Using Embeddings for Dynamic Diverse Summarisation in Heterogeneous Graph Streams

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

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

Abstract*—A high-volume of data generated nowadays by the rise of Smart Cities and Internet of Things can be represented as graph streams. While many graph processing algorithms could analyse small graphs when challenging real-world graphs occur in distributed settings like sensor-based ones, a more suitable analysis is needed. Specifically, challenges like dynamism, heterogeneity, continuity and high-volume of these graph streams could benefit from real-time analysis. This analysis should happen with reduced network traffic and latency while maintaining high data expressibility and usability. Therefore, our key question is: *Can we define a dynamic graph stream summarisation system that provides expressive graphs while ensuring high usability and limited resource usage?

In this paper, we explore this question and propose a multi-source system with windowing, data fusion, conceptual clustering and top-k scoring that can result in expressive, dynamic graph summaries with limited resources at no expense of usability. Our results show that sending top-k fused diverse summarisation, results in 34% to 90% reduction of forwarded messages and redundancy-awareness with an F-score ranging from 0.57 to 0.88 depending on the k compared to sending all the available information. Also, the summaries' quality follows the agreement of ideal summaries determined by human judges. Nevertheless, these results occur at the expense of higher latency ranging from similar latency to the baseline up to 4 times more depending on the approach; therefore, there is some trade-off between latency, the number of forwarded messages, and expressiveness.

I. Introduction

With the rise of Smart Cities and Internet of Things, we are surrounded by multiple sensors that produce a range of data streams. This data could be more informative if represented as structured graphs. For example, sensor data streams could be represented as conceptual entities with associated relations. These graph streams could then be processed for the needs of users or applications [1].

Since these graph streams are coming from multiple sources, they might present high semantic heterogeneity [2]. For example, different words could describe conceptually similar things (e.g. "energy usage" vs "energy consumption"). Furthermore, due to the possibly high volume and the frequent sampling rate of graph streams, they might contain identical information for a time period resulting in duplication. Duplicates and conceptually similar things result in redundant information. This redundancy may lead to significant propagation, storage overheads in a network of unnecessary data and slower processing time [3].

On the other hand, users may have different levels of ability to express their needs [4]. For example, their queries could range from simple keyword-based to more complex SPARQL-like queries. Where the information in question is coming from multiple sources, the users may need to define complex join queries to gather it from all sources. This complexity may lead to low usability as the user is expected to be aware of complex query languages and the structure of several data sources. At the same time, simple queries may lead to high usability; nevertheless, they may create an abundance of redundant information [5].

The challenges above, when combined with the dynamism, continuity and high volume of sources and users or sinks in smart environments, need an efficient and effective processing system of graph streams [6]. Therefore, our key question is: *Can we define a dynamic graph stream summarisation system that provides expressive (non-redundant) graphs along with high usability while using limited resources?*

As approximate solutions [1] are acceptable as quick answers [7] within a small error range with high probability while using limited resources, we propose a dynamic diverse summarisation system of heterogeneous graph streams with the use of embeddings. In this way, we aspire not only to increase the system performance by reducing the number of data sent upstream but also to create a top-k diverse conceptual data set that will not overwhelm the user with redundant information. Nevertheless, graph summarisation suggests loss of possibly useful information. Therefore, there is a trade-off between latency, the number of forwarded messages, and expressiveness.

Our main contributions are:

- A user-friendly diversity-aware query that allows users to simply express whether they need to receive top-k diverse filtered information of specific window size.
- Introduce a novel dynamic diverse summarisation system for heterogeneous graph stream windows with the use of embeddings that is based on user query relevance, importance and diversity.
- An evaluation methodology for examining the trade-off between latency, the number of messages, and expres-

978-1-7281-4129-9/19/$31.00 ©2019 IEEE
DOI 10.1109/GC46384.2019.00010

siveness within graph stream windows.

The rest of this paper is structured as follows. In Section 2 and Section 3 we present the problem analysis and the preliminaries, respectively. Section 4 contains related work, whereas Section 5 contains the approach. Evaluation and results are in Section 6. Finally, conclusions are drawn in Section 7.

II. Problem Analysis

A. Motivational Scenario

Imagine Houston is a smart city (shown in Fig. 1), and a user is interested in information about *Rice University*. Several sensor readings contain information about the university, ranging from temperature to location. The user has no other information apart from the university's name, and one needs to quickly gain knowledge about the university.

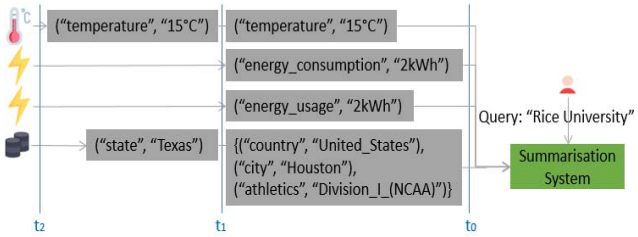

Fig. 1. A user is interested in information about *Rice University* and sources generate timestamped information records. Here, the *temperature* information is duplicate and the *energy_usage* and *energy_consumption* are conceptually similar.

B. Problem Challenges

The aforementioned scenario faces many challenges:

- **Heterogeneity**: Multiple sources create heterogeneous data about the university. Some of this data, like *temperature* is duplicate, whereas other like *energy_usage* and *energy_consumption* is conceptually similar. Duplicate and conceptually similar information lead to redundancy that may overwhelm the user.
- **Low user expressibility**: The user has limited information about the university. One might also need to create complex join queries to gather all the information from the necessary sources. Nevertheless, the user is unable to create a complex filtering query and is not an expert in query languages. On the other hand, if the user creates an abstract or general query, it may lead to redundant or undesired information.
- **Dynamism**: The sources that generate information about the university may be created or deleted at any time.
- **Continuity**: The sources constantly update the university's information, and the user needs the most recent data [8].
- **High data volume**: The high volume of information that is created by the sources needs to be properly filtered to not overwhelm the user.

III. Preliminaries

Knowledge graphs contain information regarding entities, which are real-world or abstract things [9]. Within knowledge graphs the nodes represent the entities, and the directed labelled arcs constitute relations among them. In Fig. 1 the sensor readings could be represented as structured graphs. For example, *Rice University*, *15°C*, *2kWh*, *Texas*, *United States*, *Houston*, and *Division I (NCAA)* could be entities or literals, whereas *temperature*, *energy consumption*, *energy usage*, *state*, *country*, *city*, and *athletics* could be relations among the connected entities or literals by the directed arc. 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. 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 [9].

By adopting and adapting definitions that were introduced in Cheng et al. [10], we provide some definitions for completeness.

Let E be the set of all entities, L the set of all literals, P the set of all properties, Tr the set of all triples and T the set of all timestamps.

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 labelling 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 (Graph Stream). A graph stream $Gs = \langle (tr_i, t_i) | i \in \mathbb{N} \rangle$ is a sequence of pairs where each pair consists of a triple $tr \in Tr$ and its timestamp $t \in T$.

Definition 4 (Dynamic Diverse Entity Summarisation). Given a snapshot of Gs and a positive integer $k < |Gs|$, the problem of dynamic diverse entity summarisation is to select $Summ = \langle Trsum, t \rangle$ where $Trsum \subset Tr$ such that $|Summ| = k$. In other words, $Summ$ is called a dynamic diverse summary of an entity e and it contains a set of unique and conceptually diverse triples that belong to the snapshot of Gs, as well as its timestamp $t \in T$.

To support high usability, we do not assume that users have high expressibility, that is they are experts in complex query languages, like SPARQL. Therefore, a query should ideally be a keyword-based one [9]. Nevertheless, keyword-based queries are too abstract, which may lead to receiving undesired information. Therefore, a high-level ranking policy should be defined by the user that could filter some of the undesired information. The user can select from complex ranking (e.g. diverse) to no ranking at all (e.g. none).

Definition 5 (Diversity-aware Query). A diversity-aware query DAQ is a sequence of ⟨entity, k, window size, ranking

policy$\rangle$ defined as $DAQ = \langle e, k, ws, r \rangle$, where $e \in E$, $k \in \mathbb{N}$, $ws \in \mathbb{N}$ and $r \in \{Diversity, None\}$.

IV. Related Work

A. Non-Streaming

1) Graph Summarisation: Many graph summarisation techniques are covered by Liu et al. [11]. These are split into static and dynamic graph summarisation techniques. In the static case, plain graph summarisation examines only the graph's structure, whereas the labelled graph summarisation examines the graph's labels too. In the dynamic case, plain graph summarisation examines the temporal structure. Currently, there is no dynamic labelled graph summarisation. In our understanding, plain entity summarisation is related to static labelled graph summarisation, as both the structure and the labels (subject, property, object values) are analysed. Our work is aspiring to introduce dynamic entity summarisation that could be related to dynamic labelled graph summarisation, where both temporal structure and labels are considered.

2) Plain Diverse Entity Summarisation and Approximation: Top-k diversity in entities by summaries that detect duplication and conceptual similarity are tackled by several works that also consider high usability via keyword-based queries. DIVERSUM [9] focuses on a per-property summarisation based on novelty, importance, popularity and diversity by adapting the document-based Information Retrieval to the knowledge graphs. FACES [12] emphasises on summaries based on diversity, uniqueness, and popularity via hierarchical conceptual clustering and the use of WordNet for related terms. FACES-E [13] improves on FACES by also considering types in datatype properties. Pouriyeh et al. [14] emphasise on summaries based on topic modelling by considering properties as topics and use of Word2Vec for related terms. Other works by Harth et al. [15] and Pan et al. [16] emphasise on RDF approximation (e.g. RDF sampling, histograms, compression). All of these works contain static methodologies; therefore, they need to be extended to support a complex dynamic environment.

B. Streaming

1) Stream Processing Frameworks: Existing stream processing frameworks, like Apache Spark[1], Flink[2] and Kafka[3] do not support entity summarisation techniques; therefore, they need to be extended. Also, their constraints in supporting specific non-graph-based data formats or SQL-like queries could lead to low usability if the user has low expressibility.

2) Stream Approximation: Work has been done in dynamic stream approximation, but mostly for numerical or string data. These include synopsis methods, frequent patterns, clustering or dimensionality reduction mainly in Aggarwal et al. [17] and Gupta et al. [18]. Tang et al. [19] focuses on synopsis in graph streams, but without emphasising on labelled graphs. Le-Phuoc et al. [20] focuses on RDF stream processing, but without emphasising on summaries. Dia et al. [21] extend SPARQL for supporting RDF stream sampling, which is different from diverse entity summarisation.

In conclusion, no existing approach covers the requirements of our problem.

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

V. Approach

A. Architecture

Our architecture is illustrated in Fig. 2. Sources create graph streams concerning entities and users create diversity-aware queries concerning these entities. All graph streams and queries enter the *Summarisation System*, which analyses the graph streams and notifies the users.

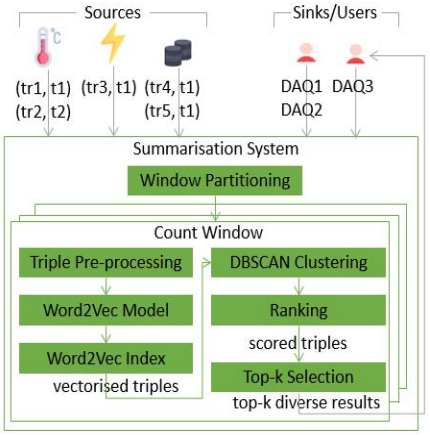

Fig. 2. Architecture of the Dynamic Diverse Summarisation System.

All graph streams enter the *Window Partitioning* that creates tumbling *Count Windows* for each user-required entity. Each window, then, is populated with triples from all sources concerning a specific entity and through *Triple Pre-processor* all of their content is extracted. Duplicate triples are discarded at this stage. A *Word2Vec Model* [22] is then used to create a *Word2Vec Index* that creates vectors related to each triple. Once the window reaches its full capacity based on the user-defined window size ws, all vectors are fused and undergo *DBSCAN Clustering* [23]. Conceptual clusters are then created that are ranked through *Ranking* based on the importance of the triples in each cluster. The top-k triples are then selected by the resulting scored triples via *Top-k Selection* and this diverse set is sent as a notification to the users. Then the process starts again.

B. Dynamic Diverse Summarisation Algorithm

Our algorithm is divided into two stages: 1) conceptual clustering of triples and 2) ranking of triples. The first stage is done by the combination of embeddings and density-based clustering. The second stage is a combination of similarity metrics and some pre-defined rules. A simplified illustration of the algorithm is provided in Fig. 3.

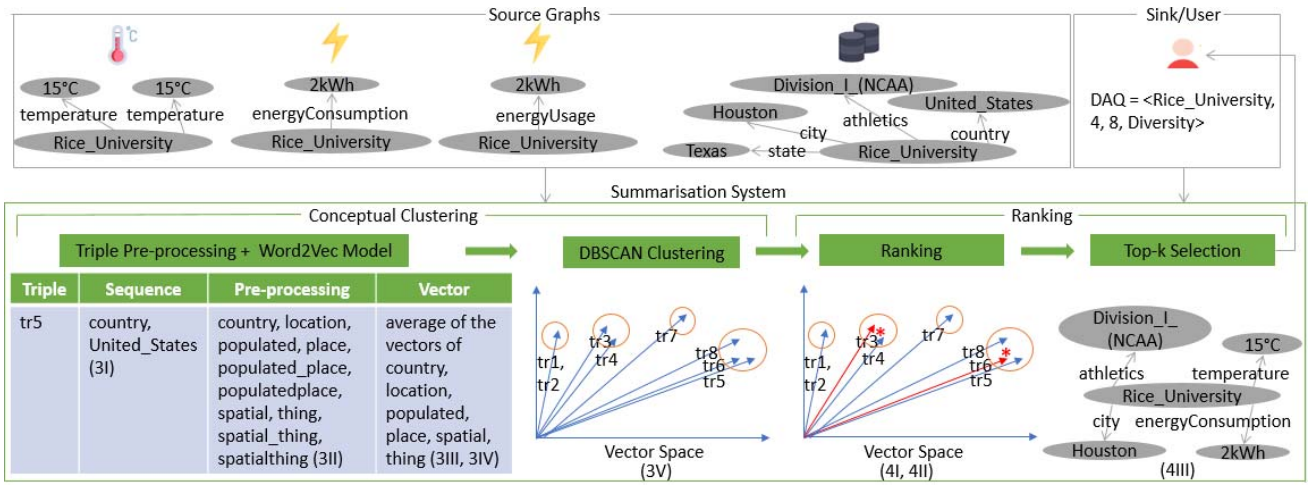

Fig. 3. The sources of Fig. 1 generate information that is represented as structured graphs. A user asks for the top-4 diverse information items for *Rice University* from a window that contains the last 8 facts of the entity. The *Summarisation System* pre-processes all triples (we see an example for triple 5), and all vectorised triples are depicted in the vector space. There, conceptual clusters are found, which are shown in orange circles, and the triples are ranked for each cluster based on their distance with the cluster centroid (red asterisk). The red vectors are the triples closest to the centroids. The top-4 triples are selected and sent as notification to the user. In the example, t1:{Rice_University, temperature, 15°C}, t2:{Rice_University, temperature, 15°C}, t3:{Rice_University, energyConsumption, 2kWh}, t4:{Rice_University, energyUsage, 2kWh}, t5:{Rice_University, country, United_States}, t6:{Rice_University, city, Houston}, t7:{Rice_University, athletics, Division_I_(NCAA)}, t8:{Rice_University, state, Texas}.

1) Word2Vec: Word embeddings are often used in Natural Language Processing (NLP) to represent words into vectors so that a word can be visualised in a vector space. Afterwards, several analysis methodologies can take place to detect, for example, semantically similar or related words. Word2Vec [22] is one of the most successful models in this category. In this model, a neural network is trained by raw text either by the use of Continuous Bag of Words (CBOW) or by Skip-Gram. CBOW predicts a word given surrounding words, whereas Skip-Gram estimates the surrounding words for a single word.

Different studies have used different parameters and datasets for the model, but we are using the pre-trained GoogleNews, Vectors and VectorsPhrase models[4]. The first model has been trained on Google News, and it has proved rather successful, whereas the other two models have been trained in a much smaller dataset. The reasons for choosing Word2Vec are: a) no need to extend existing words of triples with synonyms or hypernyms from thesauri to catch their conceptual meaning, b) words are represented based on context, therefore, semantically opposite words (antonyms), but conceptually similar (e.g. death place - birthplace) are represented closely in the vector space and c) phrases can also be trained.

2) DBSCAN: DBSCAN [23] is a density-based clustering algorithm that has been widely used in research. The algorithm starts by discovering the core points based on their neighbourhood. Then the density-reachable points from these core points are discovered iteratively to define the clusters. The algorithm terminates when no new point can be added to a cluster. The points that belong to no cluster are considered noise. There are two parameters for the algorithm, the minimum points $minPts$ in a neighbourhood for a point to be considered as a core one, including the point itself and the maximum distance ε between two points for them to be considered as in the same neighbourhood.

The reasons of choosing DBSCAN are: a) a strict density-based clustering is efficient for identifying regions of similar or related words in a vector space, b) there is no restriction on the size and the shape of the clusters and c) there is no need of specifying the number of clusters (number of conceptual clusters should not be known a priori).

3) Conceptual Clustering: The stages of conceptual clustering per count window are described below:

I) **Triple to Sequence**: For each graph stream that corresponds to an entity, we need to extract all its triples and define their position in a vector space for future clustering. A Word2Vec model can be used in this case, but not directly in graphs, as it would in a document. RDF2Vec [24] proposed to convert an RDF graph into a sequence of subjects, properties and objects that could be equivalent to a document sentence and then to use Word2Vec in the sequence. Therefore, in our case, each triple is converted into a sequence of property, object since the subject is the same for all present or future triples in a window. If a triple has already been processed (duplicate), then it is deleted and not further analysed. In this way, the user is not presented with duplicate information.

II) **Pre-processing**: Each property and object is pre-processed. Pre-processing involves lower-casing (original word), tokenising (tokens), removing stop-words and concatenating each token with an underscore (concatenated word). For the property, the pre-processing happens on the actual property, but for the object, it happens on its type.

III) **Word to Vector**: The original word, the tokens and the

[4]https://code.google.com/archive/p/word2vec/

concatenated word are sent to the Word2Vec model to extract their vector. If the vector does not exist (some words might not exist, or they might not be present in the training set of the model), then the word is not considered for further action. If the vector of the original word exists, then its vector along with the vectors of the tokens are considered for further action. If it does not exist, then the vector of the concatenated word is examined. If it exists then that vector together with the vectors of tokens are considered; otherwise, the vectors of the tokens are averaged, so that the original word is now represented by the averaged vector and along with the token vectors they are further used. The representation of a phrase (original word or concatenated word) with this average vector might be risky sometimes, as the meaning might be different, but only models that have been trained on these phrases can be accurate. If the word has been visited, then an index is created that contains the words and their corresponding vectors, so that they will not be visited again. If there are duplicate words among the processed words of the property and the object, then they are considered once.

IV) **Triple to Vector**: The vectors of all the remaining pre-processed words are averaged, and now this constitutes the vector of the triple.

V) **Triple Clustering**: All vectors are fed in DBSCAN, and conceptual clusters are created, that is a cluster might contain one or more triples.

In the implementation, for the RDF models and some processing, we use Apache Jena[5], for the Word2Vec model we use deeplearning4j[6] and for DBSCAN we use Smile[7].

4) Ranking: The stages of ranking per count window are described below:

I) **Cluster Centroid**: DBSCAN does not create centroids, therefore, for each conceptual cluster created (with size greater than 1), the average vector of all vectors of the triples that belong to this cluster is calculated. This constitutes the cluster's centroid.

II) **Triple Distance to Centroid**: The cluster's centroid is considered to be the representative vector for each cluster. Therefore, the Euclidean distance or the Cosine similarity between each member of the cluster and its centroid is calculated. Triples that are closer to the centroid are more representative of the conceptual cluster than others.

III) **Triple Selection from Cluster**: Triple selection starts from the biggest cluster to the smallest one. All clusters are visited once in the first round. The triple that has the lowest (for Euclidean distance) or the highest (for Cosine similarity) distance/similarity is picked first for each cluster. If the cluster is of size 1, then the only member is selected. If there are ties among the distances/similarities, then a random selection is made. If a triple has been selected in general, it is not selected again. If a property has been selected once in a cluster, then it is not selected again from this cluster in the next rounds, giving a chance to other properties to be selected (more diversity in properties even among members of the same conceptual cluster). If all properties for each cluster have been selected, then the selection process starts again with the remaining triples. The process continues until either all triples have been visited or the k parameter has been reached. During this selection, a score is given to each triple with descending order, as they are selected.

[5]https://jena.apache.org/

[6]https://deeplearning4j.org/

[7]https://haifengl.github.io/smile/nlp.html

VI. Evaluation

To the best of our knowledge, no one has tackled dynamic diverse entity summarisation in heterogeneous multi-source systems. Therefore, we compare our approach with the non-top-k fused approach, where the triples are fused in the window, but they are not checked for redundancy.

All experiments were run 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.

A. Dataset

The FACES dataset[8] has been selected for our evaluation, which is based on DBpedia[9] 3.9. The dataset has 50 entities of different domains (e.g. politician, actor, etc.) with 44 distinct direct features on average per entity. We only focused on resource-based objects and not literals as they provide richer information, and we pre-processed the data by keeping only the last part after a "/" or "#" in URIs so that the data makes more sense from the user perspective.

All entities and their triples follow a uniform distribution in the selection process by the sources. 50 sources are used, and each one is responsible for generating a stream related to one entity. There is only one sink that generates 50 queries, one for each entity.

B. Metrics

Several metrics have been used to evaluate the efficiency and effectiveness of our approach. These include correctness that consists of the agreement, quality and redundancy-aware F-score, and the end-to-end latency, as well as the number of messages.

1) Correctness: Correctness consists of the agreement, quality and redundancy-aware F-score metrics. We used the ideal summaries of FACES, as we wanted to identify if our approach produces correct and trustworthy summaries that are appealing to the human judgement. These ideal summaries have been created by asking 15 human judges with a background in Semantic Web to select ideal triples for specific entities for k = 5 and k = 10 triples. Each entity has at least 7 ideal summaries from 7 different judges, which constitutes the gold standard.

[8]http://wiki.knoesis.org/index.php/FACES

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

a) Agreement and Quality: The agreement Agr defines how consistent the ideal summaries are between one another, and the quality Q_t defines the commonalities between the human-defined ideal summaries and the approach's summaries for each entity. We used the agreement metric of FACES and RELIN [10] and we adapted a time-dependent version of their quality metric defined below.

$$Agr = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} |Summ_i^I(e) \cap Summ_j^I(e)| \quad (1)$$

$$Q_t = \frac{1}{n} \sum_{i=1}^{n} |\frac{Summ_t(e) \cap (Summ_i^I(e) \cap WTr_t(e))}{Summ_i^I(e) \cap WTr_t(e)}| \quad (2)$$

where n is the number of summaries, $Summ_i^I(e)$ is the i-th ideal summary for an entity e, $Summ_t(e)$ is the approach's summary in time t, and $WTr_t(e)$ are the triples of entity e existing in the window in time t.

Our quality metric is dependent on time since this is a dynamic entity summarisation and static ideal summaries might contain information that is not yet known to the system, that is it has not been published yet. Therefore, each $Summ_i t^I(e)$ contains only the common triples between the already known triples in the system $WTr_t(e)$ and the ones selected from each judge. Then each of these time-dependent ideal summaries is checked for commonalities with the approach's summary $Summ_t(e)$ that has been extracted at that specific time. In the case of duplicate triples, these commonalities are only counted once. In the quality metric in FACES and RELIN, there is no use of a denominator, because, for example, k = 10 applies for all ideal summaries (e.g. 2/10 or 8/10 common triples), but in our case the k is dependent on the commonalities between the $WTr_t(e)$ and the $Summ_i^I(e)$. Therefore, we might have diverse results (e.g. 2/8 or 5/6 common triples), so the denominator is used for normalisation.

b) Redundancy-aware F-score: We are using the metrics of redundancy-precision and redundancy-recall defined in [25], and through these, we calculate the redundancy-aware F-score. For our work, we define as "redundant" the duplicate triples. The score is defined as:

$$Red\_pr = \frac{R^-}{R^- + N^-} \quad \text{and} \quad Red\_rec = \frac{R^-}{R^- + R^+} \quad (3)$$

$$Red\_F - score = 2 \times \frac{Red\_pr \times Red\_rec}{Red\_pr + Red\_rec} \quad (4)$$

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.

2) End-to-End Latency: The end-to-end latency is the time it takes between the generation of a triple by a source until its delivery to the sink. Since our summaries involve multiple streams, our end-to-end latency is the time it takes between the earliest triple in the fusion until the time of the fusion's delivery.

3) Number of Messages: This metric is split between the number of forwarded messages, that is the number of triples within the graph that is sent to the sink and the number of redundant messages, that is the number of duplicates of the graph.

C. Results

The results are shown in Fig. 4 for 50 sources that generate 500 triples each and 1 sink with 50 queries with window sizes of 30 and 50.

1) Agreement: The agreement results are $Agr = 1.96$ and $Agr = 4.7$ for $k = 5$ and $k = 10$, respectively. We observe that there is a good agreement among judges with almost 2 out of 5 and 5 out of 10 triples being common. This proves that there are different levels of expressiveness that are ideal from one user to another or that their perception of what is important or not is different. There is though some common ground, which is shown in the agreement values.

2) Quality: In terms of DBSCAN parameters, we did not observe much difference among $\varepsilon = \{0.1, ..., 3\}$ in the quality metric. However, with the increase in $minPts$, the clustering becomes less efficient. This makes sense, as higher $minPts$ and lower ε result in highly dense clusters, which is a strict criterion. Therefore, in our case, we emphasised on $\varepsilon = 1$ and $minPts = 1$, as we wanted to relax the latter parameter so that triples can form large, as well as very small conceptual clusters. Also, the Euclidean distance against the Cosine similarity, in the ranking, did not make much difference; therefore, we used the Euclidean distance for our results.

Fig. 4(a) shows that the quality gets better with higher k, and it is analogous to the agreement for k = 5 and k = 10. This means that the overlap among the ideal summaries and the approach based ones followed the consensus among the ideal summaries that the judges gave. We also observe that all three Word2Vec models behave similarly, but the GoogleNews one is slightly better. Also, the quality gets better for smaller windows. This happens because the windows contain fewer triples compared to bigger windows, so the commonality between published and ideal triples is less probable.

3) Redundancy-aware F-score: Redundancy-aware F-score in Fig. 4(b) depicts that by using top-k filtering, we result not only in the elimination of duplicate redundant information but in possibly valuable information. Nevertheless, we observe that the F-score ranges from 0.57 to 0.88. 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.

4) End-to-End Latency: In Fig. 4(c), we observe that the slowest model is GoogleNews, followed by VectorsPhrase, Vectors and non-top-k. This happens because the GoogleNews model is a 3.5GB model that needs 150269ms to be loaded once in our system. The other Word2Vec models are much smaller (54.39 MB for Vectors and 519.81 MB for VectorsPhrase) and need far less time. Non-top-k is the best in terms

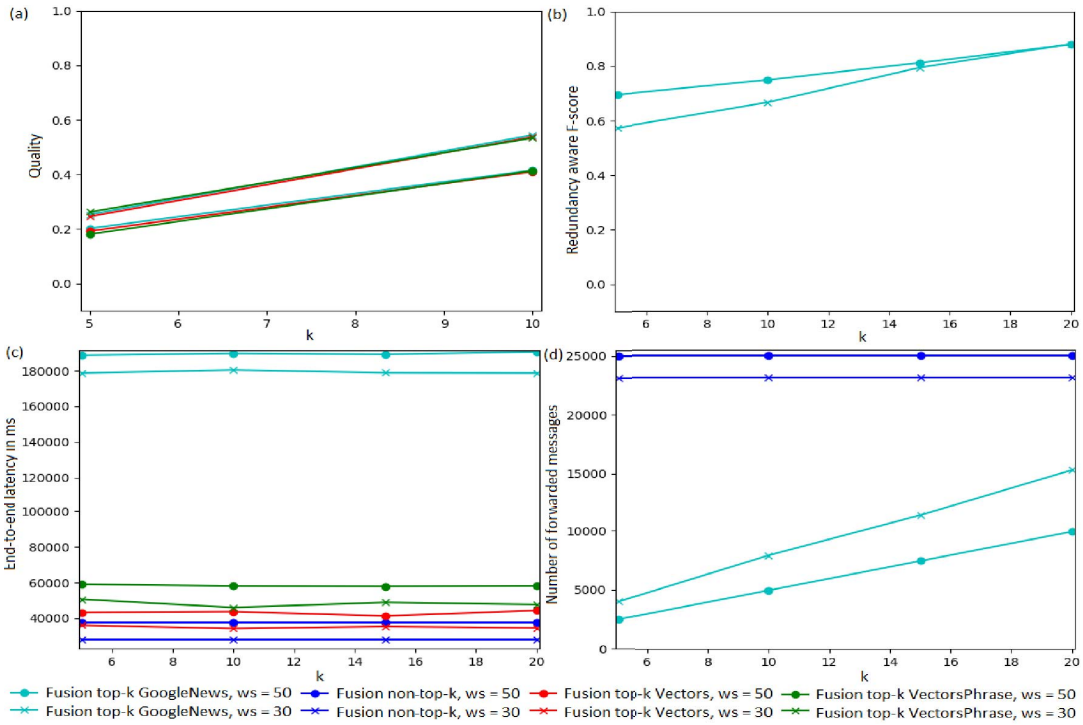

Fig. 4. Results for 50 sources that generate 500 triples each and 1 sink with 50 queries.

of end-to-end latency since no processing is involved when sending notifications, but it is not that much quicker compared to the smaller Word2Vec models with Vectors being very close to it.

There are no significant differences in the latencies for the different k as the processing is done for all data independent of k, but the latency increases with the window size. This is expected as although the fusion and top-k diversity are incremental within the window, the summary is sent after the window is populated; therefore, the population time is also considered.

The reason the latencies seem large is because they are end-to-end, that is the summary that is created each time for an entity is the accumulation of the top entity information in the window that contains the times these facts were created. So the timestamp of whichever fact contributed to the summary affects the summary's end-to-end latency.

5) Number of Messages: In Fig. 4(d), we see that the number of forwarded messages is reduced within the ranges of 90% to 60% depending on the k for the top-k approach compared to the non-top-k for ws = 50 and 82.7% to 34.1% for ws = 30 respectively. The decrease in percentage reduction with the increase of k is expected, as with higher values of k more information is sent. From these messages, non-top-k showed that 34.3% and 48.6% were duplicates for ws = 30 and ws = 50 respectively. The top-k approach can discard this duplicate information, therefore, reducing the overall forwarded messages. We observe an increase of forwarded messages in the top-k approach for smaller windows. This happens because even though 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.

6) Discussion: According to our results, we conclude that non-top-k sends all the information to the user, that is an abundance of facts that might contain all the information, but will also contain duplicate or conceptually redundant one. This scenario becomes worse with the increase of sources, as more and more data related to the entity in question is generated with different variations of conceptual similarity. The top-k approach, on the other hand, decreases significantly the amount of data that is sent as a summary to the user. Even though this information might not be all possibly non-redundant or ideal to the user, nevertheless, it manages to follow the agreement among judges.

In terms of latency, non-top-k is more efficient compared to top-k as no pre-processing is involved. The agreement, though, among the Word2Vec models in terms of quality, redundancy-aware F-score and number of forwarded messages shows that even with smaller models that do not take much more processing time than the non-top-k one, we can achieve similar results.

Therefore, there is a trade-off between latency, the number of forwarded as well as redundant messages, and expressiveness (represented by the quality and redundancy-aware F-score) between a non-top-k scenario and a diverse top-k one. This is because the latency is better in non-top-k, but the number of forwarded as well as redundant messages

and expressiveness are worse and vice verse for the top-k. We conclude, then, that slightly more processing time for finding diverse data, can lead to less data being sent upstream for further processing, data that is seen as expressive as the agreement among judges without containing redundant information (duplicate or conceptual one).

VII. Conclusion and Future Work

In this paper, we emphasised that with the rise of sensors, there are data challenges involved in high-volume, heterogeneity, dynamism, continuity and usability. Therefore, we need a system that can tackle these issues. We presented, to the best of our knowledge, a novel dynamic diverse entity summarisation in heterogeneous multi-source systems. The approach uses conceptual clustering with a combination of Word2Vec models, DBSCAN clustering, and ranking to score entity information to provide users with top-k diverse entity summaries. According to our findings, these summaries are expressive enough in terms of overlap with ideal summaries from human judges, they are redundancy-aware, that is they do not contain any duplicates and, therefore, they do not overwhelm the user and the system, as only a small percentage is sent to the sinks compared to sending all of the available information. In terms of processing time, this can vary depending on the size of the models used (smaller models demand less time), but small models are not using much more resources compared to non-top-k ones.

Future work could be related to a range of directions. One direction could be connected to improving the quality metric results by boosting expressiveness. This could involve a more sophisticated ranking approach based on the graph properties or bigger datasets that could provide more information on ideal summaries or adapting existing static diverse entity summary approaches to smart environments. Another direction could be related to the actual graphs, that is the approach could be extended for not only resource-based objects but literals or not only star-like graphs but multi-depth graphs for discovering relationships among entities. Furthermore, the extension of simple user queries to preferential or complex ones, like domain-specific or feature-specific or multi-entity could also be explored. Finally, more windowing policies will be analysed to check their suitability.

Acknowledgment

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] K. P. Lakshmi and C. Reddy, "A survey on different trends in data streams," in *Networking and Information Technology (ICNIT), 2010 International Conference on*. IEEE, 2010, pp. 451–455.

[2] Y. Qin, Q. Z. Sheng, N. J. Falkner, S. Dustdar, H. Wang, and A. V. Vasilakos, "When things matter: A survey on data-centric internet of things," *Journal of Network and Computer Applications*, vol. 64, pp. 137–153, 2016.

[3] E. Ahmed and M. H. Rehmani, "Mobile edge computing: opportunities, solutions, and challenges," 2017.

[4] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais, "The vocabulary problem in human-system communication," *Communications of the ACM*, vol. 30, no. 11, pp. 964–971, 1987.

[5] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification," in *Proceedings of the 14th international conference on World Wide Web*. ACM, 2005, pp. 22–32.

[6] N. Pavlopoulou and E. Curry, "Towards a window-based diverse entity summarisation engine in publish/subscribe systems," in *Proceedings of EYRE 19: 2nd International Workshop on Entity Retrieval*. ACM, 2019.

[7] C. C. Aggarwal, *Data streams: models and algorithms*. Springer Science & Business Media, 2007, vol. 31.

[8] H. Liu, Y. Lin, and J. Han, "Methods for mining frequent items in data streams: an overview," *Knowledge and information systems*, vol. 26, no. 1, pp. 1–30, 2011.

[9] M. Sydow, M. Pikuła, and R. Schenkel, "Diversum: Towards diversified summarisation of entities in knowledge graphs," in *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*. IEEE, 2010, pp. 221–226.

[10] G. Cheng, T. Tran, and Y. Qu, "Relin: relatedness and informativeness-based centrality for entity summarization," in *International Semantic Web Conference*. Springer, 2011, pp. 114–129.

[11] Y. Liu, T. Safavi, A. Dighe, and D. Koutra, "Graph summarization methods and applications: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, p. 62, 2018.

[12] K. Gunaratna, K. Thirunarayan, and A. P. Sheth, "Faces: Diversity-aware entity summarization using incremental hierarchical conceptual clustering." in *AAAI*, 2015, pp. 116–122.

[13] K. Gunaratna, K. Thirunarayan, A. Sheth, and G. Cheng, "Gleaning types for literals in rdf triples with application to entity summarization," in *International Semantic Web Conference*. Springer, 2016, pp. 85–100.

[14] S. Pouriyeh, M. Allahyari, K. Kochut, G. Cheng, and H. R. Arabnia, "Combining word embedding and knowledge-based topic modeling for entity summarization," in *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*. IEEE, 2018, pp. 252–255.

[15] A. Harth, K. Hose, M. Karnstedt, A. Polleres, K.-U. Sattler, and J. Umbrich, "Data summaries for on-demand queries over linked data," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 411–420.

[16] J. Z. Pan, J. M. Gómez-Pérez, Y. Ren, H. Wu, and M. Zhu, "Ssp: compressing rdf data by summarisation, serialisation and predictive encoding," 2014.

[17] C. C. Aggarwal and S. Y. Philip, "A survey of synopsis construction in data streams," in *Data Streams*. Springer, 2007, pp. 169–207.

[18] N. Gupta and I. Rajput, "Stream data mining: a survey," *Indrjeet Rajput Int. J. Eng. Res. Appl. IJERA ISSN*, pp. 2248–9622, 2013.

[19] N. Tang, Q. Chen, and P. Mitra, "Graph stream summarization: From big bang to big crunch," in *Proceedings of the 2016 International Conference on Management of Data*. ACM, 2016, pp. 1481–1496.

[20] D. Le-Phuoc, J. X. Parreira, and M. Hauswirth, "Linked stream data processing," in *Reasoning Web International Summer School*. Springer, 2012, pp. 245–289.

[21] A. F. Dia, Z. Kazi-Aoul, A. Boly, and Y. Chabchoub, "C-sparql extension for sampling rdf graphs streams," in *Advances in Knowledge Discovery and Management*. Springer, 2018, pp. 23–40.

[22] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[23] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[24] P. Ristoski and H. Paulheim, "Rdf2vec: Rdf graph embeddings for data mining," in *International Semantic Web Conference*. Springer, 2016, pp. 498–514.

[25] Y. Zhang, J. Callan, and T. Minka, "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, 2002, pp. 81–88.