International Journal of Graph Computing
Vol. 1, No. 1 (2020) 70-94
© KS Press, Institute for Semantic Computing Foundation
DOI: 10.35708/GC1868-126724

Dynamic Diverse Summarisation in Heterogeneous Graph Streams: a Comparison between Thesaurus/Ontology-based and Embeddings-based Approaches

Niki Pavlopoulou and Edward Curry

Insight Centre for Data Analytics
Data Science Institute
National University of Ireland, Galway
Galway, Ireland
{niki.pavlopoulou,edward.curry}@insight-centre.org

Received (10/22/2019)
Revised (01/10/2020)
Accepted (02/28/2020)

Abstract. Nowadays, there is a lot of attention drawn in smart environments, like Smart Cities and Internet of Things. These environments generate data streams that could be represented as graphs, which can be analysed in real-time to satisfy user or application needs. The challenges involved in these environments, ranging from the dynamism, heterogeneity, continuity, and high-volume of these real-world graph streams create new requirements for graph processing algorithms. We propose a dynamic graph stream summarisation system with the use of embeddings that provides expressive graphs while ensuring high usability and limited resource usage. In this paper, we examine the performance comparison between our embeddings-based approach and an existing thesaurus/ontology-based approach (FACES) that we adapted in a dynamic environment with the use of windows and data fusion. Both approaches use conceptual clustering and top-k scoring that can result in expressive, dynamic graph summaries with limited resources. Evaluations show that sending top-k fused diverse summaries, results in 34% to 92% reduction of forwarded messages and redundancy-awareness with an F-score ranging from 0.80 to 0.95 depending on the k compared to sending all the available information without top-k scoring. Also, the summaries' quality follows the agreement of ideal summaries determined by human judges. The summarisation approaches come with the expense of reduced system performance. The thesaurus/ontology-based approach proved 6 times more latency-heavy and 3 times more memory-heavy compared to the most expensive embeddings-based approach while having lower throughput but provided slightly better quality summaries.

Keywords: Multi-source Systems; Entity Summarisation; Diversity; Graph Streams; RDF Graphs; Word Embeddings; Thesaurus

1 Introduction

The emergence of Smart Homes, Smart Cities and Internet of Things has driven the deployment of multiple sensors (sources) that create a wide range of data streams that could be represented as structured graphs, as well as multiple users or applications (sinks) that are interested in the analysis of these graph streams for various needs. These needs could vary from transport (e.g. analysis of highway traffic to discover times of congestion [19]) to safety (e.g. seismic measurements to monitor safety risks in specific regions from possible earthquakes [31]).

Although these multi-source environments prove very useful in terms of real-time analysis and quick response to users, they present a number of challenges. From the system-perspective, these systems are dynamic, continuous, and contain a high volume of sources and sinks [29]. The dynamism involves the deletion or creation of sources and sinks at any time, which proves challenging for an analysis to be able to update its information coming from new or old sources and to only respond to active users. The continuity involves the unbounded length of graph streams [12]; that is, there is a constant generation of graphs that need to be analysed quickly enough with no possibility of backtracking over previously arrived graphs [21]. The high volume of sources may create significant amounts of graphs that could affect the system performance when dealing with a high number of users.

From the sources-perspective, these graph streams are coming from multiple sources. For example, information regarding an entity may come by gathering information from more than one sources. Therefore, the data might present high semantic heterogeneity that involves multiple schemata and semantics [29]. For example, different words could describe conceptually similar things (e.g. "energy usage" vs "energy consumption"). Furthermore, due to the possibly high volume of sources and the frequent sampling rate of graph streams, graphs might contain identical information for consecutive periods of time resulting in duplication [24]. Conceptual similarity and duplication result in redundant information. This redundancy may lead to significant propagation, storage overheads of unnecessary graphs within a network and slower processing time [3].

From the sinks-perspective, users may have various levels of ambiguity and expressibility of their needs. Ambiguity [4] involves the different user contextual interpretations of the graphs produced based on their needs or their perception. Expressibility [11] is the ability of users expressing their needs; that is, either how satisfying their level of prior understanding of these needs is to create queries with specific filters or how good their technical ability is to create queries that involve the use of complex languages (like SPARQL). Sometimes depending on the interest of the user, it is difficult for them to create an appropriate query. For example, if a user is interested in information that is linked to graphs coming from multiple sources, the user may need to define complex join queries to gather it from all these sources. This complexity may lead to low usability as the user is expected to be aware of complex query languages and the schemata of several graph sources. On the other hand, if the user creates a simple keyword-based query, although this may lead to high usability, it may create an abundance of redundant information for the user [36].

Therefore, an appropriate graph stream analysis system needs to be defined that can overcome the above challenges efficiently and effectively [26]. This system needs to be able to analyse the graphs in a quick, efficient, and expressive way in real-time to answer multiple continuous queries [12]. This can be realised with a system that will abstract the users from the underlying real-time graph analysis and the need for complex queries (high usability), it will provide to the users expressive answers (non-redundant), and it will use limited resources [30].

In previous work [27], we proposed a dynamic diverse summarisation system of heterogeneous graph streams with the use of embeddings, as approximate solutions [19] are acceptable as quick answers [1] within a small error range with high probability while using limited resources. In this way, we aspired not only to increase the system performance by reducing the number of graphs sent upstream but also to create a top-k diverse conceptual graph set that will not overwhelm the user with redundant information. In this paper, we extend the evaluation of our approach by adapting FACES [14], an existing static diverse summarisation methodology, in our dynamic system and we extend our evaluation results by taking into account the comparison between embeddings-based and thesaurus/ontology-based approaches by examining their trade-offs between latency, throughput, memory footprint, the number of forwarded messages, and expressiveness by using a range of windowing policies.

Our main contributions are:

- A user-friendly query that allows users to receive top-k diverse filtered information using a specific window type with a desired window size and window slide.
- Introduce a novel dynamic diverse summarisation system for heterogeneous graph stream windows with the use of embeddings based on user query relevance, importance, and diversity.
- Adapting FACES [14], an existing static diverse entity summarisation methodology with the use of a combination of a thesaurus and an ontology, in a dynamic environment using windows.
- An extensive evaluation comparison between embeddings-based and thesaurus/ontology-based approaches by examining their trade-off between latency, throughput, memory footprint, the number of forwarded messages, and expressiveness by using a range of windowing policies.

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

2 Problem Analysis

Some background, the motivational scenario and its challenges are analysed below.

2.1 Background

Background is given on knowledge graphs, entity summarisation and definitions that will be used throughout the paper.

Knowledge Graphs Sensor data streams could be represented as conceptual entities with their associated relations via graphs. This structure could lead to a strong feature for information searching and query processing [24].

Entities are real-world or abstract things and the information linked to them can be represented by knowledge graphs [32]. Within knowledge graphs the nodes represent the entities, and the directed labelled arcs constitute relations among them. In Fig. 1 a knowledge graph of the entity *Rice University* is given. Within that, *Rice University*, *Texas*, *United States*, *Houston*, and *Division I (NCAA)* are entities, whereas *15°C* and *2kWh* are literals (e.g. strings or dates). The *temperature*, *energy usage*, *state*, *country*, *city*, and *athletics* are relations among the connected entities or literals by the directed arc.

In Fig. 1 there is also an additional relation of *type*. The type-based properties of an entity are a category or class that the entity belongs to. For example, the entity *Rice University* is an *Educational Institute*. More than one entity can have the same type and an entity can have more than one type, either general or fine-grained ones. Types are the kind of information that can cluster group of entities with the same properties [17]. Entity typing is the process of assigning a type to an entity, and it is usually provided in popular knowledge bases, like DBpedia[1].

Resource Description Framework (RDF) is a data modelling language that represents the knowledge 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 (Fig. 1).

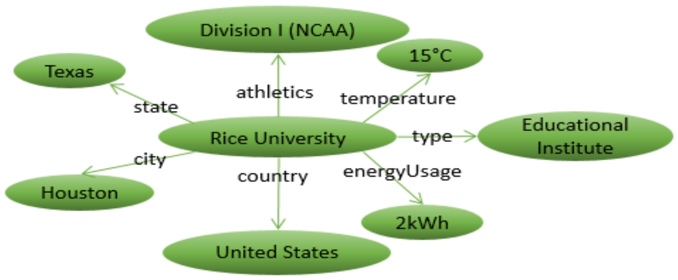


Fig. 1. The knowledge graph of *Rice University*.

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

Entity Summarisation There are two types of summaries; extractive and abstractive [34]. The extractive summary contains part of the original information; that is, in extractive entity summarisation the summary contains a subset of the entity's knowledge graph. The abstractive summary contains information that is not found in the original one, that is in abstractive entity summarisation the summary contains newly generated triples (e.g. aggregated triples).

The entity summarisation can also be split into two subtypes; relevance-centred and diversity-centred [34]. Relevance-centred summaries focus on specific relations that are important and more relevant to the entity in question. Repetition of the same property could occur in this summary. Diversity-centred summaries focus on a more diverse coverage of the information of an entity. Repetition of the same properties is avoided in this summary.

In this paper, we focus on extractive diversity-centred summaries. Therefore, 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 [32].

Definitions By adopting and adapting definitions that were introduced in Cheng et al. [6], 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, nor that they are aware of the semantics or schemata of the graph streams. Therefore, a query should ideally be a keyword-based one [32]. 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 type, window size, window slide, ranking policy⟩ defined as $DAQ = \langle e, k, wt, ws, wsl, r \rangle$, where $e \in E$, $k \in \mathbb{N}$, $wt \in \{CountTumbling, CountSliding, TimeTumbling, TimeSliding\}$, $ws \in \mathbb{N}$, $wsl \in \mathbb{N}$ and $r \in \{Diversity, None\}$.

2.2 Motivational Scenario

Imagine Houston is a smart city (shown in Fig. 2), 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.

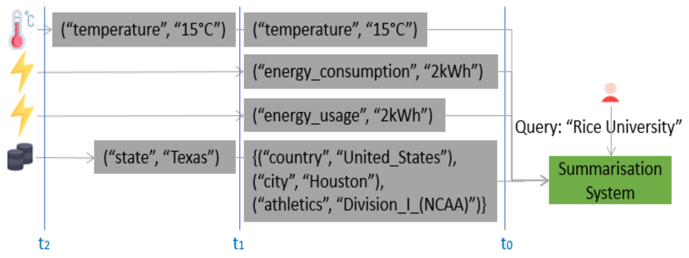


Fig. 2. A user is interested in information about *Rice University* and sources generate timestamped information records. Here, the *temperature* information is duplicate and the *energy_usage* and *energy_consumption* are conceptually similar.

2.3 Problem Challenges

In the aforementioned scenario, the sensor readings could be represented as structured graphs, like in Fig. 1. Nevertheless, this 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. 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 data semantics and schemata 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.
- **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 [21].
- **High data volume**: The high volume of information that is created by the sources needs to be properly filtered to not overwhelm the user.

3 Approach

The architecture and algorithms are analysed below.

3.1 Architecture

Our architecture is illustrated in Fig. 3. 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.

All graph streams enter the *Window Partitioning* that creates *Windows* of different types. The type of window can be one from Count Tumbling, Count Sliding, Time Tumbling, or Time Sliding. Each window is associated with a user-required entity; that is, the number of total windows equals to the number of different entities contained in the user queries. Then, each window fuses the available data by being populated with triples from all sources concerning a specific entity. The triples are further processed, depending on the approach taking place.

In the Embeddings-based approach, the triples go through the *Triple Preprocessor*, where all of their content is extracted and pre-processed. Duplicate triples are discarded at this stage. A *Word2Vec Model* [23] 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 undergo *DBSCAN Clustering* [9]. 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 from 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 by either sliding the window according to the user-defined window slide wsl or by creating a new window with size equal to the user-defined window size ws.

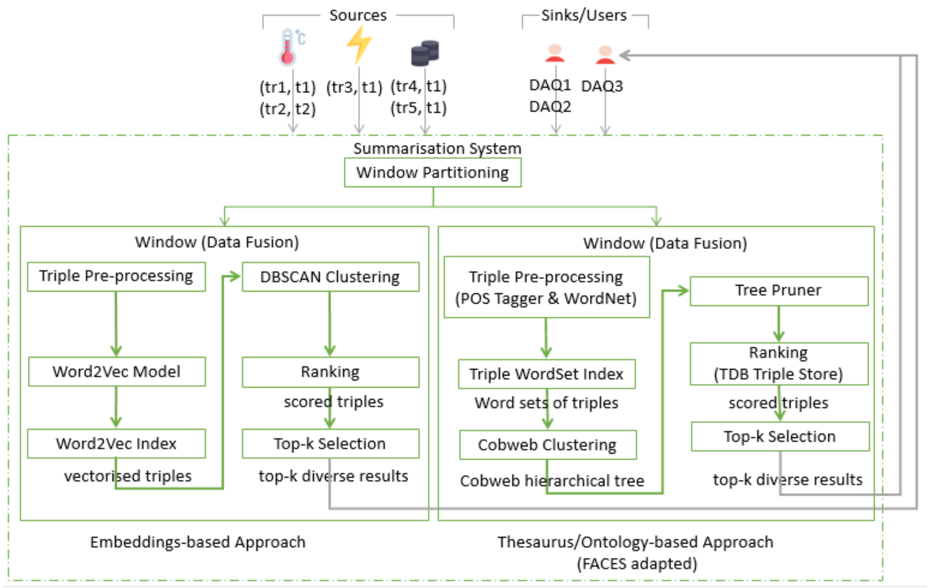


Fig. 3. Architecture of the Dynamic Diverse Summarisation System by using two approaches; an Embeddings-based one and a Thesaurus/Ontology-based one.

In the Thesaurus/Ontology-based approach (FACES [14] adapted), the triples go through the *Triple Pre-processor*, where all of their content is extracted and pre-processed with the use of a thesaurus WordNet[2] and a POS Tagger. Duplicate triples are discarded at this stage. A *Triple WordSet Index* is then created that contains the word sets related to each triple. Once the window reaches its full capacity based on the user-defined window size ws, all word sets undergo *Cobweb Clustering* [10]. A hierarchical tree is then created that contains conceptual clusters and is then pruned via the *Tree Pruner* based on algorithm-defined parameters. The triples in the tree are ranked through *Ranking* with the use of a TDB Triple Store[3] based on the importance of the triples in a knowledge base. The top-k triples are then selected from 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 as described above, depending on the type of window.

Windowing Policies

Window Lifecycle The lifecycle of windows implemented in our dynamic diverse summarisation system is inspired by that of Flink [5], but customised for our system requirements. There are the following functions taking place for a window:

– **Window Creator**: A window is created based on the entities that have been generated by the sources and requested by the users.

[2] https://wordnet.princeton.edu/
[3] https://jena.apache.org/documentation/tdb/

- **Window Assigner**: The window is populated with triples coming from multiple sources that are related to the specific entity the window is linked to.
- **Window Processor**: The triples within the window are fused and processed based on the approach selected.
- **Trigger**: The trigger specifies the conditions under which the window's elements are considered ready to be processed. In the case of triple pre-processing and creation of indices functions, these get triggered on a per-element basis that enters the window (incremental). The clustering, ranking and top-k selection get triggered once the window is full (batch).
- **Evictor**: The evictor is responsible for removing elements from the window after the processing has been done. Depending on the window type, the evictor will delete all elements from the window or a subset depending on conditions.

Window Types We have implemented the following window types in our dynamic diverse summarisation system:

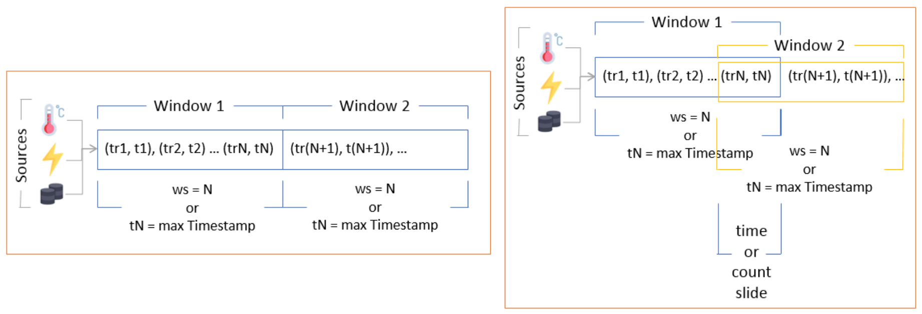
- **Count Tumbling Window**: This window contains triples up to a specific user-defined maximum count/size (ws). Then, a new window of the same size is generated with newly arrived triples.
- **Count Sliding Window**: This window contains triples up to a specific user-defined maximum count/size (ws). Then, a new window of the same size is generated by sliding up to a user-defined slide (wsl). This means that the new window contains the last triples of the previous window equal to the specified slide in number along with newly arrived triples.
- **Time Tumbling Window**: This window contains triples generated before a specific user-defined maximum timestamp (ws). Then, a new window of the same size is generated with newly arrived triples.
- **Time Sliding Window**: This window contains triples generated before a specific user-defined maximum timestamp (ws). Then, a new window of the same size is generated (for the next timestamp period) by sliding up to a user-defined time slide (wsl). This means that the new window contains the last triples of the previous window that were generated during the time span of the slide along with newly arrived triples that are generated before the new maximum timestamp (next timestamp period).

The different window types and their policies are illustrated in Fig. 4.

3.2 Algorithms

The comparison between the Thesaurus/Ontology-based approach and the Embeddings-based one is analysed below.

Thesaurus/Ontology-based approach (FACES-adapted) FACES is divided into two stages: 1) hierarchical conceptual clustering of triples and 2) ranking of triples. The first stage is done by an adapted version of Cobweb



(a) Count Tumbling or Time Tumbling Windows

(b) Count Sliding or Time Sliding Windows

Fig. 4. The window policies for Count/Time Tumbling windows and for Count/Time Sliding windows.

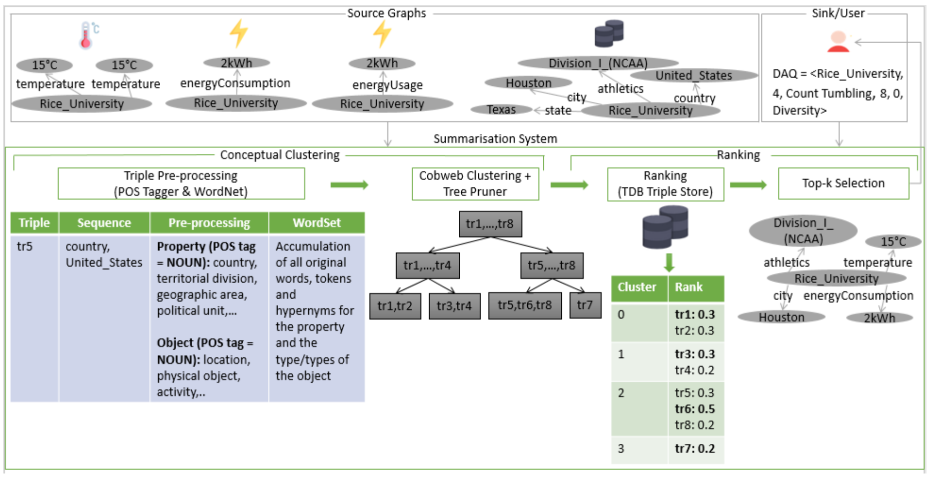
clustering [10] and the use of WordNet[4] and a POS Tagger for pre-processing. The second stage is a combination of an adapted version of tf-idf that is based on the popularity and the informativeness of a triple in a knowledge base and some pre-defined top-k selection rules. A simplified illustration of the algorithm is provided in Fig. 5. For more details, the reader is directed to Gunaratna et al. [14].

In the implementation, we use for the RDF models and some processing Apache Jena[5], a POS Tagger of OpenNLP[6] and WordNet of extJWNL[7]. The adapted version of the Cobweb algorithm is implemented by modifying the CobWeb algorithm of MOA[8]. The adapted version of tf-idf is implemented by using a TDB Triple Store[9] that stores part of the DBpedia ontology.

Embeddings-based approach Our Embeddings-based approach is divided into two stages: 1) conceptual clustering of triples and 2) ranking of triples. The first stage is done by the combination of Word2Vec models [23] and DBSCAN clustering [9]. The second stage is a combination of similarity metrics and some pre-defined top-k selection rules. A simplified illustration of the algorithm is provided in Fig. 6. For more details, the reader is directed to Pavlopoulou et al. [27].

In the implementation, we use the pre-trained GoogleNews, Vectors, and VectorsPhrase models[10]. For the RDF models and some processing, we use Apache

[4] https://wordnet.princeton.edu/
[5] https://jena.apache.org/
[6] https://opennlp.apache.org/
[7] http://extjwnl.sourceforge.net/
[8] https://moa.cms.waikato.ac.nz/
[9] https://jena.apache.org/documentation/tdb/
[10] https://code.google.com/archive/p/word2vec/



Triple	Sequence	Pre-processing	WordSet
tr5	country, United_States	**Property (POS tag = NOUN):** country, territorial division, geographic area, political unit,... **Object (POS tag = NOUN):** location, physical object, activity,..	Accumulation of all original words, tokens and hypernyms for the property and the type/types of the object

Cluster	Rank
0	**tr1: 0.3** tr2: 0.3
1	**tr3: 0.3** tr4: 0.2
2	tr5: 0.3 **tr6: 0.5** tr8: 0.2
3	**tr7: 0.2**

Fig. 5. The sources of Fig. 2 generate information that is represented as structured graphs. A user asks for the top-4 diverse information items for *Rice University* from a Count Tumbling 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 word sets are extracted. Cobweb clustering is then performed that takes as input the appearance or not of words in each triple from the whole collection of the word set. A tree is then created that contains leaves representing the conceptual clusters. Then, the triples are ranked for each cluster based on their informativeness and popularity within DBpedia that is stored in a TDB Triple Store. The bold triples are the most highly ranked ones for each cluster. The top-4 triples are selected and sent as a 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}.

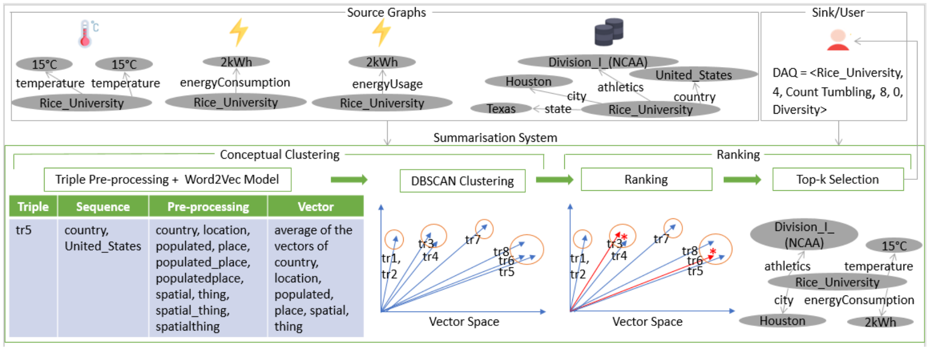
Jena[11], for the Word2Vec models we use deeplearning4j[12] and for DBSCAN we use Smile[13].

4 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 that uses three Word2Vec models (GoogleNews, Vectors, and VectorsPhrase) with the FACES-adapted and 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 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.

[11] https://jena.apache.org/
[12] https://deeplearning4j.org/
[13] https://haifengl.github.io/smile/nlp.html



Triple	Sequence	Pre-processing	Vector
tr5	country, United_States	country, location, populated, place, populated_place, populatedplace, spatial, thing, spatial_thing, spatialthing	average of the vectors of country, location, populated, place, spatial, thing

Fig. 6. The sources of Fig. 2 generate information that is represented as structured graphs. A user asks for the top-4 diverse information items for *Rice University* from a Count Tumbling 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 a 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}.

4.1 Dataset and Typing Information

The FACES dataset[14] has been selected for our evaluation, which is based on DBpedia 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.

In the *Background* section, we referred to the type-based properties and their role in an RDF graph. Both approaches use the typing information of the objects. In order to get our results, we take for granted that the typing information of each object in DBpedia is available and correct. Unfortunately, this is not the case for all object resources. For example, in the FACES dataset after extracting all available typing information from DBpedia, 54 of the object resources contained noisy/false types, like $< Automobiles >< type >< MusicGenre >$ or $< Chemistry >< type >< University >$ and around 300 of them had missing types. In order to solve this issue, we proceeded with a combination of the approaches described below:

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

- We used the Virtuoso SPARQL Query Editor of DBpedia[15] by creating a SPARQL query that asked for the type of an object resource, in case there was some missing/additional information not found in the dataset files.
- Inspired by [8], we used the pre-trained entity vectors with naming from the Freebase model[16] and we extracted the word vectors of entities. The concept is that if entities with unknown types are closer in the vector space to entities with known types, then they will share the same entity sets. We used this model and not another one because it had high chances of containing entities that related to specific names, places, etc. For example, $< 100\_metres >$ and $< 200\_metres >$ were clustered together, so we gave to both of them the type $< SportsEvent >$ that $< 200\_metres >$ had. There might be cases though where $< Baywatch >$, $< Baywatch\_Nights >$ and $< David\_Hasselhoff >$ were clustered together, and although contextually it made sense, in reality, their types are different. This information could be used though for fine-grained entity typing. For example, each of the entities $< Angelina\_Jolie >$, $< Ben\_Affleck >$ and $< Courteney\_Cox >$ had types $< Agent >$, $< NaturalPerson >$ and $< Person >$, but a more fine-grained type could be $< Actor >$ since they were clustered together. This is, nevertheless, out of the scope of our paper.
- Some additional dataset files were extracted for each entity in question from DBpedia, like "Categories", "Short Abstracts" and "wikiPageWikiLinks" that can potentially help in the extraction of types.
- If none of the above succeeded because an entity was missing, then a manual type was given. For example, the entities similar to $< http://wifo5-03.informatik.uni\text{-}mannheim.de/flickrwrappr/photos/Texas >$ were given the type $< Photos >$.

A sophisticated way on how to do typing information in a streaming fashion is out of the scope of this paper and it could be pursued as future research.

4.2 Workloads

One sink and 50 sources are used. The sink generates 50 queries, one for each entity, and the sources are each responsible for generating a stream related to only one entity.

The streaming rate of the sources is constant. The selection of which triple will be generated each time from a source follows one of the following distributions:

- **TakeAll**: A source generates all unique triples that are available for an entity in the dataset.
- **Zipf**: A source generates a triple that is available for an entity in the dataset based on popularity. Specifically, each triple in a collection is checked for its popularity by combining the frequency of occurrence of the property and the object in the collection. Then the selection of which triple a source will generate follows a Zipf distribution [18] based on this popularity.

[15] http://dbpedia.org/sparql
[16] https://code.google.com/archive/p/word2vec/

4.3 Metrics

Several metrics have been used to evaluate the efficiency and effectiveness of our approach. These include correctness (the agreement, quality, and redundancy-aware F-score), end-to-end latency, number of messages, memory footprint and throughput.

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.

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 [6], and we adapted a time-dependent version of their quality metric defined below.

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

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

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.

Redundancy-aware F-score We are using the metrics of redundancy-precision and redundancy-recall defined in [35], 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:

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

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

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.

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.

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

Memory Footprint This metric contains the memory used by the different approaches not during the run, but for the use of Word2Vec models or ontologies.

Throughput Throughput is the number of triples the system is able to analyse in a specific amount of time.

4.4 Results

The results are shown below for 50 sources that generate 100 triples each and 1 sink with 50 queries with different window policies and distributions. For the embeddings-approach, the DBSCAN parameters are $\varepsilon = 1$ and $minPts = 1$ and the Euclidean distance is used for ranking. For FACES-approach, we use the parameters of the original paper [14] with $Cobweb - cut - off = 5$ and $Cobweb{-}path{-}level = 3$. For the Zipf distribution, we use $Zipf{-}exponent = 1$.

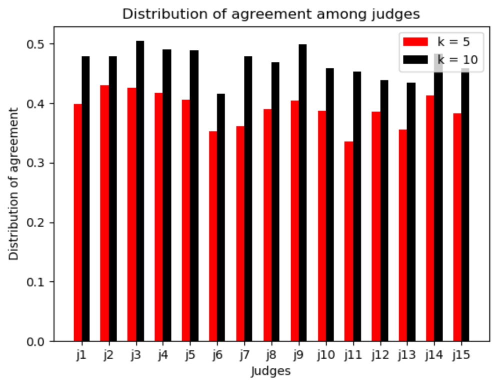


Fig. 7. Agreement distribution among judges.

Agreement The average agreement results for all entities 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. In Fig. 7, we observe the average agreement distribution for all entities among judges for the different k values. As expected, the values for $k = 5$ are lower than that of $k = 10$, as the less the number of triples, the less probable an agreement is. This shows that when a user is presented with a smaller summary, then stricter criteria take place of what this ideal summary should be. Some judges, like judges 6, 7, and 11 have the lowest agreement with the other judges for $k = 5$, while for $k = 10$, judges 6, 12, and 13 have the lowest one. This proves the different levels of ambiguity and expressibility of their needs. More specifically, the different user contextual interpretations of the summaries produced and the importance of them based on their needs are proven. There is though some common ground, which is shown in the agreement values.

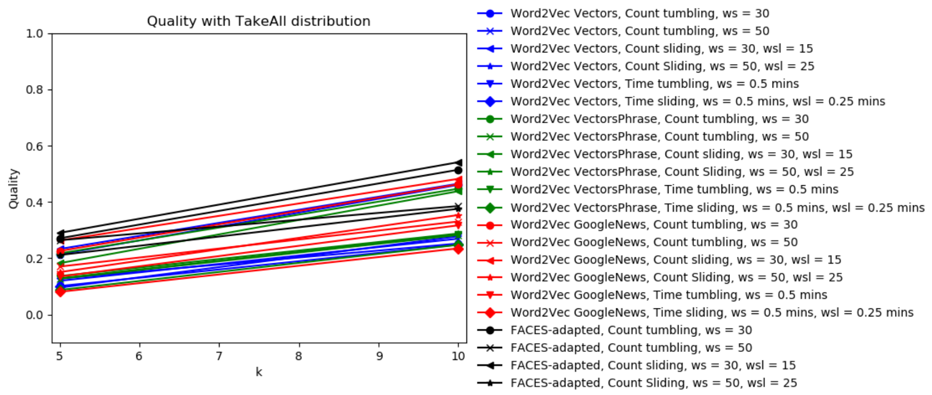


Fig. 8. Quality for top-k approaches.

Quality In Fig. 8 the quality is illustrated only for TakeAll distribution as we wanted to cover all unique triples of each entity. Zipf distribution would select mostly popular triples without possibly covering all of them, so it is not tested in this metric. Also, the time windows are not checked for the FACES-adapted approach as it was slow, so no notification was extracted for this time window period.

Fig. 8 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 follows the consensus among the ideal summaries that the judges gave. We also observe that the FACES-adapted approach is the best one, followed by the Word2Vec models with GoogleNews being the best Word2Vec model followed by VectorsPhrase and Vectors. Nevertheless, there is not a big difference between the quality values of the FACES-adapted and the GoogleNews model.

In terms of windows, the quality gets better for smaller windows as they contain fewer triples compared to bigger ones, so the commonality between generated triples and ideal ones is less probable. We also observe that sliding windows have better quality than tumbling ones, and time windows behave more poorly than count windows.

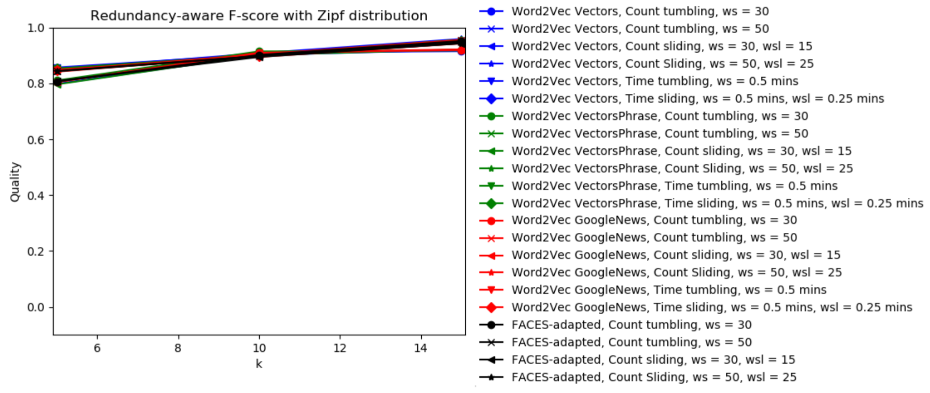


Fig. 9. Redundancy-aware F-score for top-k approaches.

Redundancy-aware F-score Redundancy-aware F-score in Fig. 9 is illustrated only for Zipf distribution as TakeAll distribution does not generate duplicate triples. We observe 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 0.80 to 0.95. Lower F-score occurs for lower k as stricter content filtering is taking place. There is not much difference among the window policies, although we see that a higher F-score is observed with the increase in window sizes, as the bigger the window, the more probable redundant information exists. The F-scores are very high, mostly because there is a lot of duplication in the generated triples due to Zipf distribution. The less duplication exists in the generated streams, the lower the F-score will be.

End-to-End Latency In Fig. 10, we observe the end-to-end latencies for count windows only, as the time windows had similar behaviour. We see that the slowest model is FACES-adapted, followed by Word2Vec GoogleNews, Word2Vec VectorsPhrase, Word2Vec Vectors, and non-top-k fused approach. This shows that FACES-adapted spends much time in pre-processing the triples and accessing the TDB triple store every time for ranking the triples. The Word2Vec GoogleNews model is the most expensive of the three Word2Vec models as it has 3.5GB memory that needs, on average, 139518ms to be loaded once in our system (included in latency). The other Word2Vec models are much smaller (see Fig. 12) and need far less time. The non-top-k fused approach has the best latency since no processing is involved when sending notifications, but it is not

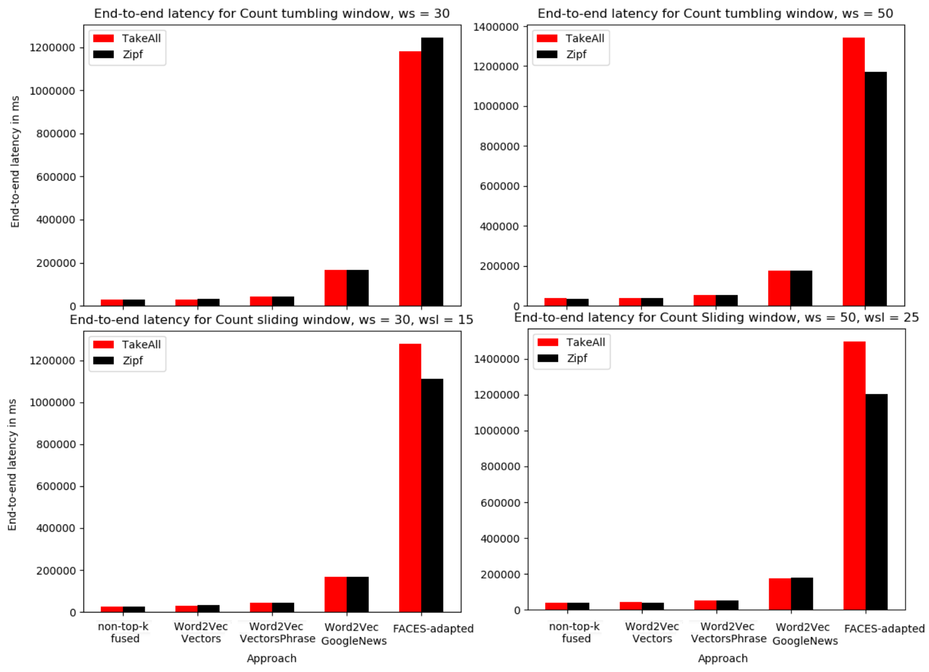


Fig. 10. End-to-End Latency for non-top-k and top-k approaches.

that much quicker compared to the smaller Word2Vec models with Vectors being very close to it.

Other observations include that the Zipf distribution results in lower latency compared to the TakeAll one. Also, the latency increases with the window size 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. Latency is independent of k as the processing is done, and then only the top-k selection of the ranked triples takes place.

The reason the latencies seem large is that 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.

Number of Messages In Fig. 11, the number of forwarded messages for the TakeAll and Zipf distribution are illustrated. For the TakeAll, we observe that the number of forwarded messages is reduced within the ranges of 34% to 92% depending on the k and the window policy for the top-k approach compared to the non-top-k one. For higher k values, more information is sent; therefore, the message reduction decreases. Smaller windows create more messages as they get populated more quickly with triples, so more regular notifications are sent. Sliding windows create more messages compared to tumbling ones of the same window size as more frequent windows are created due to the slide, so more

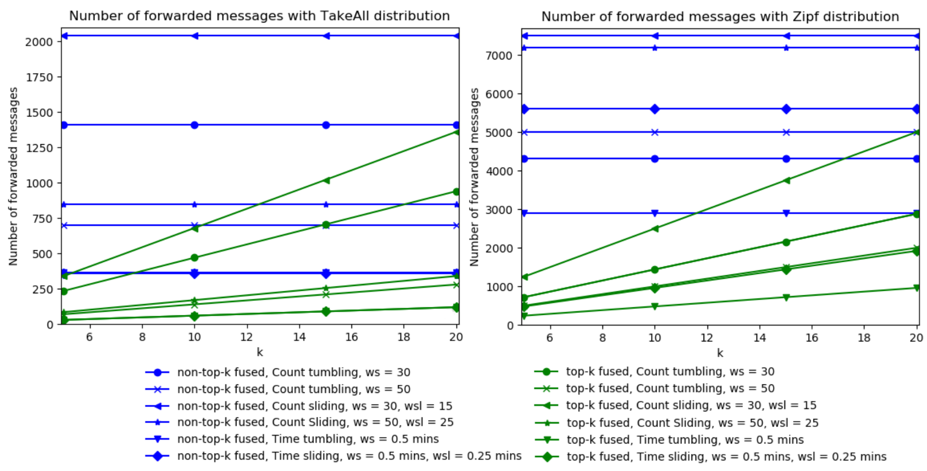


Fig. 11. Number of messages forwarded for non-top-k and top-k approaches.

notifications are sent. This is also the reason why smaller size windows with smaller slides produce more messages.

The Zipf distribution's results bear similar observations to those of TakeAll, although Zipf creates more messages in total, as repetitive triples may be produced. In TakeAll, only the unique triples are generated by the sources. Therefore, for the time windows, TakeAll has a similar number of messages for the tumbling or sliding window as at some point the sources stop generating any more streams; therefore, more time will not have any effect. On the other hand, for Zipf we see that the time sliding window produces more messages as again more frequent windows are created due to the slide, so more notifications are sent.

TakeAll does not produce repetitive triples, but for Zipf, we observe that from all messages, non-top-k approach contains 56.7% to 69.3% duplicates, depending on the window policy. This percentage is particularly high in the case of Zipf, as popular triples will be produced more frequently than others. Smaller windows have less duplication, and sliding windows have lower duplication than their equivalent tumbling windows. The top-k approach can discard this duplicate information, therefore, reducing the overall forwarded messages.

Memory Footprint In Fig. 12, we observe the memory footprint not of the approaches while they are being executed, but on the models or ontologies that they are using. We observe that FACES-adapted demands more memory compared to the Embeddings-based approaches. This occurs because FACES-adapted uses the whole DBpedia in order to rank the available triples based on popularity and informativeness. For that reason, we used a TDB Triple Store that demanded extra memory cost and several accesses for each summary extraction during execution. The memory cost indicated in Fig. 12 only contains part of the whole DBpedia that contained the format of triples in our original dataset. The memory of storing all of the DBpedia would be higher. In terms

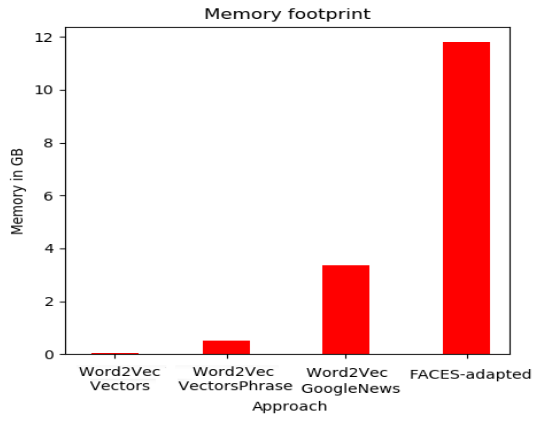


Fig. 12. Memory footprint for top-k approaches.

of the Word2Vec models, the GoogleNews model is trained on a bigger corpus; therefore, it is the biggest one among the Word2Vec models. This model is followed by VectorsPhrase and Vectors, which is the smallest model as it is trained on a small corpus. Non-top-k fused approach is not shown here as it did not demand any extra memory cost as it does not use any Word2Vec models or ontologies.

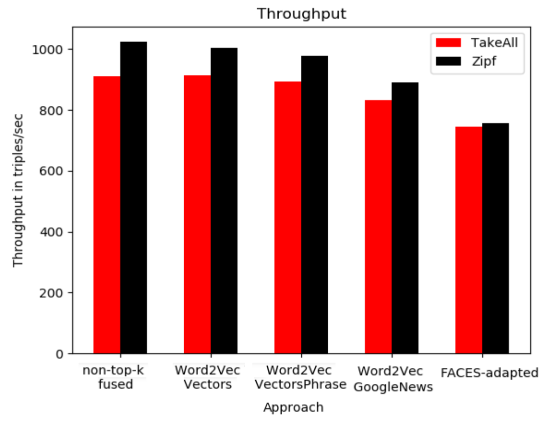


Fig. 13. Throughput for non-top-k and top-k approaches.

Throughput In Fig. 13, the average throughput for all window policies for each approach is shown as not important differences were observed. We see that the TakeAll distribution has lower throughput than the Zipf, probably because in the latter much more events were generated, so they entered the system more frequently. Non-top-k fused along with Word2Vec Vectors have the best throughput, followed by Word2Vec VectorsPhrase, Word2Vec GoogleNews, and lastly FACES-adapted. This behaviour is dependent or analogous to the memory consumption and end-to-end latency for each approach. As FACES-adapted is the most memory, and latency costly approach; therefore, the throughput will be the lowest. In terms of windows, bigger ones have higher throughput by a few events, whereas sliding windows have lower throughput to tumbling ones of the same size by a few events.

Discussion According to our results, we conclude that non-top-k fused approach is the best one in relation to memory, throughput, and latency. Nevertheless, it sends all the available information to the user that contains duplicate or conceptually redundant information. With the increase of sources, more and more data is generated with different variations of duplication or conceptual similarity; therefore, the non-top-k fused approach would perform worse. On the other hand, the top-k fused approach reduces significantly the amount of data that is sent as a summary to the user. This means that the information sent to the user might not contain some non-redundant information due to filtering, but it manages to send a summary with quality analogous to the agreement among judges.

The worst model in terms of memory, throughput, and latency was FACES-adapted, although it performed slightly better than the rest of the approaches in terms of quality of summaries. This shows that an existing thesaurus might be strict when it comes to synonyms or hypernyms, whereas a probabilistic model based on text-corpora, like Word2Vec models, covers a wider range of synonyms based on context. For example, semantically opposite words (antonyms) but conceptually similar (e.g. death place - birthplace) are taken into account, as well as phrases. Also, this shows that finding all possible hypernyms and use a memory-heavy ontology in real-time for ranking can significantly decrease the performance of the system.

In conclusion, we observe a trade-off between latency, throughput, memory footprint, the number of forwarded messages, and expressiveness (represented by the quality and redundancy-aware F-score) among a non-top-k and a top-k fused approach. This is because the latency, throughput, and memory are better in non-top-k fused approach, but the number of forwarded as well as redundant messages and expressiveness are worse and vice verse for the top-k. We also observe that a thesaurus/ontology-based top-k approach might be slightly better in terms of quality of summaries compared to an embeddings-based top-k approach, but it behaves worse in terms of system performance. We conclude, then, that slightly more processing time, memory and throughput for finding diverse data with the use of embeddings-based approaches, can lead to less data being sent upstream for further processing, as well as data that is seen as expressive as the agreement among judges without containing redundant information (duplicate or conceptual one).

5 Related Work

The related work is analysed below and it is split into the Non-Streaming and Streaming category.

5.1 Non-Streaming

Graph Summarisation Many graph summarisation techniques are covered by Liu et al. [22]. 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.

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 [32] 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 [14] 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. [28] 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. [16] and Pan et al. [25] 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.

5.2 Streaming

Stream Processing Frameworks Existing stream processing frameworks, like Apache Spark[17], Flink[18] and Kafka[19] 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.

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. [2] and Gupta et al. [15]. Tang et al. [33] 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. [7] 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.

[17] https://spark.apache.org/
[18] https://flink.apache.org/
[19] https://kafka.apache.org/

6 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. In a previous paper [27], we proposed a dynamic diverse summarisation system of heterogeneous graph streams with the use of embeddings. Here, we examine the performance comparison between our embeddings-based approach, FACES-adapted, an existing thesaurus/ontology-based approach (FACES) that we adapted in a dynamic environment with the use of windows and data fusion and non-top-k fused approach. According to our findings, although the non-top-k approach is better in system performance, it contains redundant information. The top-k fused approaches, on the other hand, are more costly for the system but create good quality non-redundant summaries based on judges. The FACES-adapted might be slightly better in terms of quality of summaries compared to our approach, but it behaves worse in terms of system performance.

Future work will be related to improving the quality metric results by boosting expressiveness with a more sophisticated ranking approach. Also, adapting more existing static diverse entity summary approaches to smart environments will be explored. An IoT-based dataset will also be examined, and its performance will be analysed on our approach. The approach will also 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. Finally, the extension of simple user queries to preferential or complex ones, like domain-specific or feature-specific or multi-entity will also be explored.

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. Aggarwal, C.C.: Data streams: models and algorithms, vol. 31. Springer Science & Business Media (2007)
2. Aggarwal, C.C., Philip, S.Y.: A survey of synopsis construction in data streams. In: Data Streams, pp. 169–207. Springer (2007)
3. Ahmed, E., Rehmani, M.H.: Mobile edge computing: opportunities, solutions, and challenges (2017)
4. Barnaghi, P., Wang, W., Henson, C., Taylor, K.: Semantics for the internet of things: early progress and back to the future. International Journal on Semantic Web and Information Systems (IJSWIS) **8**(1), 1–21 (2012)
5. Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., Tzoumas, K.: Apache flink: Stream and batch processing in a single engine. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering **36**(4) (2015)
6. Cheng, G., Tran, T., Qu, Y.: Relin: relatedness and informativeness-based centrality for entity summarization. In: International Semantic Web Conference. pp. 114–129. Springer (2011)

7. Dia, A.F., Kazi-Aoul, Z., Boly, A., Chabchoub, Y.: C-sparql extension for sampling rdf graphs streams. In: Advances in Knowledge Discovery and Management, pp. 23–40. Springer (2018)
8. van Erp, M., Vossen, P.: Entity typing using distributional semantics and dbpedia. In: International Semantic Web Conference. pp. 102–118. Springer (2016)
9. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd. vol. 96, pp. 226–231 (1996)
10. Fisher, D.H.: Knowledge acquisition via incremental conceptual clustering. Machine learning **2**(2), 139–172 (1987)
11. Furnas, G.W., Landauer, T.K., Gomez, L.M., Dumais, S.T.: The vocabulary problem in human-system communication. Communications of the ACM **30**(11), 964–971 (1987)
12. Gama, J.: A survey on learning from data streams: current and future trends. Progress in Artificial Intelligence **1**(1), 45–55 (2012)
13. Gunaratna, K., Thirunarayan, K., Sheth, A., Cheng, G.: Gleaning types for literals in rdf triples with application to entity summarization. In: International Semantic Web Conference. pp. 85–100. Springer (2016)
14. Gunaratna, K., Thirunarayan, K., Sheth, A.P.: Faces: Diversity-aware entity summarization using incremental hierarchical conceptual clustering. In: AAAI. pp. 116–122 (2015)
15. Gupta, N., Rajput, I.: Stream data mining: a survey. Indrjeet Rajput Int. J. Eng. Res. Appl. IJERA ISSN pp. 2248–9622 (2013)
16. Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K.U., Umbrich, J.: Data summaries for on-demand queries over linked data. In: Proceedings of the 19th international conference on World wide web. pp. 411–420. ACM (2010)
17. Jin, H., Hou, L., Li, J., Dong, T.: Attributed and predictive entity embedding for fine-grained entity typing in knowledge bases. In: Proceedings of the 27th International Conference on Computational Linguistics. pp. 282–292 (2018)
18. Kaoudi, Z., Miliaraki, I., Koubarakis, M.: Rdfs reasoning and query answering on top of dhts. In: International Semantic Web Conference. pp. 499–516. Springer (2008)
19. Lakshmi, K.P., Reddy, C.: A survey on different trends in data streams. In: Networking and Information Technology (ICNIT), 2010 International Conference on. pp. 451–455. IEEE (2010)
20. Le-Phuoc, D., Parreira, J.X., Hauswirth, M.: Linked stream data processing. In: Reasoning Web International Summer School. pp. 245–289. Springer (2012)
21. Liu, H., Lin, Y., Han, J.: Methods for mining frequent items in data streams: an overview. Knowledge and information systems **26**(1), 1–30 (2011)
22. Liu, Y., Safavi, T., Dighe, A., Koutra, D.: Graph summarization methods and applications: A survey. ACM Computing Surveys (CSUR) **51**(3), 62 (2018)
23. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
24. Pacha, S., Murugan, S.R., Sethukarasi, R.: Semantic annotation of summarized sensor data stream for effective query processing. The Journal of Supercomputing pp. 1–23 (2017)
25. Pan, J.Z., Gómez-Pérez, J.M., Ren, Y., Wu, H., Zhu, M.: Ssp: compressing rdf data by summarisation, serialisation and predictive encoding (2014)

26. Pavlopoulou, N., Curry, E.: 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)
27. Pavlopoulou, N., Curry, E.: Using embeddings for dynamic diverse summarisation in heterogeneous graph streams. In: 2019 First International Conference on Graph Computing (GC). IEEE (2019)
28. Pouriyeh, S., Allahyari, M., Kochut, K., Cheng, G., Arabnia, H.R.: Combining word embedding and knowledge-based topic modeling for entity summarization. In: 2018 IEEE 12th International Conference on Semantic Computing (ICSC). pp. 252–255. IEEE (2018)
29. Qin, Y., Sheng, Q.Z., Falkner, N.J., Dustdar, S., Wang, H., Vasilakos, A.V.: When things matter: A survey on data-centric internet of things. Journal of Network and Computer Applications **64**, 137–153 (2016)
30. Satyanarayanan, M., Simoens, P., Xiao, Y., Pillai, P., Chen, Z., Ha, K., Hu, W., Amos, B.: Edge analytics in the internet of things. IEEE Pervasive Computing (2), 24–31 (2015)
31. Subramaniam, S., Gunopulos, D.: A survey of stream processing problems and techniques in sensor networks. In: Data Streams, pp. 333–352. Springer (2007)
32. Sydow, M., Pikuła, M., Schenkel, R.: Diversum: Towards diversified summarisation of entities in knowledge graphs. In: Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on. pp. 221–226. IEEE (2010)
33. Tang, N., Chen, Q., Mitra, P.: Graph stream summarization: From big bang to big crunch. In: Proceedings of the 2016 International Conference on Management of Data. pp. 1481–1496. ACM (2016)
34. Thalhammer, A.: Linked data entity summarization (2017)
35. Zhang, Y., Callan, J., Minka, T.: Novelty and redundancy detection in adaptive filtering. In: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 81–88. ACM (2002)
36. Ziegler, C.N., McNee, S.M., Konstan, J.A., Lausen, G.: Improving recommendation lists through topic diversification. In: Proceedings of the 14th international conference on World Wide Web. pp. 22–32. ACM (2005)