Towards a Window-based Diverse Entity Summarisation Engine in Publish/Subscribe Systems

Niki Pavlopoulou
Insight Centre for Data Analytics
National University of Ireland Galway
Galway, Ireland
niki.pavlopoulou@insight-centre.org

Edward Curry
Insight Centre for Data Analytics
National University of Ireland Galway
Galway, Ireland
edward.curry@insight-centre.org

ABSTRACT

The rise of Smart Homes, Smart Cities and Internet of Things results in the creation of a wide range of entity-based data streams and users interested in the real-time analysis of these streams. These smart environments possess characteristics, like dynamism, continuity, heterogeneity and high volume of data and users. A suitable data dissemination paradigm is needed that can overcome these challenges and at the same time, provide expressive notifications to users, but not at the expense of usability or resources. Publish/Subscribe systems can efficiently realise some of these requirements; however, they need additional support when applied in smart environments to overcome assumptions related to usability and redundancy-awareness. Therefore, the key question of the paper is: *Can we define an entity-centric Publish/Subscribe system that provides expressive user notifications along with high usability and limited resource usage?*

In this work, we explore this question and propose a Publish/Subscribe system with windowing, data fusion, and top-k diverse ranking that can result in the creation of expressive entity summaries using limited resources. Our results show that sending a top-k fused diverse summary as a notification is better than sending all the separate notifications or the fused ones without top-k filtering. Specifically, top-k fused diverse summarisation results in 50% to 80% reduction of forwarded messages and redundancy-awareness with an F-score ranging from 0.35 to 0.73 depending on the k. Nevertheless, these results are achieved at the expense of a slightly higher latency; therefore, there is some trade-off between latency, the number of forwarded messages, and expressiveness.

KEYWORDS

Publish/Subscribe Systems, Data Fusion, Diversity, Entity Summarisation, RDF Graphs

1 INTRODUCTION

The emergence of Smart Homes, Smart Cities and Internet of Things has resulted in multiple sensors (producers) that create a wide range of entity-based data streams and multiple users or applications (consumers) that are interested in the analysis of these data streams for various needs. These needs could vary from safety (e.g. the tracking of the spread of wildfire in forests or seismic measurements [1]) to the environment (e.g. environmental and climatological monitoring where phenomena such as temperature, pressure and humidity are measured periodically [14]).

These entity-based data streams present a high level of data heterogeneity [19], either schematic or semantic, as well as duplication. For example, semantics could involve the use of different words describing conceptually similar things. Duplicates and conceptually similar things result in redundant information. On the other hand, data consumers may have different levels of expressibility. Expressibility [8] refers to users' level of prior understanding of their needs to create queries with specific filters or their level of technical ability to use complex query languages. Sometimes a user might find it difficult to create an appropriate query or one might need to create separate complex queries or join queries to bring together the information needed from multiple sources.

The challenges above when combined with dynamism (deletion or addition of data producers or consumers), continuity (unbounded data streams) and the high volume of producers and consumers, characteristics that exist in smart environments, may result in inefficient and ineffective processing of streaming data. Specifically, high data volume and redundancy may lead to significant propagation, as well as storage overheads of unnecessary data within a network and slower processing time [2]. At the same time, low user expressibility may lead to abstract user queries that result in redundant answers and high volumes that might present the user with unnecessary information [25].

Publish/Subscribe systems provide a suitable interaction scheme for dynamic large-scale applications, where subscribers (users) express their interest in an event or pattern of events, and they are notified when a suitable event was generated by a publisher [7]. These systems are characterised by space decoupling (publishers and subscribers do not need to know each other), time decoupling (publishers and subscribers do not need to be active at the same time) and synchronisation decoupling (publishers are not blocked during event production, and subscribers can be notified while performing another activity).

Nevertheless, Publish/Subscribe systems cannot cope with the challenges of redundancy-awareness and the need for high usability as defined above. For example, if a subscriber is interested in an entity, then existing Publish/Subscribe systems assume that: 1) Subscribers are experts in query languages to perform specific filtering queries, 2) Subscribers are aware of the publication semantics and format so performing semantic-specific and schematic-specific queries would necessarily lead to matches, 3) Publications are mostly seen as separate pieces of information for ranking, without considering that by fusing and ranking these pieces may lead to better notifications without redundancy.

Therefore, the key question of the paper is: *Can we define an entity-centric Publish/Subscribe system that provides expressive (non-redundant) user notifications along with high usability (no assumption of high user expressibility) while using limited resources?*

To address the key question above, we propose in this paper a window-based diverse entity summarisation engine in Publish/Subscribe systems, as approximate solutions [14] are acceptable as quick answers [1] within a small error range with high probability while using limited resources. These summaries, when derived from the fusion of multiple publishers that contain complex entity-based semantic data and when combined with diversity (and not only relatedness) will result in expressive subscription notifications. Nevertheless, there is a trade-off between latency, number of forwarded messages, and expressiveness, which we also examine.

2 PROBLEM ANALYSIS

The problem introduced is analysed more below.

2.1 Motivational Scenario

Sensors create a high amount of data streams with frequent sampling rates. Therefore, they might produce many unchanged or identical values for a period of time [15]. When users create abstract, non-sophisticated queries to gain knowledge on these data streams, they might be presented with undesired duplication.

For example, imagine Houston is a smart city, and a user is interested in information concerning *Rice University*. The user has no other information apart from the name of the university, and one needs to gain more knowledge without exactly knowing what one is looking for. A wide range of sensor readings contain information about the university, ranging from the temperature of the university to the city it is located in, which some might be redundant. The user would like to quickly gain knowledge about the university, but not to be overwhelmed, especially with duplicate data. This scenario is illustrated in Fig. 1.

2.2 Problem Challenges

The aforementioned motivational scenario faces a number of challenges:

- **Redundancy awareness**: Multiple publishers create heterogeneous data about the entity. Some of this data, like *temperature* and *city* results in redundant information due to duplication.
- **Low user expressibility**: The user has limited information about the entity and has no prior knowledge of what they are looking for. The user is unable to create a complex filtering query and is not an expert in query languages. For example, a SPARQL-like query that notifies the user when the energy usage exceeds 4kWh would be the following:

  ```
  SELECT ?energy_value
  FROM STREAM
  WHERE {
  Rice_University energy_usage ?energy_value;
  FILTER (?energy_value > 4kWh).
  }
  ```

  This query assumes a priori knowledge from the user of the publication semantics and schematics concerning "energy_usage" instead of synonyms like "energy_consumption", "kWh" instead of "Wh" or which stream or streams produce energy usage readings. On the other hand, if the user creates an abstract query like the keyword-based one "Rice University", it may lead to redundant or undesired information.

3 BACKGROUND

Some concepts and definitions concerning knowledge graphs, entity summarisation, and Publish/Subscribe systems are described below.

3.1 Knowledge graphs and Entity Summarisation

Knowledge graphs contain information regarding entities, which are real-world or abstract things [20]. Within knowledge graphs the nodes represent the entities, and the directed labelled arcs constitute relations among them. In Fig. 2 a part of the knowledge graph is represented that supports the motivational scenario, where *Rice University*, *15°C*, *5kWh*, *United States*, *Houston*, *Texas* and *Division I (NCAA)* are entities or literals and *temperature*, *energyUsage*, *country*, *city*, *state* and *athletics* are relations among the connected entities or literals by the directed arc. The Resource Description Framework (RDF) is a data modelling language that represents these representations as triples <subject, property, object>, where subject are entities, object are entities or literals and property is their relation. RDF triples with the same subject form an RDF star-like graph.

In knowledge graphs, though, there might be some redundant information. This could be addressed by summarising the triples of an entity. A summarisation of an entity e that is represented by a node v in a knowledge graph G is a subgraph of G that surrounds v [20].

By adopting and adapting definitions that were introduced in Cheng et al. [5], we provide some definitions for completeness.

Let E be the set of all entities, L the set of all literals and P the set of all properties.

Definition 1 (Data Graph). A data graph is a digraph $G = \langle V, A, Lbl_V, Lbl_A \rangle$, where V is a finite set of nodes, A is a finite set of directed edges where each $a \in A$ has a source node $Src(a) \in V$ and a target node $Tgt(a) \in V$, and $Lbl_V : V \mapsto E \cup L$ and $Lbl_A : A \mapsto P$ are labeling functions that map nodes and edges to entities or literals, and properties, respectively.

Definition 2 (Triple). A triple tr is a sequence of <subject, property, object> defined as $tr = \langle sub(tr), p(tr), obj(tr) \rangle$, where $sub(tr) \in E$, $p(tr) \in P$ and $obj(tr) \in E \cup L$.

Definition 3 (Triple Set). Given a data graph G, the triple set of an entity e, denoted by $Tr(e)$, is the set of all unique triples of e that can be found in G.

Definition 4 (Diverse Entity Summarisation). Given $Tr(e)$ and a positive integer $k < |Tr(e)|$, the problem of diverse entity summarisation is to select $DivSumm(e) \subset Tr(e)$ such that $|DivSumm(e)| = k$. $DivSumm(e)$ is called a diverse summary of e and it contains a set of unique triples.

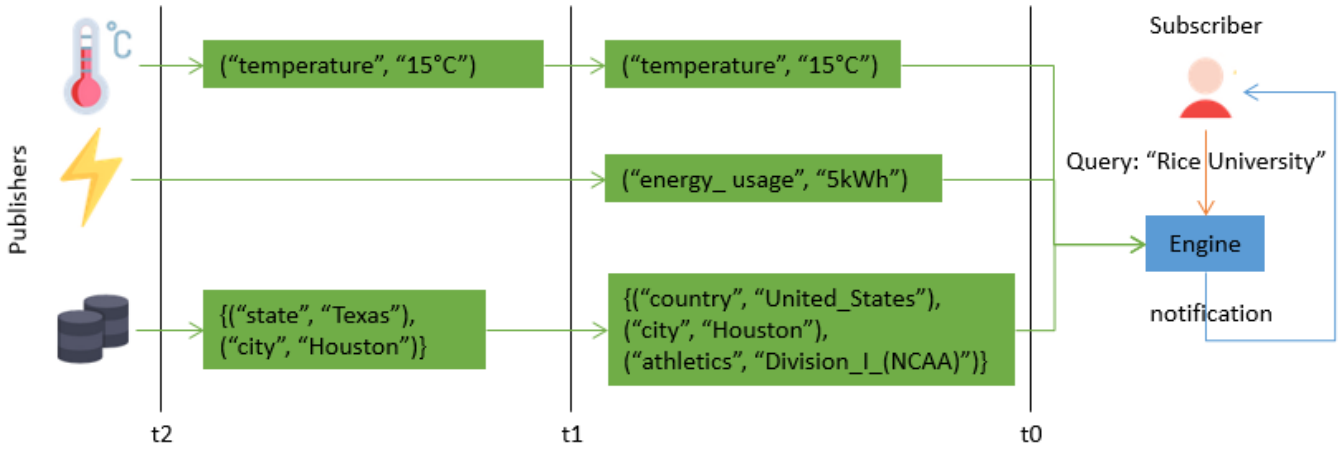

Figure 1: A subscriber is interested in information about *Rice University* and publishers publish timestamped information records about it. In this example, the *temperature* and *city* information is duplicate between timestamps t_2 and t_1.

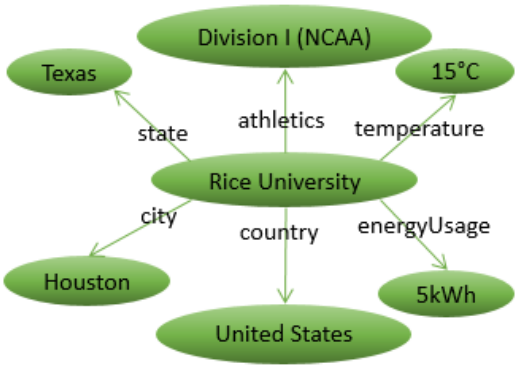

Figure 2: Part of a knowledge graph of *Rice University*

3.2 Publish/Subscribe Systems

In a typical Publish/Subscribe system [7], subscribers could be from users to applications that they subscribe their interest in an event or pattern of events. These subscriptions are sent to the Event Engine where they are stored. Publishers could be sensors, users or applications generating events or publications and sending them to the Event Engine. A matcher is contained in the engine that matches specific events to subscriptions based on their conditions. When this is happening, the subscribers are getting these events as notifications. Its decoupling capabilities in space, time and synchronisation, make it a suitable interaction scheme for dynamic large-scale applications.

Publish/Subscribe systems typically are topic-based or content-based [7]. In the topic-based, publishers publish events on specific topics expressed as keywords (e.g. Sports), and subscribers that have subscribed to these topics get notified whenever there is a match. The content-based improves on the expressiveness of the first one by adding event content filtering on the subscription side. This filtering typically involves comparison operators (=, <, ⩽, >, ⩾) on attribute-value pairs derived from the events. Complex subscription patterns can also be created by logical combinations (and, or etc.) of individual constraints. For example, an event could be (gender = female, age = 20) and a subscription that matches it could be (gender = female, age < 30).

Lately, there has been some attention drawn in graph-based Publish/Subscribe systems [3] that represent publications as graphs. Within these graphs, points of interest are nodes and relations between them are edges. Subscriptions can be SPARQL-like by asking for specific nodes and their relation among them. The notifications are those graphs that match the subscriptions.

4 RELATED WORK

Related work is analysed below, and it is mainly split into two categories; Streaming and Non-Streaming.

4.1 Streaming

4.1.1 Stream Processing Frameworks. There is a plethora of existing stream processing frameworks, like Apache Spark[1], Apache Flink[2] and Apache Kafka[3] that could be extended to support entity summarisation techniques, but some of them do not support Publish/Subscribe. Publish/Subscribe systems, like Apache Kafka, are topic-based; therefore, they are not capable of supporting entities that contain complex semantic data. Furthermore, the constraints of these frameworks in supporting specific data formats or SQL-like APIs could lead to low usability if the user has low expressibility.

4.1.2 Graphs in Publish/Subscribe Systems. As discussed above, topic-based and content-based Publish/Subscribe systems are not capable of supporting entities. Cañas et al. [3] introduce GraPS, a graph-based Publish/Subscribe system that can model publications as graphs, where points of interest are nodes and relations between them form edges. Subscriptions can be either simple ones, like a collection of nodes or complex ones, like specific relations among nodes. Nevertheless, they assume that the subscribers have limited

[1]https://spark.apache.org/
[2]https://flink.apache.org/
[3]https://kafka.apache.org/

knowledge of the graph published to filter it; therefore, they are aware of the semantics and schematics of the graph. This could lead to low usability if the subscribers have low expressibility. Also, they do not support summarisation of the semantic information.

4.1.3 Diversity in Publish/Subscribe Systems. Diversity of events has not been considerably explored in Publish/Subscribe systems. Chen et al. [4] focus on top-k diverse publications in the form of tweets by calculating their cosine similarity, whereas Drosou et al. [6] emphasise on top-k diverse publications in the form of attribute-value pairs by calculating the commonalities among the events. However, both works do not support heterogeneity neither they consider entities as publications, which is a more complex problem.

4.1.4 Summarisation in Publish/Subscribe Systems. Summarisation has been examined in Publish/Subscribe systems by several works. Triantafillou et al. [21] focus on subscription summarisation in the sense of subscription subsumption, that is an attribute-value constraint of a subscription is subsumed by that of another subscription if the values are the same or if they are contained in the values of the latter subscription. Specifically, each subscription is split into its attribute-value pairs, which are then merged into summary structures. Wang et al. [22] emphasise on subscription summaries by partitioning via random, R-tree and K-means clustering techniques and summary-based routing via R-trees among a set of servers to address high system throughput. These works focus on subscription subsumption or covering without considering publication summarisation. They also support simple attribute-value pairs, so they cannot be used for complex semantic data.

4.1.5 Fusion in Publish/Subscribe Systems. Fusion has been used in Publish/Subscribe systems before. Kolozali et al. [13] fuse sensor data from heterogeneous sources and translate attribute-value pairs as time series and then approximate them with dimensionality reduction. Nevertheless, the approximation is done outside the Publish/Subscribe system, and it is not related to entity summarisation. Wun et al. [23] fuse attribute-value pairs that result in semantic interpretations with the use of ontologies. Nevertheless, they tackle a different problem than entity summarisation.

4.1.6 Approximate Semantic Matching in Publish/Subscribe Systems. Approximate semantic matching in Publish/Subscribe systems has been examined by a number of works. These works introduce an additional layer of decoupling, that of semantic decoupling, in Publish/Subscribe systems. Hasan et al. [11] create an approximate semantic single-event processing model for attribute-value pairs coming from heterogeneous sources. Top-1 and top-k matchers are created based on Wikipedia ESA and probabilistic models. Their earlier work [12] focuses on RDF graphs as publications. S-TOPSS [17] uses synonyms, taxonomies and mapping functions specified by domain experts for creating an approximate matcher. Although approximate semantic matching could be related to our work, nevertheless, it is a different problem to entity summarisation.

4.2 Non-Streaming

4.2.1 Diverse Entity Summarisation. Top-k diversity in entities in the form of sophisticated summaries that detect duplication and conceptual similarity has been tackled by a number of works. These works consider high usability as they use keyword-based queries. DIVERSUM [20] focuses on a per-property basis summarisation based on novelty, importance, popularity and diversity by adapting the document-based Information Retrieval to the knowledge graphs. FACES [10] emphasises on summaries based on diversity, uniqueness, and popularity via hierarchical conceptual clustering and the use of WordNet for related terms. FACES-E [9] improves on FACES by considering types in datatype properties instead of only object properties for entity summarisation. Pouriyeh et al. [18] emphasise on summaries based on topic modelling by considering predicates as topics and use of Word2Vec for related terms. All of these works contain static methodologies; therefore, they need to be extended to support a complex dynamic environment.

In conclusion, no existing approach covers the requirements of our problem. Comparison among the works covered in the different subsections is shown in Table 1.

5 APPROACH

The approach is analysed below that defines the event model, the subscription model and the architecture of the summarisation engine.

5.1 Event Model

To support complex semantic data, the event payload contains RDF triples of the form <subject, property, object>. Each event is an instance of an entity (subject) with one predicate (property) and one value for this predicate (object). Below there is an example of a publication payload:

$$\{< Rice\_University >< city >< Houston >\}$$

Therefore, the definition of the event model is as follows: Let EV be the set of events, PID the set of publisher IDs, $PubID$ the set of publication IDs, T the set of timestamps and $Tr(e)$ the triple set of an entity e, respectively, then:

$$\begin{aligned} ev \in EV \Leftrightarrow ev = \{(pID, pubID, t, tr), \dots : \\ pID \in PID, pubID \in PubID, t \in T, tr \in Tr(e)\} \end{aligned} \tag{1}$$

5.2 Subscription Model

To support high usability, we do not assume that subscribers are aware of the semantics and structure of the events or that they are experts in complex query languages, like SPARQL. Therefore, a subscription should ideally be in the form of a keyword query [20].

Subscriptions, therefore, are a set of attribute-value pairs. Only conjunction has been considered in this work. This means that each event needs to fulfil all constraints of a subscription so that it can be considered a match. Each pair consists of an attribute, an equal operator and a value. Below there is an example of a subscription payload:

{entity = "$< Rice\_University >$", k = 5, windowSize = 10, ranking = "Diversity"}

In the example above, the subscriber is interested in an event summary of the entity $< Rice\_University >$ with top-5 diverse information facts derived from the analysis of data taken from count windows of size 10, that is 10 events.

Table 1: Overall comparison of different works

Subsection	High Usability	Diversity	Fusion	Dynamism	Windowing	Entities	Summarisation
4.1.1	No (SQL-like)	No	Yes	Yes	Yes	No	No
4.1.2	No (SQL-like)	No	No	Yes	No	Yes	No
4.1.3	Yes/No (keywords or attribute-value pair constraints)	Yes (duplicates)	No	Yes	Yes	No	Yes (top-k publications)
4.1.4	No (attribute-value pair constraints)	No	No	Yes	No	No	Yes (subscription summaries)
4.1.5	No (attribute-value pair constraints)	No	Yes	Yes	No	No	Yes (approximation)
4.1.6	Yes (approximate semantic matching)	No	No	Yes	No	No	No
4.2.1	Yes (keywords)	Yes (conceptual similarity)	No	No	No	Yes	Yes (top-k triples)

Therefore, the definition of the subscription model is as follows: Let S be the set of subscriptions, SID the set of subscriber IDs, $SubID$ the set of subscription IDs, T the set of timestamps, ATT the set of attributes, OP the set of operators and VAL the set of values, respectively, then:

$$s \in S \Leftrightarrow s = \{(sID, subID, t, (att, op, val)), \ldots : sID \in SID, subID \in SubID, t \in T, (att, op, val) \in ATT \times OP \times VAL\} \quad (2)$$

5.3 Architecture

Our architecture is illustrated in Fig. 3. In the architecture, a *Publisher* creates a number of entity-based publications and a *Subscriber* a number of entity-related subscriptions. All publications and subscriptions enter the *Summarisation Engine*, which is the processing engine of the system.

The engine contains a boolean *Matcher* that extracts the matched entities based on the stored subscriptions and publications. All publications enter the *Window Partitioning* that is responsible for creating tumbling *Count Windows* for each matched entity. The corresponding window is then populated with events from all publishers concerning this entity. All events are fused within the window incrementally, and through the *Summarisation* they are checked for duplicates. Then a score is given in each triple. Triples that are non-duplicates and they are the most recent ones have higher scores. Top-k filtering then involves the diverse top-k most recent triples. Once the corresponding window reaches its capacity that is based on the windowSize defined by the subscriber, the subscriber is notified by the *Notification*, and then the process starts again.

For example, if a subscriber is interested in an event summary of the entity $< Rice\_University >$ with top-5 diverse notifications deriving from the analysis of the last 10 events of Fig. 1 in a window, then, a possible notification payload would be:

$$\{< Rice\_University >< temperature >< 15°C >\}$$
$$\{< Rice\_University >< energyUsage >< 5kWh >\}$$
$$\{< Rice\_University >< city >< Houston >\}$$
$$\{< Rice\_University >< state >< Texas >\}$$
$$\{< Rice\_University >< country >< UnitedStates >\}$$

as the duplicate information of *temperature* and *city* was discarded and the rest of the triples were the most recent ones based on their timestamps.

6 EVALUATION

To the best of our knowledge, no one has tackled entity summarisation in Publish/Subscribe systems. Therefore, we compare our approach with the non-top-k non-fused approach, where all events are sent separately to the subscriber without being fused or checked for duplicates and with the non-top-k fused approach, where the events are fused in the window, but they are not checked for duplicates.

6.1 Dataset

The DBpedia dataset[4] has been selected for our evaluation, as it is highly popular in the field of entity summarisation. Following the entity selection of FACES, 50 entities were chosen that belong to different domains (e.g. politician, actor, country, etc.) and they have per entity an average of 44 distinct direct features. As in FACES, we filtered out any schema information and dataset dependent details, such as *dcterms:subject*, *rdf:type*, *owl:sameAs*, wordnet type and Wikipedia related links. We did not consider literals, only resource-based objects, as they provided richer information.

To simulate the graph evolution, we extracted information from different versions of DBpedia, and we started by adding triples from the oldest version to the newest. All entities and their triples follow a uniform distribution in the selection process by the publishers. 50 publishers are used, and each one is responsible for generating events of one entity. The subscriber is one and generates 50 subscriptions, one for each entity.

All experiments were ran for 5 times, and the average was taken. All runs took place in a laptop with Intel(R) Core(TM) i7-6600U CPU@2.60GHz 2.80GHz and 16GB of RAM.

6.2 Metrics

6.2.1 Redundancy-aware F-score. We are using the metrics of redundancy precision and redundancy recall defined in [24], and

[4] https://wiki.dbpedia.org/

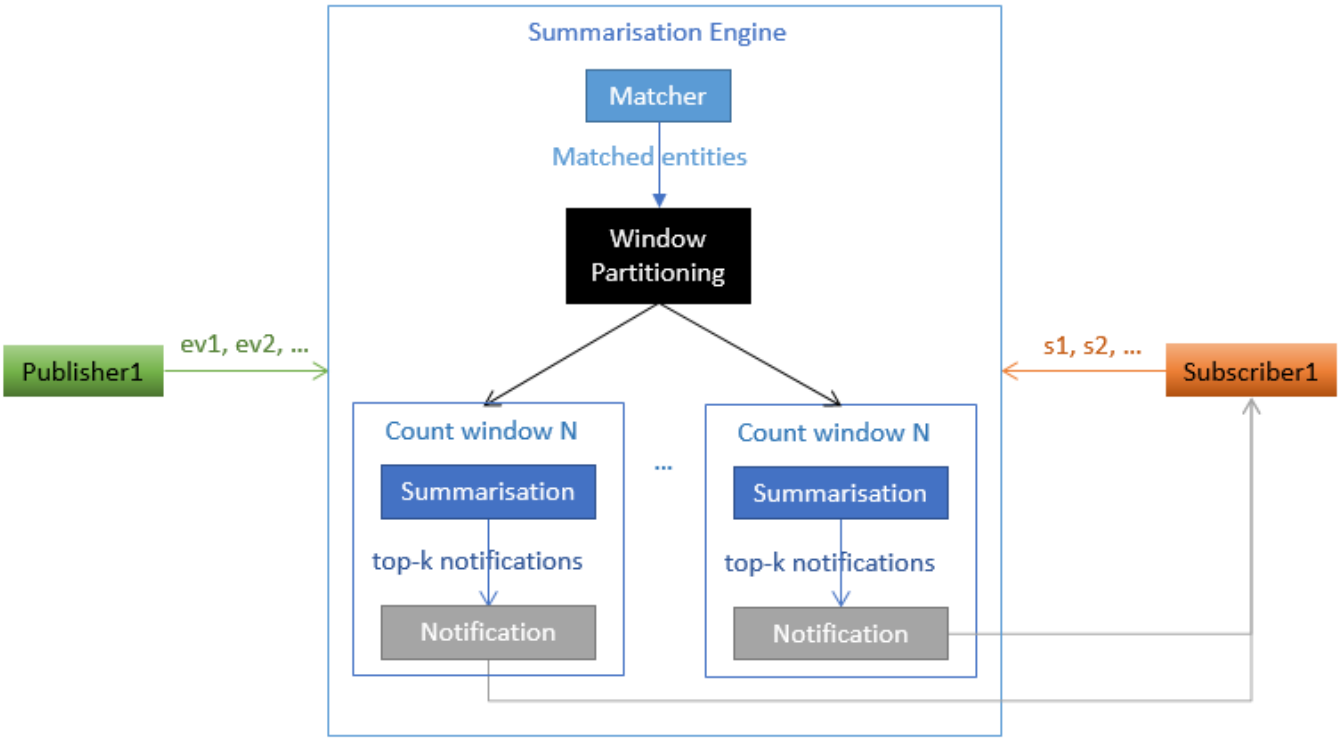

Figure 3: Entity summarisation architecture.

through these, we calculate the redundancy-aware F-score. For our work, we define as "redundant" the duplicate triples. The score is defined as:

$$Red\_precision = \frac{R^-}{R^- + N^-} \tag{3}$$

$$Red\_recall = \frac{R^-}{R^- + R^+} \tag{4}$$

$$Red\_F - score = 2 \times \frac{Red\_precision \times Red\_recall}{Red\_precision + Red\_recall} \tag{5}$$

where R^- is the set of non-delivered redundant triples, N^- is the set of non-delivered non-redundant ones and R^+ is the set of delivered redundant ones.

6.2.2 End-to-End Latency. For the non-fused events, the end-to-end latency is the time it takes between the publication of an event until its delivery. For the fused events, it is the time it takes between the earliest triple in the fusion until the time of the fusion's delivery.

6.2.3 Number of Messages. This metric is split between the number of forwarded messages, that is the number of triples that are sent upstream and the number of redundant messages, that is the number of duplicates within the event set.

6.3 Results

The results are illustrated in Fig. 4. We selected the k value to range from 5 to 30 and the window size to be either of 50 or 100 events.

In Fig. 4(a) we observe that in terms of end-to-end latency, all approaches have higher latencies for larger windows. This is expected as although the fusion and top-k diversity are incremental within the window, the notification is sent after the window is populated; therefore, the population time is also considered. No fusion non-top-k approach behaves slightly better in terms of latency, as once the window is populated all events are sent separately and their latencies are not dependent on the earliest event in the window as in the fused case. The fusion non-top-k and fusion top-k approach have similar behaviour in terms of latency, although the top-k filtering might fluctuate it according to the k, as we observe a slight rise with k.

Fig. 4(b) shows that the number of forwarded messages is reduced within the ranges of 50% to 80% depending on the k for the top-k approach compared to the baselines (both of them had similar results, so only one is shown). For higher values of k, more messages are forwarded upstream. Therefore, the power of top-k filtering is more evident for lower k, assuming not much loss of valuable information occurs. From these messages, the baselines show that 22% and 42% were duplicates for windowSize = 50 and windowSize = 100 respectively (Fig. 4(c)). The top-k approach can discard this duplicate information, therefore, reducing the overall forwarded messages. There is an increase of forwarded messages in the top-k approach for smaller windows. This happens because even though

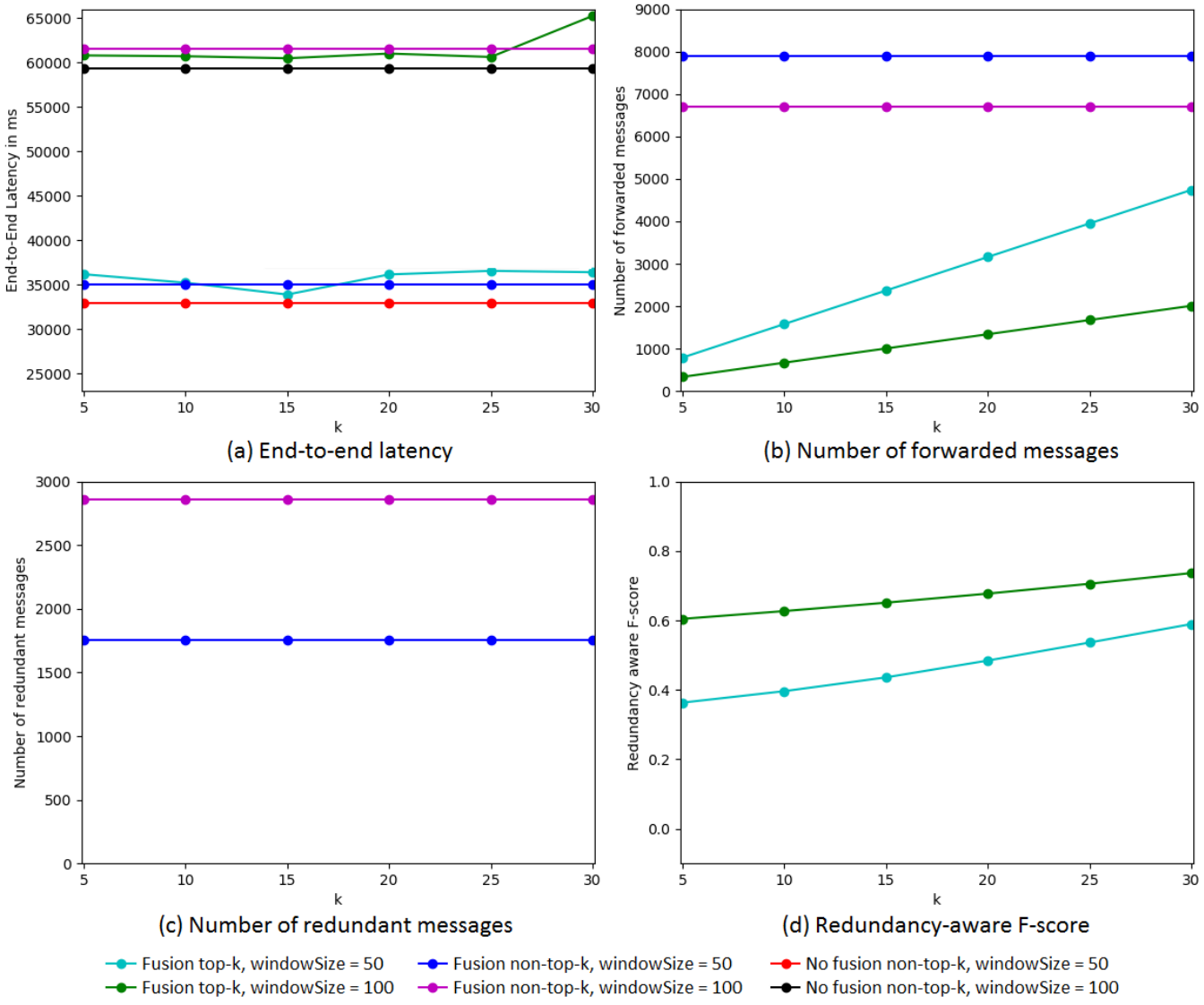

Figure 4: Evaluation results for 50 publishers and 1 subscriber with 50 subscriptions.

the number of forwarded messages is dependent on k for all window sizes, for the same duration there are more notifications produced for smaller windows compared to bigger ones; therefore, more messages are sent in total. This is dependent on the number of events produced by the publishers.

On the other hand, by using top-k filtering results not only in the elimination of duplicate redundant information but in possibly valuable information. This is depicted in Fig. 4(d) by the redundancy-aware F-score that ranges from 0.35 to 0.73. Lower F-score occurs for lower k as stricter content filtering is taking place, whereas higher F-score is observed with the increase in window sizes, as the bigger the window, the more probable redundant information exists.

Therefore, we observe a trade-off between latency, forwarded messages, and expressiveness. Specifically, although top-k filtering reduces the number of duplicates and overall messages sent to the subscriber compared to the baselines with comparable end-to-end latencies, some non-redundant information will be lost.

7 CONCLUSION AND FUTURE WORK

In this paper, we introduce the first window-based diverse entity summarisation in Publish/Subscribe systems that provides high usability and expressive notifications of data deriving from heterogeneous sources in environments like the Internet of Things. We examine the trade-off between latency, number of forwarded messages, and expressiveness. Future work will focus on diversity not only based on duplicates but also on conceptual similarity [16]

and more sophisticated ranking methodologies to boost expressiveness. This work will be further evaluated by adapting static entity summarisation techniques in streaming environments. More personalised subscriptions will also be explored that give opportunities for the subscribers to define which information might be more interesting to them. Finally, more types of windows apart from count ones will be implemented to determine their performance.

ACKNOWLEDGMENTS

This work was supported by the European Union's Horizon 2020 research programme Big Data Value ecosystem (BDVe) grant No 732630 and in part by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289_P2, co-funded by the European Regional Development Fund.

REFERENCES

[1] Charu C Aggarwal. 2007. *Data streams: models and algorithms*. Vol. 31. Springer Science & Business Media.

[2] Ejaz Ahmed and Mubashir Husain Rehmani. 2017. Mobile edge computing: opportunities, solutions, and challenges. (2017).

[3] César Cañas, Eduardo Pacheco, Bettina Kemme, Jörg Kienzle, and Hans-Arno Jacobsen. 2015. Graps: A graph publish/subscribe middleware. In *Proceedings of the 16th Annual Middleware Conference*. ACM, 1–12.

[4] Lisi Chen and Gao Cong. 2015. Diversity-aware top-k publish/subscribe for text stream. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 347–362.

[5] Gong Cheng, Thanh Tran, and Yuzhong Qu. 2011. Relin: relatedness and informativeness-based centrality for entity summarization. In *International Semantic Web Conference*. Springer, 114–129.

[6] Marina Drosou, Kostas Stefanidis, and Evaggelia Pitoura. 2009. Preference-aware publish/subscribe delivery with diversity. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*. ACM, 6.

[7] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. 2003. The many faces of publish/subscribe. *ACM computing surveys (CSUR)* 35, 2 (2003), 114–131.

[8] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. 1987. The vocabulary problem in human-system communication. *Commun. ACM* 30, 11 (1987), 964–971.

[9] Kalpa Gunaratna, Krishnaprasad Thirunarayan, Amit Sheth, and Gong Cheng. 2016. Gleaning types for literals in rdf triples with application to entity summarization. In *European Semantic Web Conference*. Springer, 85–100.

[10] Kalpa Gunaratna, Krishnaprasad Thirunarayan, and Amit P Sheth. 2015. FACES: Diversity-Aware Entity Summarization Using Incremental Hierarchical Conceptual Clustering.. In *AAAI*. 116–122.

[11] Souleiman Hasan and Edward Curry. 2014. Approximate semantic matching of events for the internet of things. *ACM Transactions on Internet Technology (TOIT)* 14, 1 (2014), 2.

[12] Souleiman Hasan, Sean O'Riain, and Edward Curry. 2012. Approximate semantic matching of heterogeneous events. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*. ACM, 252–263.

[13] Sefki Kolozali, Maria Bermudez-Edo, Daniel Puschmann, Frieder Ganz, and Payam Barnaghi. 2014. A knowledge-based approach for real-time iot data stream annotation and processing. In *2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*. IEEE, 215–222.

[14] K Prasanna Lakshmi and CRK Reddy. 2010. A survey on different trends in data streams. In *Networking and Information Technology (ICNIT), 2010 International Conference on*. IEEE, 451–455.

[15] Shobharani Pacha, Suresh Ramalingam Murugan, and R Sethukarasi. 2017. Semantic annotation of summarized sensor data stream for effective query processing. *The Journal of Supercomputing* (2017), 1–23.

[16] Niki Pavlopoulou and Edward Curry. 2019. Using Embeddings for Dynamic Diverse Summarisation in Heterogeneous Graph Streams. In *2019 First International Conference on Graph Computing (GC)*. IEEE.

[17] Milenko Petrovic, Ioana Burcea, and Hans-Arno Jacobsen. 2003. S-topss: Semantic toronto publish/subscribe system. In *Proceedings 2003 VLDB Conference*. Elsevier, 1101–1104.

[18] Seyedamin Pouriyeh, Mehdi Allahyari, Krys Kochut, Gong Cheng, and Hamid Reza Arabnia. 2018. Combining word embedding and knowledge-based topic modeling for entity summarization. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*. IEEE, 252–255.

[19] Yongrui Qin, Quan Z Sheng, Nickolas JG Falkner, Schahram Dustdar, Hua Wang, and Athanasios V Vasilakos. 2016. When things matter: A survey on data-centric internet of things. *Journal of Network and Computer Applications* 64 (2016), 137–153.

[20] Marcin Sydow, Mariusz Pikuła, and Ralf Schenkel. 2010. DIVERSUM: Towards diversified summarisation of entities in knowledge graphs. In *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*. IEEE, 221–226.

[21] Peter Triantafillou and Andreas Economides. 2004. Subscription summarization: A new paradigm for efficient publish/subscribe systems. In *24th International Conference on Distributed Computing Systems, 2004. Proceedings*. IEEE, 562–571.

[22] Yi-min Wang, Lili Qiu, Chad E Verbowski, Demetrios Achlioptas, Gautam Das, and Per-Ake Larson. 2007. Summary-based routing for content-based event distribution networks. (April 3 2007). US Patent 7,200,675.

[23] Alex Wun, Milenko Petrovi, and Hans-Arno Jacobsen. 2007. A system for semantic data fusion in sensor networks. In *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*. ACM, 75–79.

[24] Yi Zhang, Jamie Callan, and Thomas Minka. 2002. Novelty and redundancy detection in adaptive filtering. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 81–88.

[25] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*. ACM, 22–32.