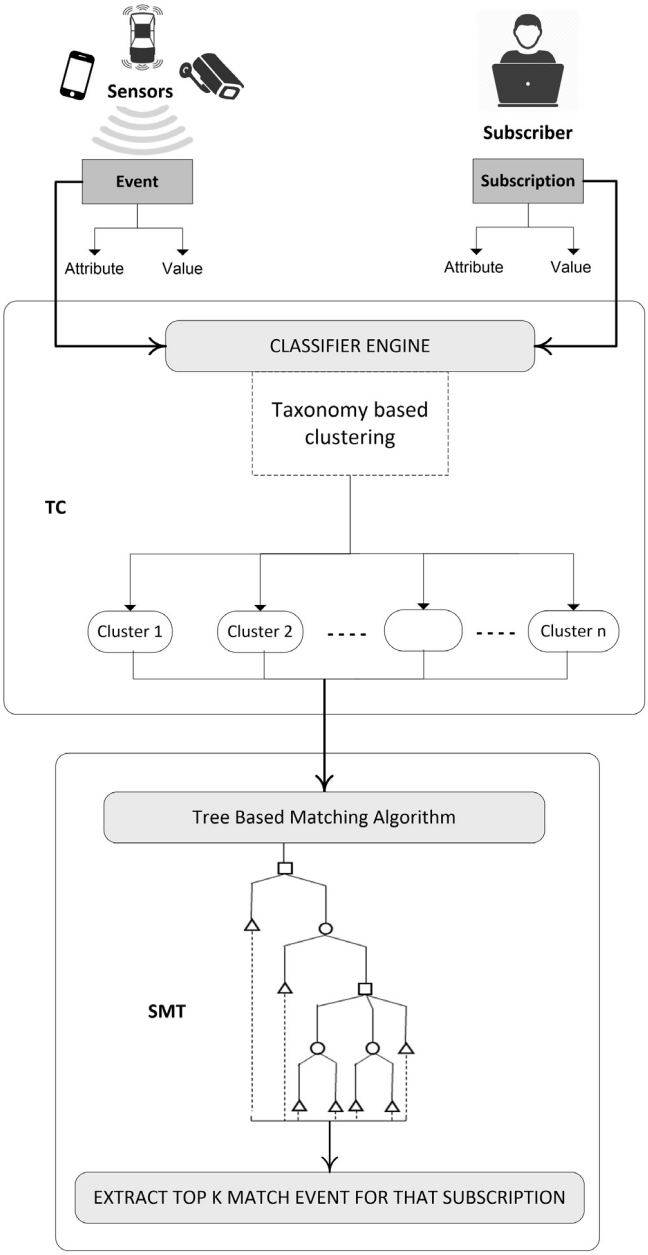


Fig. 1. Proposed system architecture for publish/subscribe using TC-SMT.

6. Proposed architecture

Given the uncertainty of approximate semantic matching of an event to a subscription; there exist different possible mappings of an event to a subscription. Finding the top_k mappings rather than the best mapping which corresponds to the top_1 increase the chance of hitting the best match to a subscription according to the statistical monotonicity principal [25]. This principal roughly states that a better precision could be achieved when mapping with a slightly smaller similarity compared to mapping with higher similarity due to the statistical distribution.

In this work, we propose the SMT algorithm, which uses the tree structure to efficiently find most related mappings to a subscription. The matching is performed approximately using ESA or exactly as required by the subscriber.

We use the tree as an optimized data structure, where each node represents a mapping, and the tree size is limited to the number of top k required mappings.

The tree is expanded from the root node to add subsequent nodes that represents other mappings. Only nodes which reflects mappings within the highest values for their similarity sums are added to the tree.

The best mapping is added to the tree as the root node accompanied with its calculated relatedness score, and since the subsequent top mappings are adjacent or relatively close to the top mapping. Computing the relatedness score of an adjacent mapping to the top mapping does not require recalculating all elements of the mapping, calculating the difference in mapping to the top mapping is sufficient to find the similarity score.

This model improves the efficiency of matching since it reduces the computation time for calculating different mappings.

Subsequently, we propose TC-SMT, a clustering method adopted from [8], for which subscriptions are classified according to their taxonomy and distributed accordingly to different classes before applying SMT. Then, within each class, subscriptions are matched to events using SMT, as explained in the next section in detail. Fig. 1 shows our architecture, which is used to match events to interested subscribers

7. Taxonomy Clustering for the Semantic matching using Tree Structure algorithm (TC-SMT)

To facilitate event matching in the IoT environment given the large number of events that need to be matched to relevant subscriptions, a topic is assigned to each event or subscription using taxonomy clustering (TC). Events and subscriptions are assigned topics according to their predicates and tuples. The events and subscriptions which are assigned the same topic are grouped into one cluster. Events are matched to subscriptions that fall within the same cluster only. This cause a reduction in the number of events to be matched to events that share the same topic only which improves effectiveness in terms of finding the right match and improves efficiency since only events that fall within the same subscription's cluster need to be matched Then events are matched to subscriptions within their relevant clusters. The details of event and subscription clustering algorithm is presented in detail in [8].

We used the AYLIEN text analysis API. We used the IPTC Subject NewsCodes, which contains 1400 classes organized into three levels of depth, to classify events and subscriptions. The dataset we used in the experiments was classified into 53 distinct clusters.

7.1. Semantic Matching Using the Tree Structure (SMT) Algorithm

(1) Building the Combined Similarity Matrix

From the language model presented above we can see that some predicates of the subscription require exact matching and others require approximate semantic matching. So, four different matrices are created to calculate the exact and semantic matching for each mapping.

Exact Attribute Matrix

$$= \begin{cases} 0 & \text{if } (\textit{Subscription\_Attribute requires exact match}) \text{ and } (\textit{Subscription\_Attribute}) \neq (\textit{Event\_Attribute}) \\ 1 & \textit{otherwise} \end{cases} \quad (3)$$

Exact Value Matrix

$$= \begin{cases} 0 & \text{if } (\textit{Subscription\_Value requires exact match}) \text{ and } (\textit{Subscription\_Value}) \neq (\textit{Event\_Value}) \\ 1 & \textit{otherwise} \end{cases} \quad (4)$$

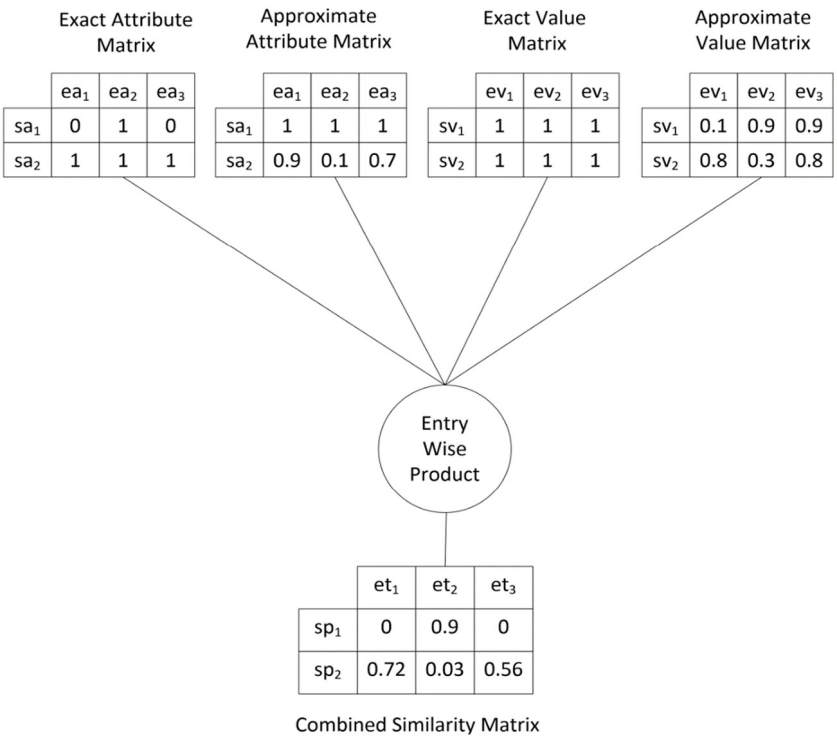


Fig. 2. Building the combined similarity matrix.

Approximate Attribute Matrix

$$= \begin{cases} ESA\,(Subscription\_Attribute, Event\_Attribute) \\ if\ (Subscription\_Attribute\ to\ be\ approximately\ matched) \\ \\ 1 \quad otherwise \end{cases} \tag{5}$$

Approximate Value Matrix

$$= \begin{cases} ESA\,(Subscription\_Value, Event\_Value) \\ if\ (Subscription\_Value\ to\ be\ approximately\ matched) \\ \\ 1 \quad otherwise \end{cases} \tag{6}$$

Exact matrix yield 1 if the event's attribute or value is an exact match to the subscription's attribute or value, otherwise it is 0. Exact matrices are built according to Eqs. (3) and (4)

Approximate matrix uses ESA relatedness measure to measure the relatedness between the subscription and events. ESA yields values in the range of [0, 1]. ESA is used whenever a tilde sign is present in the subscription's predicate. ESA semantic relatedness value calculation is explained in the next section. Approximate matrices are built using Eqs. (5) and (6).

The matrices dimensions are $n \times m$, where n is the number of subscription's predicates and m is the number of tuples in the event to be matched.

We obtain a single matrix called the combined similarity matrix M using the entry-wise product of all four matrices as presented below in Fig. 2. Entry-wise product is efficient since it uses characteristics of the identity (1) and zero (0) elements from the multiplication operator and can be computed in O($n\,m$).

(2) Explicit Semantic Analysis (ESA)

In our proposed algorithm, SMT, we use ESA [26] to measure semantic relatedness between terms, which has proven to yield acceptable results. The corpus used to determine semantic relatedness is Wikipedia, which naturally suits the field of IoT because of its dynamic, unlimited, and continuously growing in nature.

We downloaded the Wikipedia dump for 2016, and indexed it using Lucene [27]. Each Wikipedia article is represented as a vector of words that appear in the corresponding article. Each word is assigned a Term Frequency Inverse Document Frequency (TFIDF) value which reflects the relevance of a concept to a term. TFIDF is calculated by calculating the term frequency (tf) for term i across all the articles in the Wikipedia corpus and then calculating the document frequency (df) which calculates the number of documents (d) or articles that contain the termi.

$$tf\,(t_i, d_j) = \begin{cases} count\,(t_i, d_j) & if\ count(t_i, d_j) > 0 \\ 0 & otherwise \end{cases}$$

$$\mathrm{df_i} = |\{\mathrm{dk}: \mathrm{t_i} \in \mathrm{d_k}\}|$$

Next, we calculate the inverse document frequency (idf) which is obtained by dividing the total number of documents (n) by the number of documents containing the $term_i$, and then taking the logarithm of that quotient.

$$(\mathrm{idf_i} = \log \frac{\mathrm{n}}{\mathrm{df_i}})$$

Subsequently, TFIDF is calculated by calculating the product of (tf) and (idf).

$$\mathrm{tfidf} = (\mathrm{tf}) \cdot (\mathrm{idf})$$

Finally, we apply cosine normalization to the tfidf to disregard differences in document length. Cosine normalization is obtained by weighting all the components of a term vector

$$W_{ij} = \frac{\mathrm{tfidf[i,\,j]}}{\sqrt{\sum_{i=1}^{s} \mathrm{tfidf[i,\,j]}^2}},$$ where s is the number of terms.

A more detailed explanation of ESA can be found in [26].

(3) Semantic Matching using Tree Structure Algorithm

From our resultant unsorted combined similarity matrix M of size (n x m), we sort the matrix and call it SM and then create a sorted index matrix called IM, as shown below in Eqs. (7), (8) and (9).

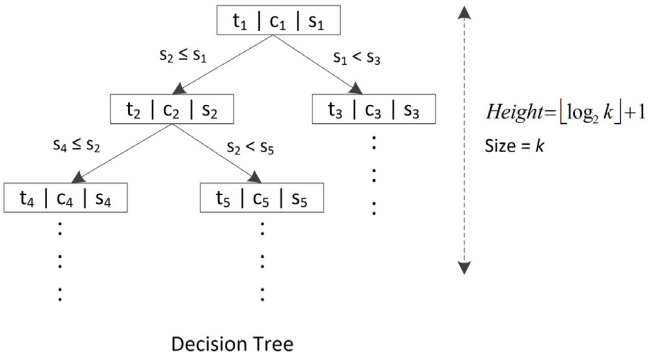


Fig. 3. Tree structure.

In our proposed algorithm, we use a limited sized tree of size k nodes, as shown in Fig. 3. Each node in our tree consists of three values: t, c, and s, where t denotes the index of the parent vector, c denotes the position of change, and s denotes the sum of the similarity values. The size of the tree is limited to k nodes, where k denotes the number of the required events to be chosen as matches to a subscription, and therefore the height of the tree is $\lfloor \log k \rfloor + 1$.

$$M = \begin{bmatrix} 0.8 & 0.7 & 0.9 & 0.3 \\ 0.5 & 0.8 & 0.2 & 0.4 \\ 0.0 & 0.1 & 0.8 & 0.3 \end{bmatrix} \tag{7}$$

$$SM = \begin{bmatrix} 0.9 & 0.8 & 0.7 & 0.3 \\ 0.8 & 0.5 & 0.4 & 0.2 \\ 0.8 & 0.3 & 0.1 & 0.0 \end{bmatrix} \tag{8}$$

$$IM = \begin{bmatrix} 3 & 1 & 2 & 4 \\ 2 & 1 & 4 & 3 \\ 3 & 4 & 2 & 1 \end{bmatrix} \tag{9}$$

The steps for building the tree from the sorted matrix are summarized as shown below in Fig. 4. Fig. 4: Building the tree from the sorted matrix we could see the tree size is limited is to k and k = 4. Nodes which have smaller sum than the smallest node when the total count of the nodes are 4 are discarded. Moreover, we can see that moving from one iteration to the other is done efficiently. It is not necessary to re-sum the similarity vectors each time neither pass the entire offspring vectors to the tree.

Assume we choose $k = 4$, So, the tree will have four nodes.

We create matrix R presented in Eq. (11) of size kn to store the best top k vectors, where k is the number of nodes, which is similar to the number of rows in R, and n is the number of rows in matrix M, which is similar to the number of columns in R

Below we explain in further detail the steps of building the tree from the sorted matrix. The first vector $V_1 = (1, 1, 1)$ from SM is the best because it has the highest sum value s.

1. Add vector $V_1 = (1, 1, 1)$ index positions as shown in the SM presented in Eq. (10) into row 1 in matrix R, as shown in Eq. (11).

$$SM = \begin{bmatrix} \mathbf{0.9} & 0.8 & 0.7 & 0.3 \\ \mathbf{0.8} & 0.5 & 0.4 & 0.2 \\ \mathbf{0.8} & 0.3 & 0.1 & 0.0 \end{bmatrix} \tag{10}$$

$$R = \begin{bmatrix} 1 & 1 & 1 \\ - & - & - \\ - & - & - \\ - & - & - \end{bmatrix} \tag{11}$$

2. Create an empty tree and assign null to the root of the tree. Then, insert the first vector with node values (1, 0, s) into the tree. S is calculated as presented in Eq. (12). Assign the node to be as the Top node since it is the node with the highest value of s in the tree, as shown in Fig. 5, the first best vector has no position change; therefore, it is assigned zero.

- node $(1, 0, s)$
  - $s = M[1, IM[1, 1]] + M[2, IM[2, 1]] + M[3, IM[3, 1]] = 0.9 + 0.8 + 0.8 = 2.5.$ (12)

3. Create temporary node and call it Temp. Store Top into Temp and delete Top. Then expand Temp, as shown in Fig. 6, to the following nodes by exchanging one element at a time. The calculation of the sum of the expanded nodes presented below in Eqs. (13), (14), and (15).

- node $(1, 1, s_1)$
  - $s1 = s - M[1, IM[1, 1]] + M[1, IM[1, 2]] = 2.5 - 0.9 + 0.8 = 2.4$ (13)
- node $(1, 2, s_2)$
  - $s2 = s - M[2, IM[2, 1]] + M[2, IM[2, 2]] = 2.5 - 0.8 + 0.5 = 2.2$ (14)
- node $(1, 3, s_3)$
  - $s3 = s - M[3, IM[3, 1]] + M[3, IM[3, 2]] = 2.5 - 0.8 + 0.3 = 2.0$ (15)

4. Insert the expanded nodes into the tree organized by sum value, as shown in Fig. 6, individually as follows:
5. Add vector $V_2 = (2, 1, 1)$, which corresponds to Top in the tree in Fig. 6 shown in SM presented in Eq. (16), as the vector with the highest sum to row 2 in matrix R, as presented in Eq. (17).

$$SM = \begin{bmatrix} 0.9 & \mathbf{0.8} & 0.7 & 0.3 \\ \mathbf{0.8} & 0.5 & 0.4 & 0.2 \\ \mathbf{0.8} & 0.3 & 0.1 & 0.0 \end{bmatrix} \tag{16}$$

$$R = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 1 \\ - & - & - \\ - & - & - \end{bmatrix} \tag{17}$$

6. Store Top in Temp and delete Top. Then expand Temp to the following nodes and calculate the sum values as presented in Eqs. (18), (19), and (20).

- node $(2, 1, s_1)$
  - $s1 = s - M[1, IM[1, 2]] + M[1, IM[1, 3]] = 2.4 - 0.8 + 0.7 = 2.3$ (18)
- node $(2, 2, s_2)$
  - $s2 = s - M[2, IM[2, 1]] + M[2, IM[2, 2]] = 2.4 - 0.8 + 0.5 = 2.1$ (19)
- node $(2, 3, s_3)$
  - $s3 = s - M[3, IM[3, 1]] + M[3, IM[3, 2]] = 2.4 - 0.8 + 0.3 = 1.9.$ (20)

7. Insert the expanded nodes organized by sum value into the tree, as shown in Fig. 7, individually, keeping in mind that k = 4, and because node (2, 3, 1.9) sum is less than the worst node, it is not inserted into the graph.
8. Add vector $V_3 = (3, 1, 1)$ as shown in SM presented in Eq. (21) which corresponds to Top in the tree presented in Fig. 7, to matrix R as the third row presented in Eq. (22).

$$SM = \begin{bmatrix} 0.9 & 0.8 & \mathbf{0.7} & 0.3 \\ \mathbf{0.8} & 0.5 & 0.4 & 0.2 \\ \mathbf{0.8} & 0.3 & 0.1 & 0.0 \end{bmatrix} \tag{21}$$

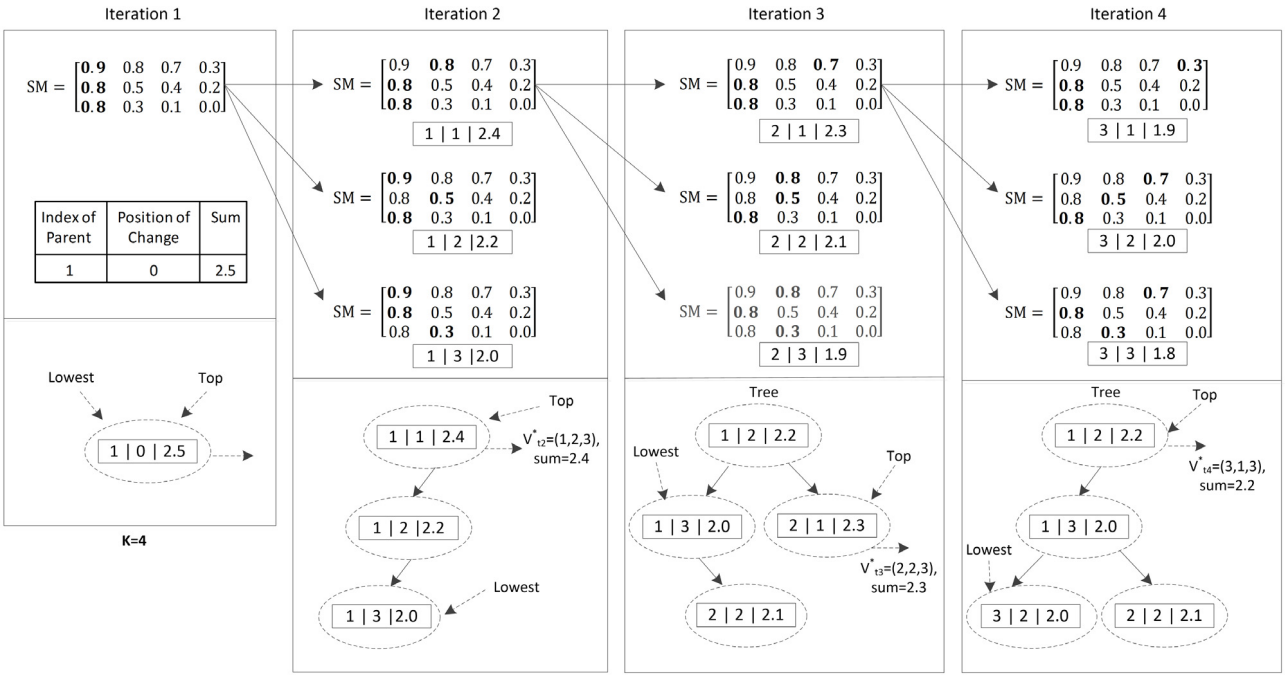


Fig. 4. Building the tree from the sorted matrix.

$$R = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 1 \\ 3 & 1 & 1 \\ - & - & - \end{bmatrix} \tag{22}$$

9. Store Top in Temp and delete Top. Then expand Temp as shown in Fig. 8 to the following nodes:
   - node $(3, 1, s_1)$
     - $s1 = s - M[1, IM[1, 3]] + M[1, IM[1, 4]]$
       $= 2.3 - 0.7 + 0.3 = 1.9$ (23)
   - node $(3, 2, s_2)$
     - $s2 = s - M[2, IM[2, 1]] + M[2, IM[2, 2]]$
       $= 2.3 - 0.8 + 0.5 = 2.0$ (24)
   - node $(3, 3, s_3)$
     - $s3 = s - M[3, IM[3, 1]] + M[3, IM[3, 2]]$
       $= 2.3 - 0.8 + 0.3 = 1.8.$ (25)
10. Insert the expanded nodes with the highest sum into the tree individually, as shown in Fig. 8, maintaining the size of the tree, $k = 4$. Therefore, node $(3, 1, 1.9)$ presented in Eq. (23) is inserted into the proper position and then node $(3, 2, 2.0)$ presented in Eq. (24) replaces it because it is the node with the lowest sum, and then node $(3, 3, 1.8)$ presented in Eq. (25) is not inserted.
11. Add vector $V_4 = (1, 2, 1)$, as shown in SM presented in Eq. (26) which corresponds to Top in the shown in Fig. 8, to row 4 in matrix R, as shown in Eq. (27).

$$SM = \begin{bmatrix} \mathbf{0.9} & 0.8 & 0.7 & 0.3 \\ 0.8 & \mathbf{0.5} & 0.4 & 0.2 \\ \mathbf{0.8} & 0.3 & 0.1 & 0.0 \end{bmatrix} \tag{26}$$

$$R = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 1 \\ 3 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix} \tag{27}$$

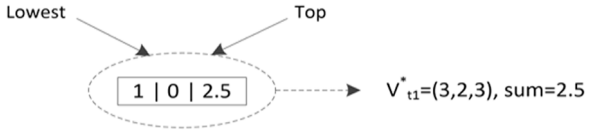


Fig. 5. First node of the tree.

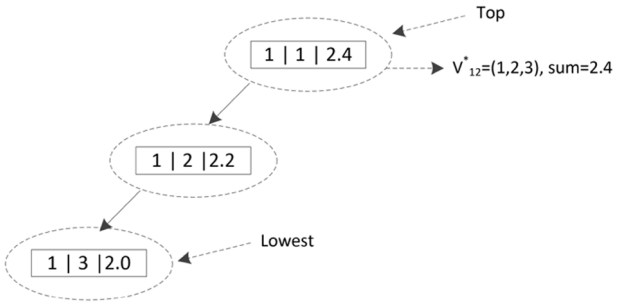


Fig. 6. Tree expansion (1).

The merit of our tree is that the top and lowest nodes can be identified easily, where the top is the rightmost node and the lowest is the leftmost node. Two pointers are used to store the addresses of these two nodes.

8. Experiment and experimental results

We performed our experiment on a Dell personal computer with Intel Core i7-3770, 3.40 GHz CPU, and 32GB RAM. The computer ran Java Virtual Machine 1.8, and Microsoft Windows 8 with a 64-bit operating system. We evaluated the efficiency and effectiveness of our proposed algorithm and compared them with the results presented in [7] and [8]. Next, we explain the dataset used in our experiment.

Pseudocode of the Semantic Matching Algorithm Using Tree Structure (SMT)

Algorithm: Identify Top-k similarity vectors

Input: M, k.
M : Matrix of n rows and m attributes of similarity values between zero and one.
k : Required number of the best vectors.
Output: Best k vectors of n dimensions.

Begin

$IM \leftarrow$ sort(M) //Sort M and retrieve corresponding sorted indices matrix (IM)
Tree ← null //Create an empty tree and assign null to the root of the tree
$R_1 \leftarrow (1, 1, 1, \ldots, 1)$ //Assign the first best vector
$s_1 \leftarrow 0$ // Compute the sum similarity values for the first best vector
For i←1, 2, …, n
  begin
    $s_1 \leftarrow s_1 + M\,[i, IM[i, R_1[i]]]$
End
Create new node p
p.t←1, p.c←0, p.s←s_1 // Parent index (p.t), position of change (p.c), sum (p.s)
Top ← Tree, Lowest ← Tree // Assign pointers to top and lowest values
Insert (Tree, p, Top, Lower)
TSize ←1
/----------Expand V_1 to n vectors----------/
For q←1, 2, …, k
 begin
  $TTemp.t \leftarrow (Top.t), Temp.c \leftarrow (Top.c), Temp.s \leftarrow (Top.s)$
  Print $(Temp\_V_t^*, Temp.s)$ // $Temp\_V_t^* = (IM[1, R_{Temp\_t}[1]], \ldots, IM[n, R_{Temp\_t}[n]])$
  Delete (Tree, Top) // With updating Top and Lowest
  For j ←1, 2, …, n
   begin
    Create new node P
    p.t←q, p .c←j
    p. s← $s_1 - M\left[j, IM\left[j, R\,[q][j]\right]\right] + M\left[j, IM[j, (R[q][j]) + 1\right]$
    if (TSize < k)
      insert (Tree, p, Top, Lowest) //With updating Top and Lowest
      TSize ←TSize + 1
    elseif (p.s > Lower.s)
      insert (Tree, p, Top, Lowest) // Replace Lower by p
    else
      discard p //no insertion
    end{if}
   end{for}
  $R_{q+1} \leftarrow R_{Top.t}$, $R_{q+1}\,[Top.c] \leftarrow R_{q+1}\,[Top.c] + 1$ //assign q+1 best vector.
end{for}

End

8.1. Dataset

We chose a publicly available dataset [22] that mostly simulated the heterogeneous nature of the IoT environment and was used in [8]. The dataset consisted of events similar to those produced by the SmartSantander Smart City Project [28] and Linked Energy Intelligence (LEI) [29] data space. They are smart projects that focus on traffic, parking vacancies, speed, environmental metrics, energy saving and consumption.

Seed Events and Exact Subscriptions

A set of 166 seed events were randomly chosen from the datasets. The exact subscriptions for the seed events were assigned as a match to generate the ground truth:

{category = ground water flow slight drop event,
suburban area = galway}.

Event Set Semantic Expansion and Approximate Subscriptions

Based on the EuroVoc thesaurus [30], the events were semantically expanded by replacing terms with their synonyms or other related terms. Below we present an expanded example based on the above seed event is

{category = ground water flow drop event,
continent = european countries,
measurement scale = cubic meters per second,
suburb = galway, country = ireland}.

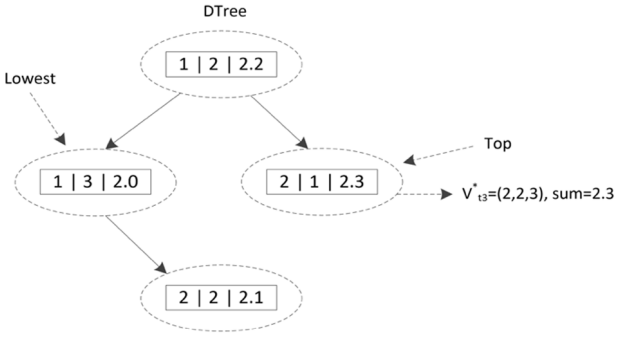


Fig. 7. Tree expansion (2).

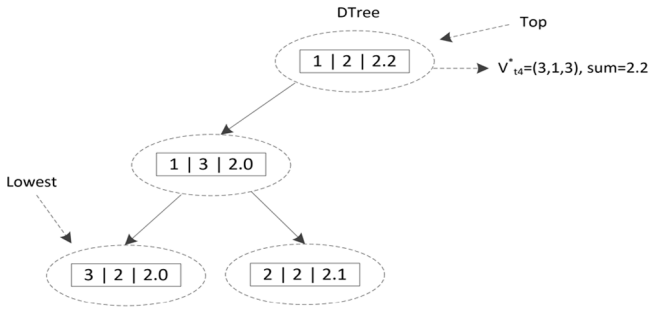


Fig. 8. Tree expansion (3).

A set of 94 subscriptions were approximated by appending the tilde operator (∼) to the attribute and/or value. Below is an example

{category ∼= ground water flow slight drop event ∼,
suburban area ∼= galway ∼}.

Matching Based on the Relevance Ground Truth

For the relevance ground truth, approximate subscriptions were matched first to exact match events, and then to events from the expanded set with similar or related terms. We discarded the theme tags appended to events and subscriptions in the original dataset. There are many other works where IoT dataset are used for various purposes [31–40].

8.2. Efficiency

To test for efficiency, we performed SMT with a varying number of events tuples, subscription predicates, and number of desired nearest matches, represented by m, n, and k, respectively. As shown in Fig. 9, we observe that the time needed for a larger number of required matches improved when we used SMT compared with the two previous algorithms. However, the improvement was substantial when increasing subscription predicates or number of events because the time increase was linear with increasing n, as shown in Fig. 10, or m, as shown in Fig. 11, compared with the exponential time increase in [7] or [8].

8.3. Effectiveness

To test the effectiveness, we tested the precision, recall, and F-score of SMT, TC-SMT and compared them with the algorithms from earlier work presented in [7] and [8]. The comparisons of the precision, recall, and F-score for all four algorithms are shown in Figs. 12, 13, and 14 respectively. TC-SMT demonstrated a clear improvement in terms of the precision values, as shown in Fig. 12, because of clustering, which limited the choices to events within a similar taxonomy. The recall values shown in Fig. 13 exhibited similar results for all tested algorithms. The F-score values in Fig. 14 showed a significant increase, particularly when the subscriptions were fully approximated. These results demonstrated the benefits of TC for the precision of the returned results, which consequently affected the F-score. The benefits were more apparent when the subscriptions were fully approximated and uncertainty increased.

9. Discussion

The proposed algorithm, SMT, was more efficient because of the following reasons:

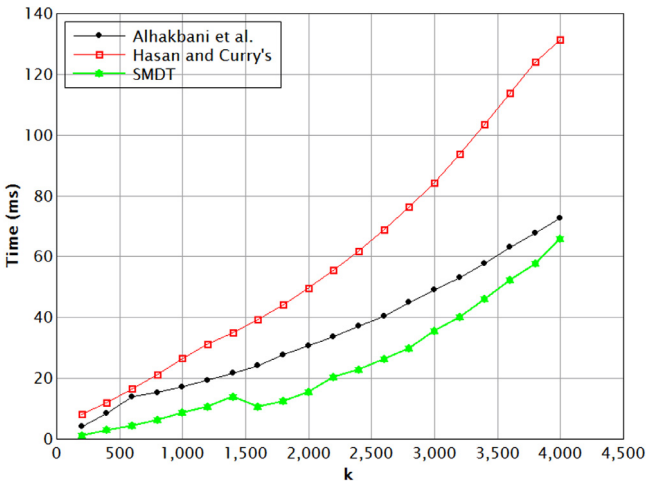


Fig. 9. Number of matches vs. time.

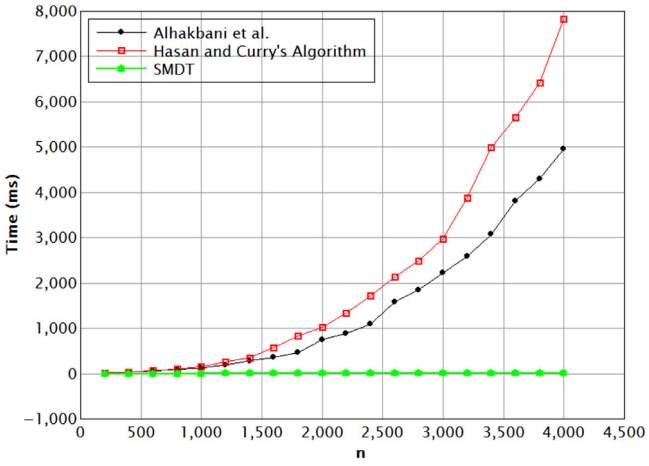


Fig. 10. Number of subscription predicates vs. time.

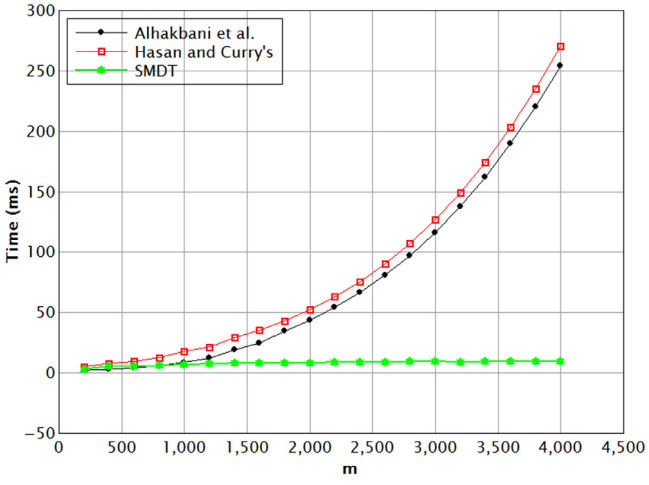


Fig. 11. Number of event tuples vs. time.

1. It used an optimized data structure, a limited sized tree of size only k nodes, where each node consisted of three values: index of parent vector, position of change in the parent vector, and sum of similarity values. Additionally, we used array R of size kn to store the best top k vectors. Thus,

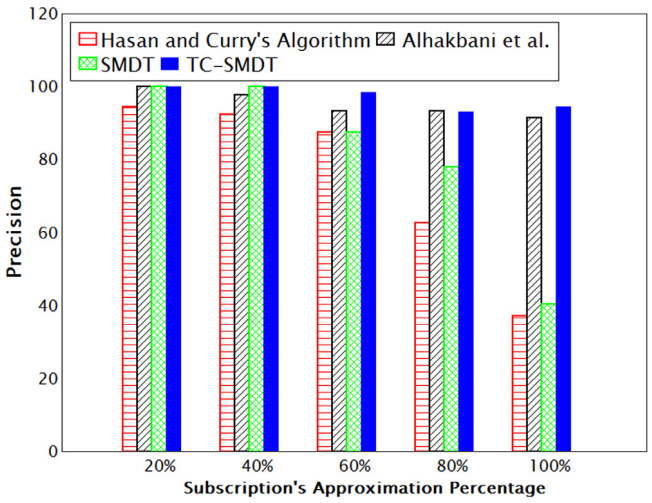


Fig. 12. Comparison of precision values.

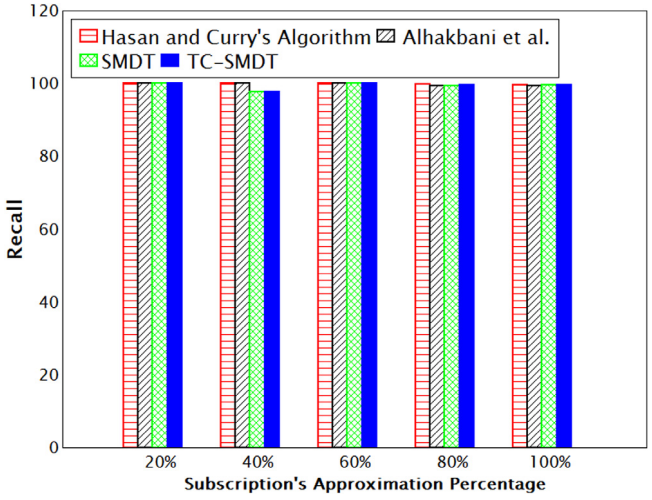


Fig. 13. Comparison of recall values.

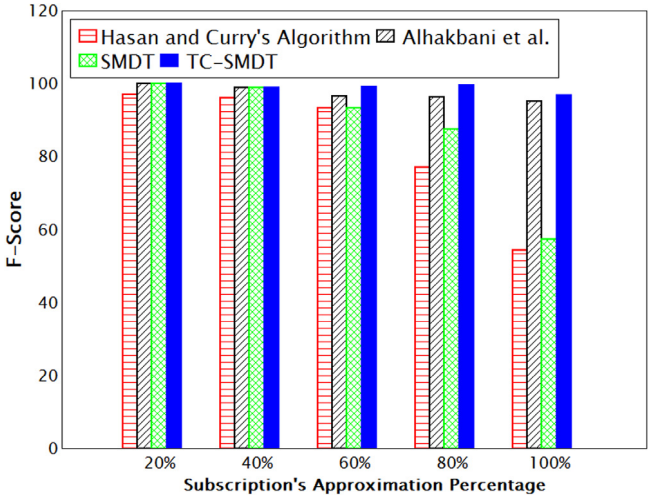


Fig. 14. Comparison of F-Score values.

the total space complexity, excluding the size of the input similarity matrix and sorted indices matrix, is presented in Eq. (28).

$$3k + kn = O(nk + k). \tag{28}$$

2. It demonstrated efficient expansion from the current top vector (parent vector), where it was not necessary to re-sum the similarity vectors each time or pass the entire offspring vectors to the tree. This was performed as follows: Let M be the unsorted matrix of size nm (n rows and m attributes), IM be the sorted index matrix of size nm, and $R_t = (a_{t1}, a_{t2}, \ldots, a_{tn})$ be the current top vector of array R or parent vector with total similarity value s. Each node is a vector of three values (t, c, s), where t is the index of the parent vector in array R, c is the position of change in parent vector t, and s is the sum of similarity vector t. This top node can be expanded to n nodes as presented in Eqs. (29), (30) and (31):

$$V_{t1} = (t, 1, s_1) \text{ and } s_1 = s - M[1, IM[1, a_{t1}]] + M[1, IM[1, a_{t1} + 1]] \tag{29}$$

$$V_{t2} = (t, 2, s_2) \text{ and } s_2 = s - M[2, IM[2, a_{t2}]] + M[2, IM[2, a_{t2} + 1]] \tag{30}$$

$$\ldots$$

$$V_{tn} = (t, n, s_n) \text{ and } s_n = s - M[n, IM[n, a_{tn}]] + M[n, IM[n, a_{tn} + 1]]. \tag{31}$$

These offspring nodes were inserted into the tree in the form (t, c, s). Note that, $V_{tj} = (t, j, s_j)$ was mapped to $(a_{t1}, \ldots, a_{tj} + 1, \ldots, a_{tn})$ and $R_1 = (1, 1, \ldots, 1)$. This expansion was performed if the following condition in Eq. (32) was satisfied:

$$a_{ti} + 1 \leq m \tag{32}$$

Total time complexity analysis

- First, we sorted the similarity matrix and retrieved the sorted indices matrix, which costed O (nmlogm) , using an average case quick sort algorithm or worst case merge sort.
- Second, we inserted n nodes for $(k - 1)$ iterations, and the insertion of each node cost O(logk) in the tree of size k and average height $\lfloor \log k \rfloor + 1$.
- Note that, at iteration 1, we had only one node, that is, the top vector. Thus, the total time complexity is presented in Eq. (33)

$$O(nmlogm + n(k - 1)(logk)) = O(nmlogm + nklogk). \tag{33}$$

10. Conclusions

The ubiquitous nature and scale of IoT applications has raised the need for efficient and effective communication among them. Approximate semantic matching is a better option compared with exact syntactic matching; however, achieving a high throughput given a large scale is a challenge. The proposed model, SMT, accomplished linear performance time with n, and m. TC-SMT achieved more than a 95% F-score for effectiveness given subscriptions that required a 100% degree of approximation. This model can contribute to the middleware layer, for which applications are highly critical in terms of time, and with no prior knowledge of event semantics. In future, we will extend our work in distributed environment such as cloud platform to implement parallel processing of event and subscriptions.

Acknowledgment

The authors would like to extend their sincere appreciation to the Deanship of Scientific Research at King Saud University for its funding of this research through the research group project no. RGP-281.

References

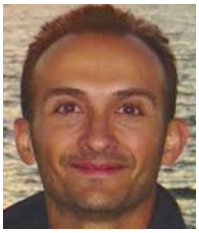
[1] Gartner Says 8.4 Billion Connected & quot;Things & quot; Will Be in Use in 2017, Up 31 Percent From 2016, Gartner, Inc. 2017. [Online]. Available: http://www.gartner.com/newsroom/id/3598917 [Accessed: 19-Sep-2017].
[2] J. Mineraud, O. Mazhelis, X. Su, S. Tarkoma, A gap analysis of internet-of-things platforms, Comput. Commun. 89–90 (2016) 5–16.
[3] S. Hasan, E. Curry, Thingsonomy: tackling variety in internet of things events, IEEE Internet Comput. 19 (2) (2015) 10–18.
[4] M. Pelino, F.E. Gillett, C. Voce, C. Mines, P. Matzke, M. Mai, The Internet Of Things Heat Map, 2016 Where IoT Will Have The Biggest Impact On Digital Business, Forrester, 2016. [Online]. Available: https://www.forrester.com/report/The+Internet+Of+Things+Heat+Map+2016/-/E-RES122661 [Accessed: 19-Sep-2017].
[5] S. Patel, S. Jardosh, A. Makwana, A. Thakkar, Publish/Subscribe mechanism for IoT: A survey of event matching algorithms and open research challenges, in: Advances in Intelligent Systems and Computing (AISC), 2017, pp. 287–294.
[6] T. Wang, X. Tao, A distributed semantic filtering model based on approximate automata for heterogeneous multi-sensor networks Yucai Zhou Pengcheng Li and Chunhui Zhao, Int. J. Sens. Networks 20 (1) (2016) 46–53.
[7] S. Hasan, E. Curry, Approximate semantic matching of events for the internet of things, ACM Trans. Internet Technol. 14 (1) (2014).
[8] N. Alhakbani, M.M. Hassan, M. Ykhlef, An effective semantic event matching system in the internet of things (IoT) Environment, Sensors 17 (9) (2014) 2017.
[9] M. Petrovic, I. Burcea, H.-A. Jacobsen, S-ToPSS : semantic toronto publish / subscribe system, in: VLDB '03 Proceedings of the 29th international conference on Very large data bases, 2003, pp. 1101–1104.
[10] J. Wang, B. Jin, J. Li, An ontology-based publish/subscribe system, in: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, 2004, pp. 232–253.
[11] L. Zeng, H. Lei, A semantic publish / subscribe system, in: IEEE International Conference on E-commerce Technology for Dynamic E-Business, 2004, pp. 32–39.
[12] W.Z.W. Zhang, J.M.J. Ma, D.Y.D. Ye, FOMatch: a fuzzy ontology-based semantic matching algorithm of publish/subscribe systems, in: 2008 International Conference on Computational Intelligence for Modelling Control & Automation, 2008, pp. 111–117.
[13] S. Wenting, X. Xiaoping, W. Xiaoping, An ontology-based event matching dealing with semantic heterogeneity in Pub/Sub systems, in: Proceedings of 2009 4th International Conference on Computer Science and Education, ICCSE 2009, 2009, pp. 1225–1230.
[14] D. Preuveneers, Y. Berbers, μC-SemPS: Energy-efficient semantic publish/subscribe for battery-powered systems, in: Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng. LNICST, vol. 73 LNICST, no. 1, 2012, pp. 39–50.
[15] N. Ahmed, J. Bryant, G. Hasseler, M. Paulini, Enabling semantic technologies in publish and subscribe middleware, in: International Conference on Semantic Computing IEEE ICSC, 2015, pp. 338–343.
[16] H. Kim, S. Kang, S. Oh, Ontology-based quantitative similarity metric for event matching in publish/subscribe system, Neurocomputing 152 (2015) 77–84.
[17] F. Gao, M.I. Ali, E. Curry, A. Mileo, Automated discovery and integration of semantic urban data streams: The ACEIS middleware, Futur. Gener. Comput. Syst. 76 (2017) 561–581.
[18] H. Liu, H.-A. Jacobsen, A-TOPSS: a publish/subscribe system supporting approximate matching, in: Proceedings of the 28th International Conference on Very Large Data Bases, 2002, pp. 1107–1110.
[19] S. Hasan, S. O'Riain, E. Curry, Approximate semantic matching of heterogeneous events, in: 6th ACM Int. Conf. Distrib. Event-Based Syst. (DEBS 2012), 2012, pp. 252–263.
[20] S. Hasan, Loose Coupling in Heterogeneous Event-Based Systems via Approximate Semantic Matching and Dynamic Enrichment, National University of Ireland, Galway, 2016.
[21] Y. Qin, L. Yao, Q.Z. Sheng, Approximate semantic matching over linked data streams, in: LNCS, Vol. 9828, 2016, pp. 37–51.
[22] S. Hasan, E. Curry, Thematic event processing dataset, 2014. [Online]. Available: https://www.researchgate.net/publication/263673956_Thematic_event_processing_dataset [Accessed: 28-Feb-2017].
[23] S. Hasan, E. Curry, Thematic Event Processing, in: Middleware, 2014, pp. 109–120.
[24] M. Serrano, P. Barnaghi, F. Carrez, P. Cousin, O. Vermesan, P. Friess, IoT semantic interoperability:research challenges, best practices, recommendationsand next steps, in: Eur. Res. Clust. Internet Things IERC, 2015.
[25] Avigdor Gal, Managing Uncertainty in Schema Matching with Top-k Schema Mappings, Berlin: Springer-Verlag, 2006.
[26] E. Gabrilovich, S. Markovitch, Computing semantic relatedness using wikipedia-based explicit semantic analysis, in: IJCAI Int. Jt. Conf. Artif. Intell, 2007, pp. 1606–1611.
[27] Apache Lucene - Welcome to Apache Lucene. [Online]. Available: https://lucene.apache.org/ [Accessed: 05-Mar-2017].
[28] L. Sanchez, J.A. Galache, V. Gutierrez, J.M. Hernandez, J. Bernat, A. Gluhak, T. Garcia, SmartSantander: the meeting point between future internet research and experimentation and the smart cities, in: Future Network & MobileSummit Conference Proceedings, 2011, pp. 1–8.
[29] E. Curry, S. Hasan, S.O. 'riain, Enterprise energy management using a linked dataspace for energy intelligence, in: Second IFIP Conference on Sustainable Internet and ICT for Sustainability, 2012.
[30] EuroVoc. [Online]. Available: http://eurovoc.europa.eu/drupal/.
[31] M.Z.A. Bhuiyan, G. Wang, J. Wu, J. Cao, X. Liu, T. Wang, Dependable structural health monitoring using wireless sensor networks, IEEE Trans. Dependable Secure Comput. 14 (4) (2017) 363–376.
[32] M.Z.A. Bhuiyan, G. Wang, A.V. Vasilakos, Local area prediction-based mobile target tracking in wireless sensor networks, IEEE Trans. Comput. 64 (7) (2015) 1968–1982.
[33] E. Luo, M.Z.A. Bhuiyan, G. Wang, M.A. Rahman, J. Wu, M. Atiquzzaman, PrivacyProtector: privacy-protected patient data collection in IoT-Based Healthcare Systems, IEEE Commun. Mag. 56 (2) (2018) 163–168.
[34] M.Z.A. Bhuiyan, J. Wu, G.M. Weiss, T. Hayajneh, T. Wang, G. Wang, Event detection through differential pattern mining in cyber-physical systems, IEEE Trans. Big Data (2017) http://dx.doi.org/10.1109/TBDATA.2017.2731838, Published online.
[35] G. Fortino, W. Russo, C. Savaglio, W. Shen, M. Zhou, M, Agent-oriented cooperative smart objects: From IoT system design to implementation, IEEE Trans. Syst. Man. Cybern.: Syst. (99) (2017) 1–18.
[36] M.G.R. Alam, M.M. Hassan, M.Z. Uddin, A. Almogren, G. Fortino, Autonomic computation offloading in mobile edge for IoT applications, Future Gener. Comput. Syst. 90 (2019) 149–157.
[37] L. Yang, W. Li, M. Ghandehari, G. Fortino, People-centric cognitive internet of things for the quantitative analysis of environmental exposure, IEEE Internet Things J. 5 (4) (2018) 2353–2366.
[38] R. Casadei, G. Fortino, D. Pianini, W. Russo, C. Savaglio, M. Viroli, Modelling and simulation of opportunistic IoT services with aggregate computing, Future Gener. Comput. Syst. Online First (2018).
[39] G. Aloi, G. Caliciuri, G. Fortino, R. Gravina, P. Pace, W. Russo, C. Savaglio, Enabling IoT interoperability through opportunistic smartphone-based mobile gateways, J. Netw. Comput. Appl. 81 (2017) 74–84.
[40] W.N. Ismail, M.M. Hassan, H.A. Alsalamah, Mining of productive periodic-frequent patterns for IoT data analytics, Future Gener. Comput. Syst. 88 (2018) 512–523.

Noura Alhakbani received the M.S. degree in computer science from King Saud University, Riyadh, Saudi Arabia in 2008. She got her Ph.D. degree in Information Systems at King Saud University in 2018. She is now an Assistant Professor at the Information Technology Department at King Saud University. Her research interest includes IoT, sensor communication, matching algorithms, and Smart Cities.

Mohammad Mehedi Hassan is currently an Associate Professor of Information Systems Department in the College of Computer and Information Sciences (CCIS), King Saud University (KSU), Riyadh, Kingdom of Saudi Arabia. He received his Ph.D. degree in Computer Engineering from Kyung Hee University, South Korea in February 2011. He received Best Journal Paper Award from IEEE Systems Journal in 2018. He also received Best Paper Award from CloudComp conference at China in 2014. He received Excellence in Research Award from CCIS, KSU in 2015 and 2016 respectively. He has published over 130+

research papers in the ISI-Indexed journals of international repute. He has also played role of the guest editor of several international ISI-indexed journals such as IEEE IoT Journal, Future Generation Computer Systems etc. He is currently an Associate Editor of IEEE Access Journal. His research areas of interest are cloud federation, multimedia cloud, sensor-cloud, Internet of things, Big data, mobile cloud, sensor network, publish/subscribe system and recommender system. He is a member of IEEE.

Mourad Ykhlef got his MS in artificial intelligence and PhD in Computer Science from Paris and Bordeaux, France, respectively. He is a professor in the Department of Information Systems, College of Computer and Information, Systems, King Saud University, Saudi Arabia. His main research interests include data mining and bio-inspired computing.



Giancarlo Fortino a Full Professor of Computer Engineering at the Dept. of Informatics, Modeling, Electronics, and Systems of the University of Calabria (Unical), Italy. He received a Ph.D. in Computer Engineering from Unical in 2000. He is also guest professor at Wuhan University of Technology (Wuhan, China), high-end expert at HUST (China), and senior research fellow at the Italian National Research Council ICAR Institute. He is the director of the SPEME lab at Unical as well as co-chair of Joint labs on IoT established between Unical and WUT and SMU Chinese universities, respectively. His research interests include agent-based computing, wireless (body) sensor networks, and Internet of Things. He is author of over 400 papers in int'l journals, conferences and books. He is (founding) series editor of IEEE Press Book Series on Human-Machine Systems and EiC of Springer Internet of Things series and AE of many int'l journals such as IEEE TAC, IEEE THMS, IEEE IoTJ, IEEE SJ, IEEE SMCM, Information Fusion, JNCA, EAAI, etc. He is cofounder and CEO of SenSysCal S.r.l., a Unical spinoff focused on innovative IoT systems. Fortino is currently member of the IEEE SMCS BoG and of the IEEE Press BoG, and chair of the IEEE SMCS Italian Chapter.