VisCARS: Knowledge Graph-Based Context-Aware Recommender System for Time-Series Data Visualization and Monitoring Dashboards

Pieter Moens, Bruno Volckaert, *Senior Member, IEEE*, and Sofie Van Hoecke

***Abstract*—Data visualization recommendation aims to assist the user in creating visualizations from a given dataset. The process of creating appropriate visualizations requires expert knowledge of the available data model as well as the dashboard application that is used. To relieve the user from requiring this knowledge and from the manual process of creating numerous visualizations or dashboards, we present a context-aware visualization recommender system (VisCARS) for monitoring applications that automatically recommends a personalized dashboard to the user, based on the system they are monitoring and the task they are trying to achieve. Through a knowledge graph-based approach, expert knowledge about the data and the application is included as contextual features to improve the recommendation process. A dashboard ontology is presented that describes key components in a dashboard ecosystem in order to semantically annotate all the knowledge in the graph. The recommender system leverages knowledge graph embedding and comparison techniques in combination with a context-aware collaborative filtering approach to derive recommendations based on the context, i.e., the state of the monitored system, and the end-user preferences. The proposed methodology is implemented and integrated in a dynamic dashboard solution. The resulting recommender system is evaluated on a smart healthcare use-case through a quantitative performance and scalability analysis as well as a qualitative user study. The results highlight the performance of the proposed solution compared to the state-of-the-art and its potential for time-critical monitoring applications.**

***Index Terms*—Data visualization, dynamic dashboard, knowledge graph, multiple-view visualization, ontology, visualization recommendation.**

I. Introduction

MONITORING solutions are the backbone of many processes and applications today. Domains such as Industry 4.0, Smart Healthcare and Cloud Computing rely on monitoring data to determine the state of the monitored system. In many state-of-the-art applications, the monitoring data is directly interpreted using Machine Learning (ML) techniques such as Anomaly Detection (AD). In the case of anomalous behavior or a system fault, the root cause of the detected anomaly must be derived. This is often done by a dedicated domain expert, called an operator, who uses data visualizations to gain insights in the monitored data which often consists of real-time sensor observations. In critical applications, this operator is tasked with taking the necessary actions as quickly as possible to retain the healthy state of the system. Real-time dashboard software solutions used for monitoring applications today, e.g. Dynamic Dashboard [1], Grafana [2], Tableau [3] enable the operator to construct monitoring dashboards containing a number of data visualizations in order to communicate all the information required to make timely decisions. To create a dashboard, the operator selects the sensor properties they wish to visualize and chooses the desired option out of a plethora of visualization design choices. This process of manually creating data visualizations, especially during the event of a system fault, is time consuming and costly. To automate this process of creating dashboards, a Visualization Recommender System (VisRS) can be integrated in these dashboards.

State-of-the-art research in VisRS focuses primarily on insight mining in a given dataset, e.g. for analytical tasks and data communication. These datasets are often assumed to consist of static tabular data and the visualization recommendations are based on data analysis (e.g. trend or pattern detection) and data attributes (e.g. statistical features). In recent years, however, the importance of including the user preferences in the recommendation process and moving from purely data-driven recommendation techniques (e.g. statistical analysis, machine learning) towards personalized recommendation techniques has been highlighted. When approaching data visualization recommendation from a system monitoring use-case, these user preferences could be related to the expertise of the end-user. In the domain of Industrial Internet of Things (IIoT), an operator might want to see "simple" colorful visualizations of Key Performance Indicators (KPIs) that immediately highlight whether or not sensor values are deviating from the norm to be alerted as quickly as possible. The engineer tasked with root cause analysis, however, might be interested in more historical data showing the degradation of the system over time. Similarly, in the domain of remote healthcare, an operator needs a global overview of a patient's health values to be able to quickly decide what to do, while nurses dispatched to the patient need fine-grained medical information and healthcare sensor data to optimize their care for the patient.

Manuscript received 16 October 2023; revised 31 May 2024; accepted 7 June 2024. Date of publication 13 June 2024; date of current version 1 August 2025. This work was supported by the Flanders AI Research program and the IMEC, ICON PROTEGO project under Grant HBC.2019.2812. Recommended for acceptance by R. Maciejewski. *(Corresponding author: Pieter Moens.)*

The authors are with the IDLab, Ghent University–IMEC, 9052 Gent, Belgium (e-mail: pieter.moens@ugent.be; bruno.volckaert@ugent.be; sofie.vanhoecke@ugent.be).

Digital Object Identifier 10.1109/TVCG.2024.3414191

So, the visualization design choices, i.e. how the data should be visualized, are thus dependent on the end-user of the dashboard.

Related work in the domain of data visualization has introduced a plethora of visualization design choices and increasingly complex ways to visualize a given dataset, however, the data recommendation remains overlooked. In a monitoring application, the data that should be visualized depends on the state of the monitored system. It is therefore important to move towards Context-Aware Recommender Systems (CARS) that take into account the context in which they are applied. This contextual metadata is specific to the use-case. For example, in the IIoT domain, contextual information includes a description of the physical machine and its components. Alternatively, in the smart healthcare domain, the monitored system can be a patient wearing a smartwatch. Information about this patient provides the contextual metadata to drive the CARS and introduces adaptability to changes in context and time.

This paper proposes the use of a Knowledge Graph (KG) to capture contextual information about the monitored system and the end-user. This data representation format captures semantic knowledge of different concepts and describes relations between these concepts, or entities, as (*subject*, *predicate*, *object*) triples. The information contained in a KG can be interpreted by both humans and machines. Leveraging knowledge graphs for recommender systems enables explainability which is otherwise difficult to obtain when dealing with black box ML models. Providing an explanation for the output of the Recommender System (RS) increases the trust in the system which is important in time-critical decision making. In addition to explainability, the inclusion of contextual metadata in the KG introduces relations between entities that would otherwise not be available, e.g., (*PatientX*, *has*, *Diabetes*) and (*PatientY*, *has*, *Diabetes*). These additional links in the dataset tackle the problem of sparsity in the dataset, the need for large datasets and the resulting scalability issues which RS are prone to [4].

This paper presents a Knowledge Graph-based Context-Aware Recommender System for Visualizations (VisCARS) which takes into account the contextual metadata about the monitored system and the end-user preferences to automatically recommend the most relevant data visualizations. The output of this research can be subdivided in four contributions:

- *C1:* The proposed KG-based, context-aware recommender system for visualizations and dashboards.
- *C2:* A dashboard ontology used to construct the KG and semantically describe the key concepts in dashboard applications and monitoring use-cases.
- *C3:* The accompanying open-source VisCARS framework enabling extension of the proposed solution and comparative evaluation with new context-aware models.
- *C4:* A context-aware dataset for visualization and dashboard recommendations collected during the conducted research and user study.

The remainder of this paper is structured as follows. First, Section II discusses related research in the field of VisRS as well as KG-based approaches to recommendations. Section III provides background information about the underlying techniques on which the presented VisCARS is based. Section IV presents the methodology used to achieve the outcome of this paper. Section V describes the resulting framework and design choices made during implementation. Section VI contains a complete evaluation of VisCARS with regard to performance and scalability, as well as an end-user study. Section VII reflects on the proposed solution and discusses its limitations and future work. Finally, Section VIII summarizes and concludes the conducted research.

II. Related Work

As, to the best of our knowledge, this paper is the first to present a context-aware solution to data visualization, the literature discussed in this section is divided in three categories: (i) recommender systems focusing on data-driven recommendations for data visualizations, (ii) personalized recommendation techniques applied to data visualization and (iii) generalized context-aware recommendation techniques.

A. Visualization Recommender Systems

1) Data-Driven Recommendation Techniques: Research in the field of data visualization recommendation has mostly been focused on data-driven techniques in the domain of data analytics and exploration [5]. Related work such as SeeDB [6] and ZenVisage [7] apply statistical analysis to provide insights in a given dataset. They start from an anchor visualization created by the end-user to generate data visualizations displaying similar trends or patterns within the data. ML-based approaches have been introduced which train models that learn appropriate visualizations directly from data without the need for advanced statistical analysis. For example, DeepEye [8] trained a ranking deep neural network model to classify visualizations as "good" or "bad" and rank them. The model is trained on 33 k bivariate visualizations of columns drawn from 42 public datasets, annotated by 100 students. VizML [9] collected and annotated a corpus of 2.3 million dataset-visualization pairs from Plotly [10], and extracted design choices from each visualization.

Compared to traditional ML-based recommendation systems, KG-based visualization recommendation systems bring a variety of entities and relations connecting data attributes and visualization design choices, and are capable of illustrating the reasoning process. While KG-based approaches can be grouped in three categories: (i) embedding-based techniques, (ii) connection-based techniques and (iii) propagation-based techniques, VisRS have been limited to embedding-based approaches [11]. KG4Vis [12] proposes a KG-based visualization recommendation system by capturing relations between data attributes (discretized continuous features and categorical features) and visualization design choices in the dataset. They then leverage a Knowledge Graph Embedding (KGE) technique to infer general rules between a data column and a visualization design choice (e.g. "The data type in the column is decimal" is associated with a line chart). The proposed recommendation system is trained on the same Plotly dataset used by VizML [9].

In recent years, multiple-view visualization (MV), or dashboard, recommender systems for analytical applications have gained attention in academia. MEDLEY [13] focuses on the

composition of dashboards consisting of multiple views by introducing an intent-based recommender system that derives a collection of visualizations based on the analytical task the end-user is trying to complete. The user has to select their intent from a list of supported choices (i.e. measure analysis, change analysis, category analysis, or distribution analysis) and the explicit data attributes they are interested in (e.g. profit) in order to receive a recommended dashboard. MultiVision [14] introduces a mixed-initiative system for designing multiple-view visualizations to analyze tabular datasets. They developed two deep learning models: (i) first, to assess the correctness of a single data visualization based on a large corpus of 200 k Excel data tables and charts [15] and (ii) second, to assess the correctness of a complete dashboard consisting of multiple data visualizations. Due to the lack of large datasets for MV, they introduced an interactive approach where the user can select visualization design choices. The model is then trained on these provenance data, i.e., log data from user interactions with the system.

To address the challenge of collecting large-scale and high-quality datasets required by the previously mentioned ML-based approaches, Dashbot [16] introduces a deep reinforcement learning (RL) approach to generate analytical dashboards. They use knowledge about data visualization (e.g. rules of diversity and parsimony) to design a training environment and rewards for the RL agents to explore and imitate human exploration behavior with a well-designed agent network. They compared their approach against MultiVision [14] through an elaborate user study as both introduced a user interaction-based approach.

DMiner [17] is a recommendation system for design rule mining. The authors focus on two prominent aspects of multiple-view dashboard design: (i) arrangement, which describes the position, size, and layout of each view in the display space, and (ii) coordination, which indicates the interaction between pairwise views. They constructed a new dataset containing 854 dashboards crawled online, and developed feature engineering methods for describing the single views and view-wise relationships in terms of data, encoding, layout, and interactions.

2) Personalized Recommendation Techniques: The previously discussed data-driven approaches provide visualization recommendations based purely on data attributes. They do, however, not take into account end-user preferences to obtain a personalized recommendation. VizRec [18] introduces the added dimension of personalization for data visualization and highlights the importance of capturing the preferences of the end-user to drive the recommendation process. VizRec applies classical recommendation techniques (i.e. content-based filtering and collaborative filtering) in a hybrid approach to predict recommendations for a given user and dataset. They create a user-item rating matrix to capture the preferences of users to a specific visualization. This approach, however, requires a lot of training data and examples of visualizations created for that specific dataset.

PVisRec [19] tackles this shortcoming by deconstructing the created visualization into data attributes (e.g. data type of a specific column that is visualized) and the supported visualization design choices (e.g. a bar chart x-axis accepts a string or date, the y-axis accepts a number). They then capture the preferences of the user to these features instead of a complete visualization. By doing this, they are able to provide personalized recommendations across multiple heterogeneous datasets. They released their corpus consisting of 17.4 k users exploring 94 k datasets with 2.3 million attributes and 32 k user-generated visualizations. VisGNN [20] introduces a knowledge graph-based approach to capture relations between data attributes, users and visualization design choices to introduce personalization based on user preferences in the visualization recommendation. They trained a Graph Neural Network (GNN) model to predict the likelihood of links in the graph, and consequently the interest of a user i in data attribute j.

B. Context-Aware Recommendation Techniques

Context-aware recommender systems (CARS) improve on classic recommendation techniques by taking into account contextual features that impact the decision process of a user. While previous work in the domain of visualization recommender systems focuses on recommendations for a specific dataset, and even a specific user, they do not yet consider the context to which the recommender system is applied. As mentioned in Section I, capturing and including such contextual features in the recommendation process is paramount for real-time monitoring applications as the state of the monitored system influences which data should be visualized. As to the best of our knowledge, we are the first to introduce a context-aware recommender system for data visualization, this section of the related work is limited to general context-aware recommendation techniques outside of the data visualization domain.

In the early stages of CARS, research has been conducted on how to extend the classical recommender system techniques. For example, Context-Aware Matrix Factorization (CAMF) [21] is an extension of the classical Matrix Factorization technique that takes into account contextual features. Introducing these additional features, however, increases the dimensionality of the dataset which requires an increasingly large dataset in order to accurately predict the users' ratings [22]. Musto et al. [23] presented a graph-based approach to context-aware recommendations using a Personalized PageRank (PPR) algorithm. By adding different weights to different nodes in the graph, a personalization is introduced to the random walk algorithm. This approach shows robustness towards sparse and small datasets and flexibility of the overall recommender system. However, the graphs explored in their work are limited in size and complexity as they consist only of three different types of nodes (i.e. users, contexts and items). As discussed in Section II-A2, the visualization (i.e. the rated item) should be deconstructed in its components and the preference of the user towards each of these components should be captured to accurately capture user preferences and recommend data visualizations without the need for large datasets.

C. Positioning

The proposed methodology, VisCARS, is situated within the field of multiple-view visualization recommender systems and

recommends multiple complementary visualizations in order to present a complete dashboard to the end-user. The aforementioned (multiple-view) visualization recommender systems focus mainly on recommending design choices and visual encodings for a given chart (e.g. KG4Vis [12], MultiVision [14]), or on dashboard composition and lay-out (e.g. DMiner [17]). These approaches rely on the user to manually select the data columns they wish to visualize in order to recommend a visualization. In cases where they do suggest data columns to be visualized (e.g. MultiVision [14], DashBot [16]), the focus lies on strategic decision-making and the outputs consist primarily of visualizations that highlight statistical aggregations, such as frequency of the values in a specific column or correlations between two data columns. The presented recommender system, however, focuses on recommending the content to visualize at a given moment in time without relying on the expertise of the end-user to derive which data should be visualized. The recommended data is based primarily on the state of the monitored system. In contrast to the strategic decision-making dashboards in related work, this paper focuses on operational decision-making and visualization of real-time monitoring data streams [24].

Based on this recommended content, VisCARS then suggests visualizations for the data of interest by taking into account the end-user preferences and thus provides a personalized recommendation similar to PVisRec [19] and VisGNN [20]. In contrast to the discussed related work, the proposed methodology introduces contextual features to find similarities and differences between contexts and users to drive the recommendation process and overcome the problem of data sparsity. This approach reduces the need for large datasets which are challenging to obtain, especially in specific monitoring use-cases where a limited number of visualizations are created on which a model can be trained.

III. Background

A. Collaborative Filtering

Collaborative Filtering (CF) is a well-established technique within the field of recommender systems due to its introduction of personalization. Even though it first came to the surface over a decade ago, it still shows state-of-the-art accuracy in recent studies [25]. There are two major approaches to CF: (i) neighborhood-based method and (ii) latent factor-based methods. Both techniques, including their advantages and disadvantages, are discussed in this section to provide the background information upon which the remainder of this paper builds.

1) Neighborhood-Based Methods: Neighborhood-based methods are based on the similarity between users, respectively items [26]. This similarity can be measured using different metrics, e.g. the Pearson correlation as shown in (1). It calculates similarity between target user *u* and user *v* based on the mutually rated items. $r_{u,i}$ is the rating for item i given by user u, and $\bar{r}_u$ is the average of all the ratings given by the same user u.

$$sim(u,v) = \frac{\sum_{i=1}^{m}(r_{u,i}-\bar{r}_u)(r_{v,i}-\bar{r}_v)}{\sqrt{\sum_{i=1}^{m}(r_{u,i}-\bar{r}_u)^2}\sqrt{\sum_{i=1}^{m}(r_{v,i}-\bar{r}_v)^2}} \quad (1)$$

The rating prediction for item *i* can then be calculated using the K-Nearest Neighbors (KNN) of target user *u* as shown in (2).

$$A_{u,i} = \bar{r}_u + \frac{\sum_{v=1}^{k}(r_{v,i}-\bar{r}_v)sim(u,v)}{\sum_{u=1}^{k} sim(u,v)} \quad (2)$$

This user-user approach can also be used to calculate the similarity between items, and consequentially provide a rating prediction for target item *i* based on the items previously rated by user *u*. User-based Collaborative Filtering (UBCF) and Item-based Collaborative Filtering (IBCF) are commonly used as a hybrid RS by assigning weights to the output of both approaches.

Neighborhood-based methods are intuitive and can be easily interpreted by looking at the similarities or differences between the target user and the users in its neighborhood. However, these methods are usually slower to compute, especially as the dataset grows larger.

2) Latent Factor-Based Methods: Simon Funk introduced Matrix Factorization (MF) during the Netflix prize competition [27]. He realized that the users' ratings aren't random. Users weigh the features they are interested in (e.g. favorite actor or genre) against features they dislike (e.g. long duration, bad jokes) to derive their final rating. This relation between users and items can be noted as a linear function, as shown in (3) and (4). $\mathbf{U}_u$ is the vector containing the weights describing the preference of user *u* towards the set of features and $\mathbf{I}_i$ the vector containing the latent factors, or features, for a specific item *i*.

$$\mathbf{A} \approx \mathbf{U}\mathbf{I}^T \quad (3)$$

With $\mathbf{A} \in \mathbb{R}^{m\times n}$ as the user-item rating matrix, and consequently $\mathbf{U} \in \mathbb{R}^{m\times d}$ and $\mathbf{I} \in \mathbb{R}^{n\times d}$ as the matrices containing the latent factors of the users and items respectively.

$$A_{u,i} \approx \mathbf{U}_u\mathbf{I}_i^T \quad (4)$$

Feature vectors, or embeddings, $\mathbf{U}_u$ and $\mathbf{I}_i$ can be approximated using Stochastic Gradient Descent (SGD) and a loss function. Over the years, many loss functions have been proposed in research, but the most basic one is to take into account only the observed ratings by the user. The simplest loss function that can be used is the Mean Squared Error (MSE) over all the observed ratings, as shown in (5).

$$\frac{1}{\|obs\|}\sum_{(i,j)\in obs}(A_{i,j}-\mathbf{U}_u\mathbf{I}_i^T)^2 \quad (5)$$

Latent factor-based methods, e.g., MF, Neural CF [28], have been proven to outperform their neighborhood-based counterparts due to the additional latent features being taken into account. However, the introduction of these embedded features reduces the explainability of the recommendations as they can not be interpreted and traced back to explicit features.

Both the neighborhood and the latent factor-based approaches are susceptible to sparsity in the dataset and especially the cold start problem. When a user has not consumed or rated any or only a handful of items, both the rating-based similarity and the embedding calculation will be inaccurate. This is a problem that has been the topic of recommender system research for many years [29], [30].

The methodology proposed in this paper is inspired by the discussed methods and aims to address the challenges they face using a Context-Aware Collaborative Filtering (CACF) approach. The performance achieved by the introduction of latent factors, or features, has been a motivation to research the impact of contextual features. While latent factors are calculated, or learned, by the machine learning model, contextual features can be introduced through domain expert knowledge. Moving towards a context-aware extension of these methods can thus increase the explainability of the RS while achieving good performance results without requiring a large training dataset. Furthermore, VisCARS leverages contextual features in addition to user-item ratings to calculate the similarities between contexts, users and items which is then used to drive the neighborhood-based CF process. As these features are considered domain expert knowledge, they are available for context, users and items which have not actively (been) consumed or rated. This provides an advantage over classic approaches and resolves issues with data sparsity and the cold start problem.

B. Knowledge Graph Embedding

As mentioned in Section II, recent works such as KG4Vis [12] and VisGNN [20] have introduced KGs to capture relations between features in the dataset. KG4Vis extracted data attributes and visualization design choices from the Plotly dataset. VisGNN introduced an additional dimension of personalization by linking individual users to these data attributes and visualization design choices to capture the user preferences. The proposed methodology, VisCARS, applies a similar strategy and introduces additional knowledge about the monitored system, data, users and visualizations as contextual features.

To extract these features from the KG and use them in the recommendation process, Knowledge Graph Embedding (KGE) techniques are applied. There are two categories of embedding-based techniques: (i) node embedding and (ii) graph embedding. The node embedding calculates an embedding for each entity and relationship in the graph, while the graph embedding calculates the embedding of an entire graph. Within each category, different approaches have been researched over the last years. The explored techniques are briefly discussed to provide the necessary background information.

1) Translational Distance Models: Translational Distance Models [31] (e.g. TransE, TransH, TransR, TransD) adopt distance-based scoring functions as a measure of the plausibility the triple (h, r, t) will occur in the graph. TransE [32], the most representative of the aforementioned techniques, projects both the entities and the relations in the graph into a low-dimensional vector space and maximizes its scoring function towards $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ for all triples in the graph. TransH [33] is an extension of TransE which tackles its most prominent shortcoming, i.e., dealing with multi-relational data. It introduces a hyperplane for each relation r in the graph. Each head entity h and tail entity t are then projected on this hyperplane. Therefore, each entity in the graph has a distinct vector representation when involved in different relations.

2) Semantic Matching Models: In contrast to distance-based scoring functions, Semantic Matching Models (e.g. RDF2Vec [34]) use language modeling approaches for unsupervised feature extraction from sequences of words (e.g. Word2Vec [35]), and adapt them to graph structures. Due to this representation as sentences, RDF2Vec is a node embedding technique able to capture complex sequences of relations in the graph, e.g., (*Ghent*, *isSituatedIn*, *Belgium*, *isSituatedIn*, *Europe*). While translational distance models are limited to the first-hop relations, semantic matching models are capable of capturing the relationship between *Ghent* and *Europe*. Graph2Vec [36] is a graph embedding technique that belongs to the category of semantic matching models. It considers graphs as a document or a sequence of sentences and leverages Doc2Vec [37], an extension of Word2Vec, to calculate an embedding for the entire graph.

IV. Methodology

In order to recommend a dashboard, an answer must be provided to two questions. First, *"What data does the user want to see?"*. This question focuses on recommending the most relevant data properties (e.g. different sensors) and depends on the current state of the monitored system. Second, *"How does the user want to see the data?"*. This derives the best way to visualize the data based on the end-user preferences. Both stages in the recommendation process are integrated in the dashboard recommendation pipeline which is visualized in Fig. 1. In this section, all components and underlying building blocks of the pipeline are presented and discussed.

The proposed solution aims to combine both the K-Nearest Neighbors-based CF (KNNCF) methods and the Latent Factor-based CF methods to tackle the disadvantages of both approaches. Introducing contextual knowledge of the system in the recommendation process provides information about the relationship between users and items similar to the latent factors learned during MF. As discussed in Section III-A2, these features positively impact the performance of the model. In comparison to the latent factors, the semantically annotated knowledge describes known concepts and can therefore be interpreted by humans. The combination of high performance and explainability increases the trust in the system and its adoption, which is critical when designing recommender systems that drive the decision making process. Additionally, the data sparsity and cold start problems which state-of-the-art recommender systems are prone to, are addressed. Capturing not only relations between users and items, but also relations between different users, or different items, can provide sufficient information to provide a recommendation for users who have not previously interacted with the dashboard platform, or newly created items that have not yet been visualized.

A. Ontology

The concept of context, and consequently the contextual metadata that is available varies based on the domain the RS is applied to. For example, in an IIoT use-case, the monitored

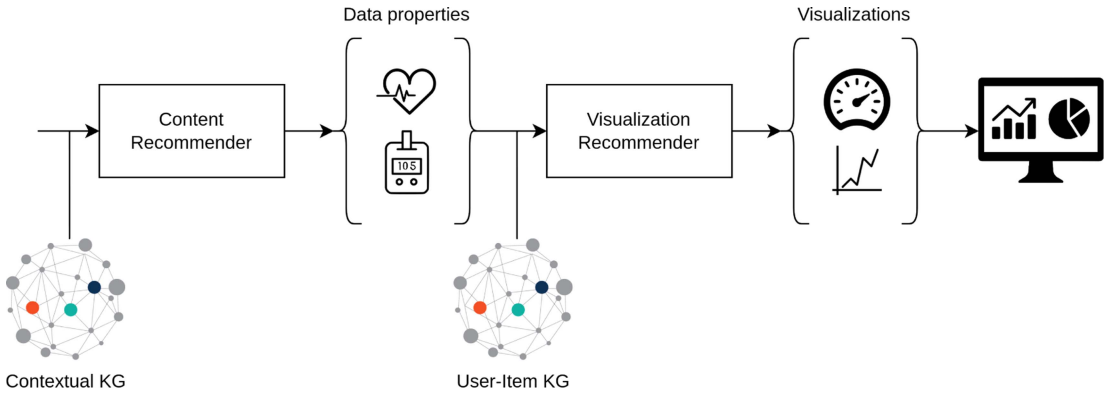

Fig. 1. Complete pipeline of the Knowledge Graph-based Context-Aware Recommender System including the input and output of the different components. The content recommender derives the most relevant data to visualize based on the context. The visualization recommender derives the best visualization type for the data based on the end-user of the dashboard.

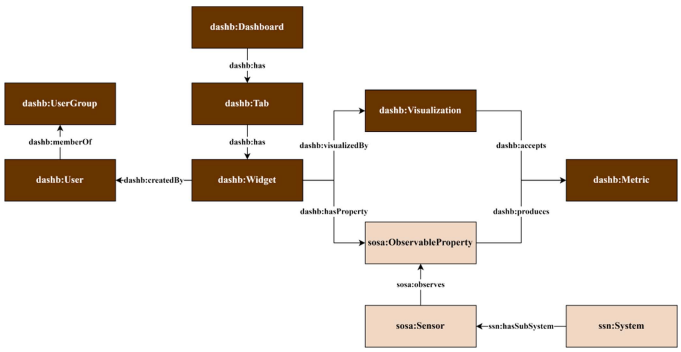

Fig. 2. Visualization of the most relevant concepts and relations described by the dashboard ontology. The lighter boxes indicate imported concepts used to describe the context (i.e. the monitored system and the available data).

system can be a physical machine. The contextual information about this machine includes a description of its components, the location of the machine in the industrial process, etc. In a smart healthcare use-case, the monitored system can be a patient. The contextual information contains the patient profile, medical records, etc. Both contextual KGs capture knowledge about distinct concepts. Considering both concepts as a system, (*Machine*, *a*, *ssn:System*) and (*Patient*, *a*, *ssn:System*), enables the RS to abstract and interpret the concept of *Machine* and *Patient* within the dashboard domain.

To achieve this, an ontology that describes the key concepts with regard to the monitored system and dashboard application is designed using the Web Ontology Language (OWL) [38], an extension of the Resource Description Framework (RDF) [39] for rich descriptions of concepts. The resulting Dynamic Dashboard ontology includes expert knowledge about the system or the data (e.g. the available sensors in the system and the properties they observe) as well as the dashboard components (e.g. different visualization types that are supported) and the user metadata (e.g. the expertise of the user). Its concepts with regard to dashboard applications, metrics, sensors and properties, and visualizations are discussed in this section. A schematic overview of this ontology is shown in Fig. 2 and the complete documentation of the designed ontology is available online.[1]

[1]https://docs.dynamicdashboard.ilabt.imec.be/

1) System, Sensors and Properties: As per Semantic Web best practices [40], the proposed ontology uses concepts of the SSN (Semantic Sensor Network) [41] and SOSA (Sensor, Observation, Sample, and Actuator) [42] ontologies in order to describe a system. A monitored *System* consists of a number of (virtual) sensors. Systems, subsystems and the relations between them can be described by hierarchically grouping a set of sensors. Each *Sensor* observes one or more sensor properties, or *ObservableProperty*. The semantic description of the monitored system is domain, or use-case specific and is created using domain expert knowledge. As an example, a GPS sensor in a car for an automotive IoT use-case can be semantically annotated as:

```
<car> a ssn:System ;
    ssn:hasSubSystem <gps> .

<gps> a sosa:Sensor ;
    sosa:observes <longitude> ;
    sosa:observes <latitude> .

<longitude> a sosa:ObservableProperty ;
    dashb:produces metrics:longitude .

<latitude> a sosa:ObservableProperty ;
    dashb:produces metrics:latitude .
```

Note that the <> notation is used to describe instances of concepts which are identified with a Uniform Resource Identifier (URI). A description should thus be provided for every monitored car, sensor and property.

2) Visualizations: The semantic description of the monitored system and its sensors provides insights into the data that can be visualized in the dashboard. Additionally, different supported visualizations can be semantically annotated. This captures important metadata about the visualization such as the type of data it can visualize. To continue the example, the semantic description of a map visualization is given:

As can be seen, the specified visualization consists of a single component that requires a latitude and longitude pair. The semantic description of the visualization provides information about which data types the visualization supports and expects.

```
<map> a dashb:Visualization, dashb:MarkerMap;
    rdfs:label "Marker Map"@en ;
    dashb:component
        [ dashb:accepts (
                [ a om:EclipticLatitude ]
                [ a om:EclipticLongitude ]
            ) ;
          dashb:min     1
        ] .
```

This information can be used by the recommender system to match data with supported visualization types.

3) Metrics: The concept of a *Metric* is used to describe a relation between visualizations, which accept a metric, and sensors properties, which produce that specific metric. The concept is inspired by, and extends the Ontology of Units of Measurement (OM) [43].

```
metrics:longitude a dashb:Metric ;
    a om:EclipticLongitude ;
    dashb:datatype xsd:double .

metrics:latitude a dashb:Metric ;
    a om:EclipticLatitude ;
    dashb:datatype xsd:double .
```

4) Dashboards, Tabs and Widgets: In addition to the conceptualization of the monitored system, the internal components of the dashboard application can be semantically described. Similar to systems and subsystems, a *Dashboard* consists of one or more dashboard tabs. Each *Tab* displays one or more widgets. A *Widget* is the core component of the dashboard. It describes relations to determine which sensor properties are visualized, which visualization type is used and which user created the specific widget. From the RS perspective, this component thus contains the *context*, *item* and *user* relations.

```
<dashboard> a dashb:Dashboard ;
    dashb:hasTab <tab> .

<tab> a dashb:Tab ;
    dashb:hasWidget <widget> .

<widget> a dashb:Widget ;
    dashb:createdBy <user> ;
    dashb:hasProperty <longitude> ;
    dashb:visualizedBy <map> .
```

B. Content Recommender

The first stage of the pipeline, i.e., content recommendation, provides an answer for the question *"What does the user want to see?"*. The answer is dependent on the current state of the monitored system. A KG can be constructed, based on the ontology described in Section IV-A, that provides an abstraction of the system. This enables the proposed solution to be applied across different domains. The backbone of both the context-driven and the user-driven recommendation stage is a Collaborative Filtering (CF) method. This method has been extended towards a Context-Aware Collaborative Filtering (CACF) approach to address the previously mentioned requirements such as performance and explainability. Throughout this section, the building blocks of the proposed methodology are discussed. A schematic overview of these building blocks and their dependencies is shown in Fig. 3.

1) Context Similarity: Since the proposed solution is an extension of neighborhood-based CF methods where the neighborhood is determined based on knowledge about the context, calculating the similarity between contexts is crucial. As mentioned in Section III-A1, the choice of the similarity metric influences the performance of the neighborhood-based approaches. As a result, various methods to measure the similarity have been researched, implemented and evaluated. These methods can be divided in two distinct categories: (a) embedding-based methods and (b) graph kernel-based methods. In the remainder of this section, both approaches will be discussed and compared.

(a) Embedding-based Similarity: As introduced in Section III-B, KGE techniques are commonly used to extract features, or embeddings, from graphs. These embeddings are often used directly as input features for machine learning models (e.g. neural networks). VisCARS, however, leverages the learned embeddings to calculate the similarity between different nodes in the KG (e.g. the different monitored systems). There are two distinct approaches to KGE: (i) node embedding techniques and (ii) graph embedding techniques. Node embedding techniques, such as TransE [32] and RDF2Vec [34], calculate an embedding vector for each node, or entity, and each vertex, or relation, in the KG. This means that the embedding of node *n*, calculated directly from the KG, contains all the information about that specific node. In comparison, graph embedding techniques, such as Graph2Vec [36], calculate a single embedding vector for the entire graph containing all the information captured within.

Once the embeddings have been obtained using one of these techniques, the cosine similarity between two embedding vectors can be calculated to provide a measure of the context similarity as shown in (6).

$$sim(A, B) = \frac{A.B}{\parallel A \parallel \parallel B \parallel} \tag{6}$$

(b) Graph Kernel-based Similarity: Instead of embedding the KG and calculating the distance between entities in the graph through their representation in the embedding space, graph kernel-based similarity metrics directly calculate the similarity between graphs using Graph Kernels [44]. A graph kernel is a symmetric, positive semidefinite function on a set of graphs $\mathcal{G}$. When a function $k : \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}$ is defined, it is known that a map $\phi : \mathcal{G} \rightarrow \mathcal{H}$ exists in the Hilbert space $\mathcal{H}$, such that:

$$k(G_i, G_j) = \langle \phi(G_i), \phi(G_j) \rangle_{\mathcal{H}} \tag{7}$$

for all $G_i, G_j \in \mathcal{G}$ where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is the inner product in $\mathcal{H}$. In essence, graph kernels provide a metric of similarity between graphs and are commonly used in combination with kernel-based ML models, such as the Support Vector Machine (SVM), to perform graph classification tasks. A state-of-the-art graph

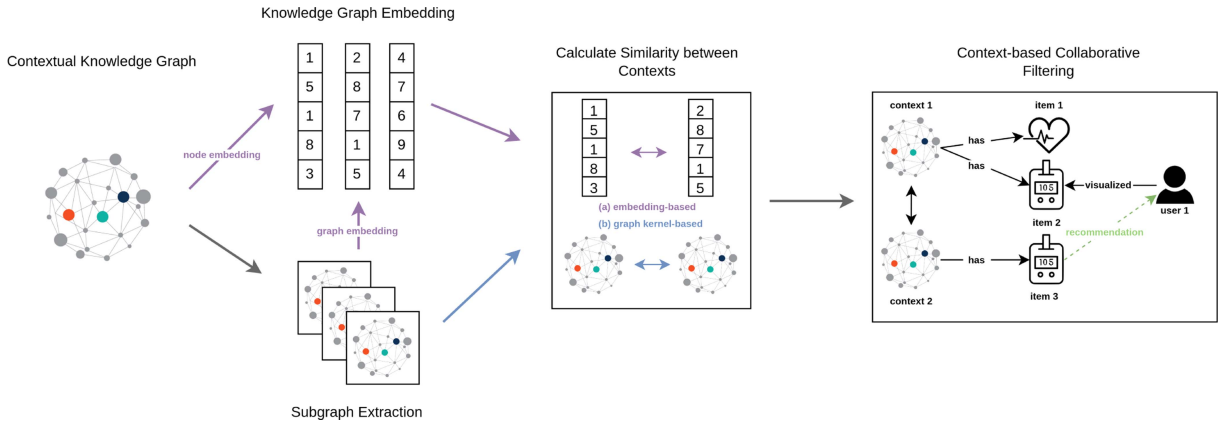

Fig. 3. Schematic overview of the proposed methodology. Starting from a knowledge graph-based representation of the contextual metadata, similarities between different contexts are calculated. This measure of similarity is used to determine the neighborhood of contexts and drives the context-aware collaborative filtering approach.

kernel is the Weisfeiler-Lehman graph kernel [45] which is based on the Weisfeiler-Lehman test for graph isomorphism.

Analogously to the distances between embeddings, the graph kernel values can be intuitively used to determine the KNN in neighborhood-based CF.

2) Ontology-Based Subgraph Extraction: While the node embedding-based techniques are executed directly on the entire dataset, the graph embedding-based and graph kernel-based methods require a (sub)graph that contains all the knowledge about the node of interest, e.g., a single context (i.e. monitored system), or a single user. A subgraph is defined as $G_i(\hat{V}, \hat{E}) \in G(V, E)$ where i is the node of interest and $(\hat{V}, \hat{E})$ the subset of vertices, or nodes, and edges in the KG. A common technique for subgraph extraction is to simply deploy a depth-first or breadth-first search starting from root node i.

When working with RDF graphs, however, attention should be given to the complexity introduced by different relationships (e.g. inverse, many-to-one) to ensure all the required information about the context, or user, is contained in the resulting graph, but no information about any other contexts, or users, is included to obtain an accurate comparison of contexts and users, and to avoid data leakage during evaluation. For example, using the dashboard ontology (see Fig. 2), a directed relationship exists between a specific widget w (w, *a*, *dashb:Widget*) and the target user u (u, *a*, *dashb:User*) we might be interested in. When traversing the directed graph, starting from root node u, no information about the widgets created by the user will be available due to the direction of the relationship between widgets and users. When approaching the graph as a bidirectional graph, too much information would be retrieved due to the many-to-one relationships in the graph (e.g. class declarations).

To overcome this complexity, a twofold approach is introduced. First, the ontology is used to extract all relevant relationships between entities. By traversing the ontology, rather than the KG directly, only paths between different concepts are taken into account, rather than paths between different instances of these concepts. Second, a recursive walker algorithm traverses the KG based on the paths extracted from the previous stage. Using the ontology yields an additional advantage by ensuring that the method can be dynamically applied to different use-cases when the relevant ontology is provided.

3) Context-Aware Collaborative Filtering: As mentioned earlier, the core component of the proposed RS is the Context-Aware Collaborative Filtering (CACF) technique. It is an extension of the well-established Neighborhood-based Collaborative Filtering approach where the neighborhood is not solely decided by the ratings in the dataset, but also by the contextual features that are available. These contextual features can provide information about the monitored system and the user for whom the dashboard is created. Depending on which features are taken into account, two approaches to CACF can be formulated.

As shown in (8), a prediction $C_{u,c,i}$ can be made for target user u and target item i for a specific context c (e.g. monitored system). In this approach, a similarity is calculated between all contexts in the dataset in order to obtain the top-K most similar contexts for context c. $r_{u,d,i}$ denotes the rating given by target user u for target item i in the similar context d.

$$C_{u,c,i} = \frac{\sum_{d=1}^{k} r_{u,d,i}.sim(c,d)}{\sum_{d=1}^{k} sim(c,d)} \tag{8}$$

Analogously, (9) approaches the CACF approach from a user perspective. Instead of calculating similarities between contexts, the similarity between the different users in the dataset are taken into account., resulting in a recommendation score $U_{u,c,i}$ based on the user similarity. $r_{v,c,i}$ here denotes the rating for target item i in the context of interest c given by similar user v.

$$U_{u,c,i} = \frac{\sum_{v=1}^{k} r_{v,c,i}.sim(u,v)}{\sum_{v=1}^{k} sim(u,v)} \tag{9}$$

The Context-Aware Collaborative Filtering can be used as a hybrid CF where a prediction $A_{u,c,i}$ is made based on the target context c and the target user u respectively. The resulting prediction is thus a trade-off between both approaches.

$$A_{u,c,i} = \alpha.C_{u,c,i} + (1-\alpha).U_{u,c,i} \tag{10}$$

The hyper-parameters of the proposed CACF method are the size of the neighborhood $k \leq |C|$ or $k \leq |U|$ and the trade-off ratio

$0 \leq \alpha \leq 1$ in case of a hybrid solution. A clustering method can be used to determine the size of the neighborhood. The trade-off ratio is dependent on the use-case and the expressiveness of the context features.

As mentioned earlier, the goal of the content recommender is to suggest the most relevant data properties to visualize on the dashboard. Consequently, the output of this recommender should be a batch of properties, or items. A decision should therefore be made, based on the prediction scores of all items, which items are recommended and where the cut-off is. This can be achieved by determining a threshold. The difficulty here is that the threshold differs for each algorithm and can drift over time as more users, items and contexts are added to the system. Another approach is to determine the size of the batch based on the average number of properties visualized for the target context c for all users in the neighborhood of target user u. VisCARS adopts the second approach due to its consistency across algorithms, which is important to maintain its modular design, and robustness with regard to the dataset.

C. Visualization Recommender

Once the most relevant data properties to be visualized on the dashboard are recommended by the previous stage, the visualization recommender needs to provide an answers to the question *"How does the user want to see the data?"*. This question is dependent on the user preferences and the specific data property that is being visualized. Unlike the content recommender, the output of the visualization recommender is a single item, as only one visualization type can be used to visualize a given property. Therefore, the additional logic required for batch recommendations is not required in this stage of the recommendation process.

Due to the modular design of each building block in the methodology, the visualization recommender pipeline consists of the same building blocks as the content recommender. The difference here is the concepts that are considered items and contexts. Whereas the content recommender provided recommendations for which data properties (*sosa:ObservableProperty*) to visualize on the dashboard based on the monitored system (*ssn:System*), the visualization recommender suggests the correct visualization (*dashb:Visualization*) based on the properties to visualize. The fact that both recommender pipelines are identical validates the applicability of VisCARS beyond the domain of data visualization, as long as the concepts within the domain are described by an ontology, and the users, items and contexts are defined.

D. Explainability

The acceptance and adoption of ML solutions beyond the context of research is often halted by the lack of trust in AI from the end-user. This trust is particularly difficult to obtain in critical domains such as healthcare [46] and (automated) decision making [47]. In addition to the performance and robustness of the models, the explainability of their output is a driving factor to obtain the trust of the end-user. The subject of eXplainable AI (XAI) for recommender systems has received more and more attention over recent years [48]. For KNNCF, an explanation is often derived from the selected neighborhood, e.g., *"You are similar to user X, and user X has rated item Y"*. Item-based CF, however, leverages item features to provide explanations such as *"You liked item X with feature Z. Item Y also has feature Z"*. CACF combines these approaches and utilizes not only the derived neighborhood, but also the knowledge in the KG to highlight the similarities and differences between the contexts in the neighborhood. While using graph kernel-based similarity techniques to drive the CACF, the similarities between contexts can be derived directly from the subgraphs. When utilizing emebedding-based techniques, however, the explanation is more difficult to obtain. KGE techniques are often black box models and the importance of each relationship in the KG, or contextual feature, during the calculation of the embeddings is unknown. To present the end-user with an explanation of the output of the RS, a reversed approach is implemented. Starting from the neighborhood, calculated from the node embeddings, the same subgraph extraction method can be used to extract a graph for each context in the specified neighborhood. Similarly to the previous method, a graph kernel-based model can then be deployed to derive the most important similarities and differences between these subgraphs. Using this approach, VisCARS provides an explanation to the end-user based on the contextual knowledge available about the context, or monitored system.

V. Implementation

A. VisCARS Framework

To conduct a comparative evaluation between different algorithms and models, a Python framework has been designed and implemented. The framework enables extension of the proposed solution and integration of alternative approaches. The accompanying code has therefore been made publicly available on GitHub.[2] Fig. 4 shows a high-level overview of the described architecture.

Fig. 4(a) visualizes the components essential to the functionality of the recommender system. An emphasis is placed on modular design and maximum reusability. Every implemented model, or *Recommender* implements a uniform interface. The recommender system proposed in this paper is based on KNNCF. As mentioned in Section IV, different algorithms, or *SimilarityMetrics*, can be used to determine the neighborhood, e.g. Knowledge Graph Embedding (KGE) based techniques. Many implementations already exist for KGE in Python. For the Translational Distance Models, the Python library for KnowlEdge EmbeddiNgs (PyKEEN) is used [49]. This library provides an implementation of 44 KGE models. In addition to these models, RDF2Vec has also been included in our framework as an example of a Semantic Matching Model. A Python implementation for this model has been provided by the pyRDF2Vec library [50]. For the graph kernel models, the GraKeL Python library is incorporated [51]. It unifies a collection of 16 graph kernels into a single framework, including the Weisfeiler-Lehman graph kernel used in the proposed solution.

A Data Access Object (DAO) pattern is used for all data queries on the graph, subgraph extraction and conversion of the data into a user-context-item matrix. Encapsulating all data

[2]https://github.com/predict-idlab/VisCARS

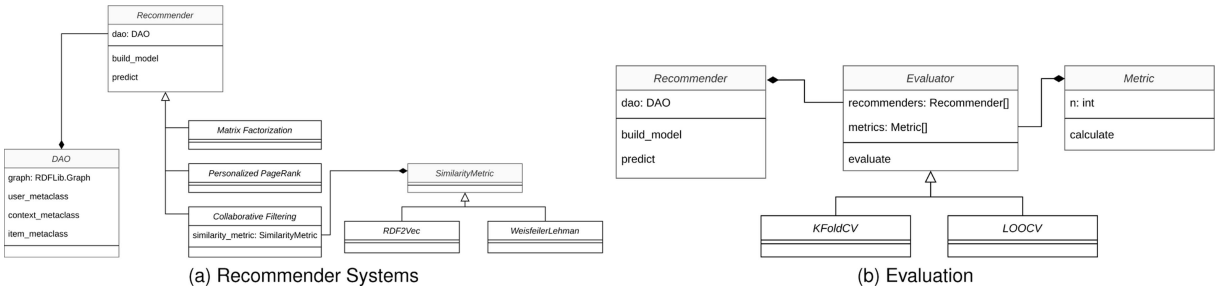

Fig. 4. High-level architecture of the VisCARS framework. The grey components describe parent classes or interfaces. The white components are subclasses or implementations of these interfaces. Note that not all implemented models are shown in the diagram.

access functionality in a single component allows generalisation towards different use-cases and a plug-and-play approach to different models and techniques. RDFLib, a Python library for RDF graphs [52], is used as graph representation. In addition to its support for various graph serialization formats, it supports SPARQL [53] queries on the graph.

Fig. 4(b) includes all the components with regard to evaluation of the models. Two evaluators have been implemented: K-Fold Cross Validation (KFoldCV) and Leave-One-Out Cross Validation (LOOCV). Analogously to the *SimilarityMetric* used in KNNCF, various common evaluation metrics for recommender systems are supported, e.g. hits, F1-score, Normalized Discounted Cumulative Gain (NDCG) [54]. All metrics can be calculated for top-N recommendations which is required for the batch of recommendations outputted by the content recommender.

B. Multiple-View Visualization Generation

The VisCARS framework calculates and outputs the top-N contextual recommendations for a specific user and context in the form of (*user*, *context*, *item*, *score*) quads in each stage of the recommender system (i.e. content recommendation and visualization recommendation). The final recommendations can be interpreted as dashboard widgets or visualizations using the dashboard ontology presented in Section IV-A:

```
<recommendation> a dashb:Widget ;
    # Context
    dashb:hasProperty <dht11.temp> ;
    # Item
    dashb:hasVisualization <thermometer> .
```

This simplified example describes a widget displaying temperature sensor data on a thermometer visualization. The VisCARS framework itself is agnostic to the visualization tool or dashboard application used to generate the visualizations and dashboards based on these recommendations.

To evaluate the presented recommender system, VisCARS has been integrated in the Dynamic Dashboard [1]. This semantic dashboard queries the semantic knowledgebase to obtain information about the property that should be visualized (e.g. *dht11.temp*) such as a human-readable label (e.g. "Temperature"), and a unit of measurement (e.g. "°C"). This metadata is then used to generate the visualizations, as shown in Fig. 5.

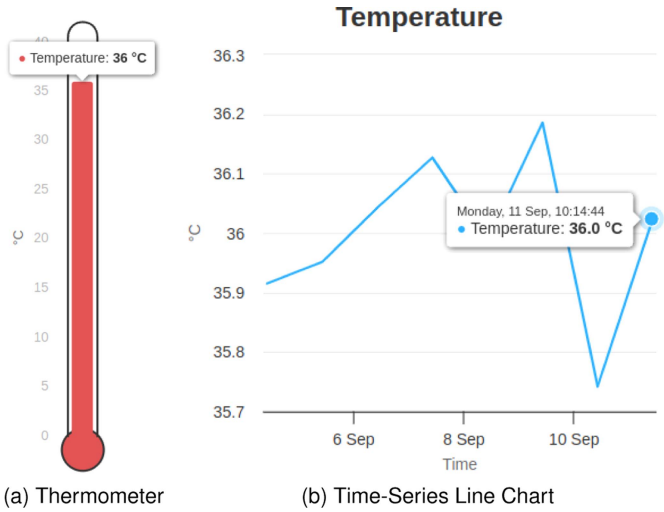

Fig. 5. Example visualizations generated for a sensor observing a temperature property. This sensor property can be visualized using different visualization types such as (a) a thermometer and (b) a more generic time-series line chart.

```
<dht11> a sosa:Sensor ;
    sosa:observes <dht11.humidity> ;
    sosa:observes <dht11.temp> .

<dht11.temp> a sosa:ObservableProperty ;
    rdfs:label "Temperature"@en ;
    dashb:produces metrics:celciusTemperature .

metrics:celciusTemperature a om:Temperature ;
    dashb:unit om:degreeCelsius ;
    dashb:datatype xsd:double .

om:degreesCelius rdfs:label "°C" .
```

As explained in Section IV-A, in addition to the sensor metadata, each supported visualization is enriched with a semantic description. The dashboard application provides an implementation for each of the supported visualization types. For the Dynamic Dashboard, the majority of these visualizations are designed using the Highcharts interactive charting library [55]. Additionally, visualizations not supported by Highcharts (e.g.

thermometer, generic JSON or HTML) and custom visualizations are implemented to enable the extension of the dashboard towards specific use-cases or projects.

```
<thermometer> a dashb:Visualization ;
    dashb:component
        [ dashb:accepts [ a om:Temperature ] ;
          dashb:min    1 ;
          dashb:max    1 ] .

<time-series-line-chart>
    a dashb:Visualization ;
    dashb:component
        [ dashb:accepts [
                    dashb:datatype xsd:double
                ] ;
          dashb:min    1 ;
          dashb:max    1 ] .
```

Each visualization has a one-to-one relationship with an implementation. This means that for each variation of a visualization, such as layout and design choices, a new subclass of the visualization should be added. For example, when visualizing multiple sensor properties, or data series, on a time-series line chart, a legend should be displayed. To achieve this, the underlying implementation (e.g. Highcharts *chartOptions*) should be adapted. This adaption results in a second implementation and a distinction between a *time-series-line-chart* and a *multi-time-series-line-chart*.

```
<multi-time-series-line-chart>
    a dashb:Visualization ;
    dashb:component
        [ dashb:accepts [
                    dashb:datatype xsd:double
                ] ;
          dashb:min    2 ;
          dashb:max    10 ] .
```

By doing this, the recommendation framework does not have to take into account individual visualization design choices and only outputs the correct visualization type (e.g. *time-series-line-chart*) to be used. Based on this approach, there are currently 21 different visualizations supported. Due to the modularity and dynamic character of the dashboard, additional visualizations can be easily added by providing an implementation (HTML/JavaScript) and the corresponding semantic description. The complete list of currently supported visualization types is given below. Note that not each parent class refers to an implemented visualization (e.g. Gauge, Map).

In contrast to specifications and grammars for visualizations, which include low-level specifications of each aspect of the chart (e.g. type of marker, encoding channels, data transformations) and even interactions with the graph (e.g. mouse events), the proposed ontology provides a high-level description of which type of data can be visualized using a specific visualization type (e.g. maps, gauges, line charts) and is used mainly to drive the RS. The dashboard application is thus tasked with providing an implementation for each supported visualization type. While

```
Visualization
├── LineChart
│   ├── TimeSeriesLineChart
│   └── MultiTimeSeriesLineChart
├── Gauge
│   ├── TraditionalGauge
│   └── ModernGauge
├── Map
│   ├── MarkerMap
│   └── RouteMap
├── ScatterPlot
├── PieChart
├── BarChart
│   └── StackedBarChart
├── ColumnChart
│   ├── StackedColumnChart
│   └── Thermometer
├── ColumnRange
├── KPI
├── DataTable
├── NetworkGraph
├── Audio
├── JSON
└── HTML
```

the Dynamic Dashboard utilizes Highcharts as underlying visualization tool, other popular charting libraries (e.g. Vega [56] and Plotly [10]) can be also used.

VI. Evaluation

A. Use-Case: Smart Healthcare

The proposed solution has been evaluated on a smart healthcare use-case. The goal of this project is to provide a safe environment for (elderly) patients in their own homes and decrease their stay in the hospital through remote monitoring solutions using wearables (e.g. smart watches) and sensors in the house. These patients are being monitored by various sensors and ML services including Anomaly Detection (AD) models. When something goes wrong, e.g. the patient presses an alarm button or one of the ML services detects an anomaly, a call is initiated with an operator. At that point in time, it is the task of the operator to derive the cause of the alarm and take the necessary action as quickly as possible. These actions include calling an ambulance, a nurse or a caregiver. The proposed dashboard recommender system can decrease the time required for the operator to make the right decision by providing them with an automatically generated dashboard based on the available contextual information about the patient and the situation they find themselves in. If according to the operator, a nurse should be called to check in on the patient, they too can receive a dashboard to assist them in providing the best care for the patient. Through the dashboard, a nurse can easily access the patient information including medical history and sensor data from the previous days or weeks. By providing the nurse with this information, they can optimize and personalize their care for the patient.

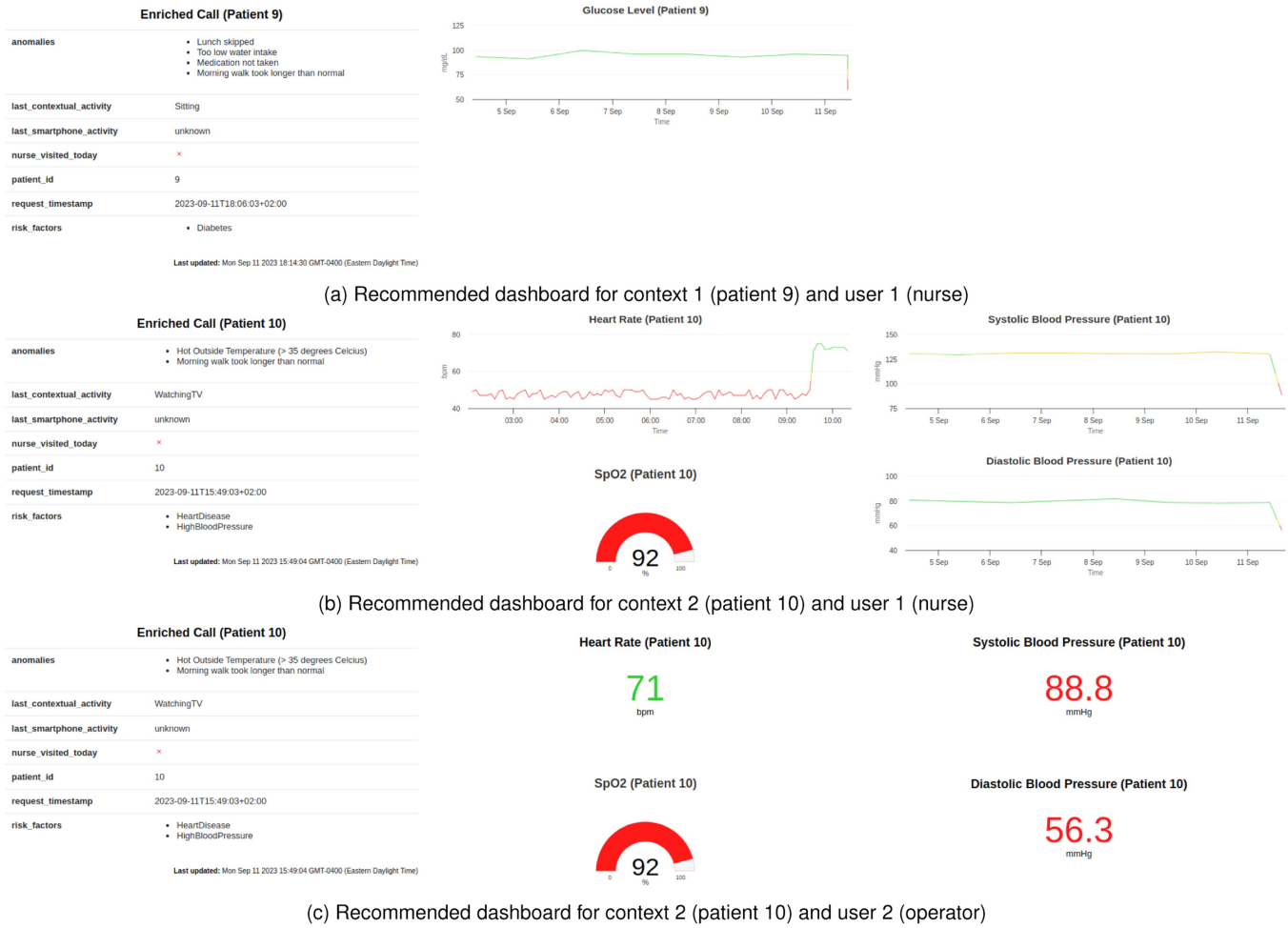

Fig. 6. Output of the presented context-aware recommender system integrated in the Dynamic Dashboard [1].

From the perspective of the recommender system, the patient can be abstracted as the *context* or the *monitored system* which has a number of sensors that can be observed. The operators and nurses in this use-case are considered as the *dashboard user* and the end-users of our system. Fig. 6 showcases different dashboards received from recommendations for two patients (patient 9 and 10) and two distinct users (a nurse and an operator). These examples highlight the importance and impact of contextual features about the context (i.e. monitored patient) and the end-user. When comparing a dashboard generated for a patient with diabetes (Fig. 6(a)) and a dashboard generated for the same end-user, but for a different patient with a known heart disease and high blood pressure (Fig. 6(b)), the visualized content shown completely changes. As discussed in Section IV-B3, the (number of) data properties that are visualized on each dashboard is based on patients most similar to the target. A comparison between dashboards generated for the same patient, but for distinct users, indicates the importance of end-user preferences and awareness to the task they are trying to complete. This is clearly shown by a dashboard for a nurse (Fig. 6(b)) who appears interested in seeing historical values displayed on a line chart while a call operator (Fig. 6(c)) is only interested in the current data values displayed as KPIs with clear color indicators to draw their attention. This is mostly due to the fact that a call operator is tasked with assessing the alarm and making a critical decision as quickly as possible. The end-user preference defined in this paper thus has a direct influence on the desired visualization type.

Dataset: Based on the available sensors and observed properties in the use-case, a KG was created taking into account the concepts described in the dashboard ontology discussed in Section IV-A. This contains all the information about the sensors, metrics, visualizations and dashboard users as well as the sample dashboards they have created. Information about the context is semantically annotated according to a use-case specific ontology, namely the Data Analytics in Health and Connected Care (DAHCC) ontology [57]. This ontology describes concepts about the monitored patients such as known diseases, impairments and prescribed medication. The data was collected through a user survey describing different scenarios with varying context. Domain experts (i.e. call operators and nurses) indicated what information they want to see depending on the medical background of the patient. For each sensor property they wish to visualize (e.g. diastolic blood pressure), the preferred visualization type (e.g. line chart), alerting thresholds (e.g. healthy between 110 and 130 mmHg) and data time range (e.g. last week) were indicated. The collected data has been made publicly

available.[3] The dataset contains dashboards created by 15 users (i.e. 10 call operators and 5 nurses) totalling 115 dashboards for 282 different patients, or contexts. These 282 contexts are monitored using 8 sensors each, resulting in a total of 1974 unique sensor properties that can be visualized. As mentioned in Section V-B, the dynamic dashboard supports 21 visualization types.

B. Baseline: Personalized PageRank

In addition to the classic approaches, i.e., Collaborative Filtering and Matrix Factorization, a state-of-the-art context-aware RS is used as baseline, namely Personalized PageRank (PPR) [23]. PPR is a context-aware extension of the PageRank (PR) algorithm first introduced by Google [58]. PR deploys a random agent on the graph which hops from one *node* to the next following the outgoing *edges*. The number of outgoing edges at node $\imath$ is given by $outdegree(\imath)$. The probability of traveling from node $\imath$ to node $\jmath$ can thus be defined by:

$$M_{\imath\jmath} = \begin{cases} \frac{1}{outdegree(\imath)} & \text{if} \quad \imath \to \jmath \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

At every step during the random walk, the agent can *teleport* to a random node instead of following the outgoing edges. The probability of following an outgoing edge is given by the *damping factor* $0 \leq \alpha \leq 1$, and the probability of teleportation to a random node is $\frac{1-\alpha}{N}$ with N being the number of nodes in the graph. The new probability matrix M' can then be defined as:

$$M' = \alpha M + (1-\alpha)\left[\frac{1}{N}\right]_{N\times N} \tag{12}$$

Given the column vector PR^t, where $PR^t_\imath$ is the probability that the random walk agent is at node $\imath$ at step t, the rank at step $t+1$ is defined by:

$$PR^{t+1} = \alpha M \times PR^t + (1-\alpha)\left[\frac{1}{N}\right]_{N\times 1} \tag{13}$$

The stationary probability distribution can be obtained by iterating (13) until convergence, i.e., $\|PR^{t+1} - PR^t\|$ is below a certain tolerance threshold or a maximum number of iterations is reached.

Due to the uniform distribution of probabilities in the PR algorithm, the output is an indication of the popularity of each node in the graph. When using this algorithm for a RS, the most popular items in the system would always be recommended, regardless of the user or context. If we no longer assume the probability that the random agent visits a specific node in the graph uniform, a bias can be introduced towards certain nodes in the graph. (13) can thus be rewritten as:

$$PR^{t+1} = \alpha M \times PR^t + (1-\alpha)p \tag{14}$$

where the personalization vector p is a N-dimensional vector containing a weight for each node in the graph, resulting in a Personalized PageRank (PPR) algorithm.

[3]https://github.com/predict-idlab/VisCARS/tree/main/data

The importance of two types of nodes in the graph is assumed: (i) the set of user nodes U, containing one user or a group of related dashboard users for which the recommendation is made and (ii) the set of context nodes C, containing one or more properties that should be visualized. The personalization for a specific node $\imath$ in the knowledge graph is defined by:

$$p_\imath = \begin{cases} w_u & \text{if} \quad \imath \in U \\ w_c & \text{if} \quad \imath \in C \\ 1 - w_u - w_c & \text{otherwise} \end{cases} \tag{15}$$

where w_u and w_c are the weights assigned to the user and the context nodes respectively. Musto et al. [23] suggest a normalized personalization vector and divide the weights assigned to each node by the size of the node set it belongs to. However, by doing so, the ratio of the importance between the different nodes is influenced by the size of the KG.

C. Performance

The complete recommendation pipeline, as shown in Fig. 1, consists of both the content recommender and the visualization recommender. As both components focus on solving a specific problem, they are evaluated separately using both a 5-Fold Cross Validation (KFoldCV) and Leave-One-Out Cross Validation (LOOCV). KFoldCV focuses on the general performance of the deployed models while LOOCV provides an indication of how well the different models deal with the cold start problem, i.e., when new contexts and users are added.

The performance benchmark has been conducted by comparing VisCARS to various alternative models. First, PageRank is used as baseline. It is a measurement of popularity and equivalent to recommending the most popular items in the graph. Collaborative Filtering (CF) and Matrix Factorization (MF) are examples of non-contextual algorithms discussed in Section III-A. As highlighted in Section VI-B, Personalized PageRank (PPR) is a context-aware extension of the PageRank algorithm that applies weights to specific user and context nodes to drive the popularity measurement. The VisCARS models include the two researched approaches discussed in Section IV: (i) a knowledge graph embedding-based approach and (ii) a graph kernel-based approach. These approaches are both implemented and evaluated. As discussed in Section V, RDF2Vec, denoted as VisCARS (CACFRDF2Vec), and Weisfeiler-Lehman, denoted as VisCARS (CACFWL), are used for the KGE-based and the graph kernel-based approach respectively. Additionally, a context-aware extension of the CF, namely Context-Aware Collaborative Filtering (CACF), is explored. This model uses data engineering based on the presented ontology to aggregate the users, items and contexts to tackle the data sparsity in the dataset. Instead of considering *dashb:User* and *sosa:ObservableProperty* as users and items respectively, the overlapping *dashb:UserGroup* and *dashb:Metric* is used. This reduces the dataset to 2 users and 8 items (per context) and thus increasing the number of ratings for each user (group) and item. Even though it tackles the data sparsity in the dataset, the use of these aggregations introduces limitations to the recommender system with regard to generalization.

TABLE I
PERFORMANCE OF DIFFERENT MODELS AND TECHNIQUES FOR CONTENT RECOMMENDATIONS

Model	KFoldCV		LOOCV	
	F1	NDCG	F1	NDCG
PageRank	0.3837	0.3888	0.4693	0.4610
Collaborative Filtering	0.3610	0.3600	0.2799	0.2802
Matrix Factorization	0.2615	0.2659	0.4265	0.4356
Personalized PageRank	0.4785	0.5071	0.4551	0.4613
VisCARS (CACF)	0.6641	0.6948	0.6498	0.6309
VisCARS (CACFRDF2Vec)	0.8158	0.8596	0.7978	0.8500
VisCARS (CACFWL)	**0.9807**	**0.9862**	**0.9888**	**0.9921**

TABLE II
PERFORMANCE OF DIFFERENT MODELS AND TECHNIQUES FOR VISUALIZATION RECOMMENDATIONS

Model	KFoldCV F1@1	LOOCV F1@1
PageRank	0.3048	0.2281
Collaborative Filtering	0.1147	0.0697
Matrix Factorization	0.3310	0.2990
Personalized PageRank	0.5746	0.3755
VisCARS (CACF)	0.7635	0.7878
VisCARS (CACFRDF2Vec)	0.8871	0.7673
VisCARS (CACFWL)	**0.9461**	**0.8000**

Content Recommender: The output of the content recommender is a batch of sensor properties to visualize in the dashboard. As mentioned in Section IV-B, the size of this batch is based on the context and is determined by the model, e.g. based on a threshold. Consequentially, top-N recommendations are generated with a varying N. Table I shows the averaged performance results of the content recommender. The F1-score and Normalized Discounted Cumulative Gain (NDCG) are calculated for both the KFoldCV and the LOOCV evaluators. The proposed CACF model based on the Weisfeiler-Lehman graph kernel, VisCARS (CACFWL), achieves an F1-score of 98,07% on the KFoldCV and 98,88% using the LOOCV. It outperforms all alternative models, including the baseline. The difference in performance between the traditional CF and MF models and the context-aware counterparts highlights the importance of the contextual features in the dataset as well as the impact of the data sparsity on the former models.

Visualization Recommender: The performance results for the visualization recommender are shown in Table II. A property can only be visualized by a single visualization type. That means that the visualization recommendation is either correct or incorrect. Consequentially, the F1@1 score is calculated. The performance of the visualization recommender shows similar trends as the content recommender. Again, the VisCARS (CACFWL) outperforms the alternative models and has an F1-score of 94,61% for the KFoldCV and 80,00% for the LOOCV. The drop in performance of the classic CF model during the LOOCV is noteworthy. It struggles to determine the neighborhood for a newly added user when not taking into account any semantic metadata, e.g. *UserGroup*.

D. Scalability

As VisCARS is targeted towards time-critical dashboard solutions, the proposed solution must be fast and scalable. An analysis towards the ability of the system to scale has been conducted using increasing graph sizes. As the graph increases in size, the number of contexts, users and previously rated items increases. An evaluation is done with regard to both the build stage of the model (e.g. extracting embeddings, subgraphs and calculating the similarities between them) and the prediction time given a specific user and context. In each iteration of the scalability evaluation, 100 new users are artificially added to the dataset. According to the collected dataset, a dashboard consisted of 3.1 widgets on average. A dashboard containing three random widgets (i.e. randomly selected sensor property and visualization type from the dataset) is thus automatically generated for each user. In total, 2285 additional dashboards were added to the dataset discussed in Section VI-A which includes 115 dashboards created by real users. This results in a total of 2400 dashboards, containing 7213 visualized widgets from 2365 users. This augmented dataset for scalability appeared to be sufficiently large to capture the trends for each evaluated model. A comparison was made between the proposed Context-Aware Collaborative Filtering (CACF) approaches, the classical CF and MF models and Personalized PageRank (PPR). All tests are executed on a Linux (Ubuntu 20.04) server with 2x Hexacore Intel E5645 (2.4 GHz) CPU and 24 GB RAM.

Fig. 7 shows the evaluation results with regard to the prediction time (Fig. 7(a)) and the build/train time (Fig. 7(b)) of each model. The prediction stage of the evaluation is executed five times and the scores for each iteration are averaged. The standard deviation is visualized as error bars. As indicated by the results, the prediction time for the MF model is negligible. It calculates the prediction scores for all users and items during the training stage of the model. The complete opposite is the case for the PPR model, no training stage exists and recommendation scores are calculated by applying the weights for the target user and context at prediction time. The time shown for the building stage is due to the conversion of graph representations. PPR is considerably faster than all other models. The Fast PageRank implementation used [59], utilizes Compressed Sparse Row (CSR) representations for sparse matrices to optimize its performance towards scalability.

The evaluation results, as discussed in Section VI-C, highlight the performance of VisCARS. The accuracy that is achieved, however, comes at a cost in computation time and scalability. The proposed CACF approach is slower than the previous models with regard to prediction time. The integration of a variety of third-party libraries introduces a complexity with regard to data format uniformity. All of these libraries introduce their own implementation of a graph. To improve scalability of the framework while still maintaining its modular design and plug-and-play characteristics, and avoid computing time spent on converting the data between different graph representations, models from these external libraries can be re-implemented using an uniform graph representation, e.g., RDFLib [52]. Furthermore, an additional optimization can be achieved by moving all SPARQL queries to extract information about the ratings to the build stage. During this step, the data can be transformed into a user-item-context matrix which can then be used for fast matrix calculations during the prediction stage.

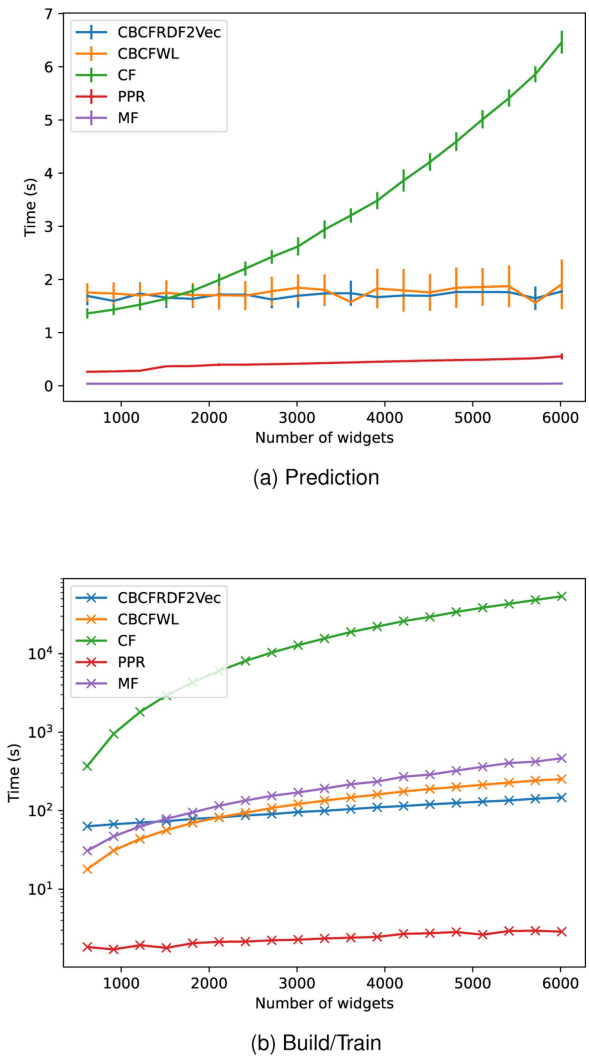

Fig. 7. Scalability evaluation of the best performing models. The timings are averaged over five iterations. Figure (a) shows the standard deviation as error bars.

The build stage of the CACF models are currently similar to the build time for the MF model, however, they scale better for larger datasets. During this stage, similarities between users or contexts are calculated. The embedding-based similarity requires recalculation of the embedding vectors and thus the similarity scores for all users or contexts once a new entity is added to the graph. This has a complexity of $O(n^2)$ where n equals the number of users, or contexts. The graph kernel-based similarity on the other hand generates a new subgraph for the newly added user or context. The previously generated subgraphs can be cached, therefore requiring a similarity calculation between this user and context and the already existing ones. The graph kernel-based methods are thus $O(n)$ complex.

E. User Study

After implementation of the proposed solution for the smart healthcare use-case described in Section VI-A, a user acceptance study has been conducted with participating call operators and nurses. These two distinct groups of users were asked a number of questions through a survey in order to determine (i) the acceptance rate of the dashboard recommender and (ii) the improvement on the traditional monitoring approach. Two scenarios were considered:

- S1 Given a specific context (i.e. patient), use the Dynamic Dashboard application to manually create visualizations in order to determine the root cause of the incoming alarm.
- S2 Given the output of the recommender system for a specific context, determine the root cause of the incoming alarm. If the output of the recommender system proves insufficient, the user can edit visualizations or create additional ones.

A total of ten participants (7 call operators and 3 nurses) were invited. All participants went through both scenarios for five random patients. For each cycle, the participant was asked to time themselves from the moment they started the exercise until they were able to determine the root cause of the alarm and consequently derive the correct action to be taken (e.g. call an ambulance, a nurse, a caregiver). Apart from the quantitative evaluation with regard to the execution time, a qualitative evaluation was also conducted through personal feedback on the recommender output and the derived explanation, incorporation of the recommender in their day-to-day workflow and shortcomings or possible improvements.

1) Quantitative Analysis: On average, the participants created 4 widgets per patient which took 4 minutes and 5 seconds ($\sigma = 54$ seconds) during the first scenario (S1). This time was reduced to 59 seconds ($\sigma = 17$ seconds) during the second scenario (S2) in which the recommender system was used. Only one participant added an additional visualization during this scenario. Once the user became more familiar with the dashboard application and gained more insights in the available sensor data within the use-case, the time used to create widgets dropped significantly (up to 34.17%). However, even for experienced users, the dashboard recommender proved reliable and beneficial in terms of time required to make a decision.

2) Expert Feedback: Both the quantitative and qualitative evaluation yield positive results. The proposed methodology received positive feedback and enthusiasm from the participating experts. The qualitative feedback is summarized based on an indication of their level of agreement, on a scale of one (strongly disagree) to five (strongly agree) on three questions with regard to the overall acceptance and usefulness, the transparency and explainability of the output, and the accuracy and reliability.

Acceptance and usefulness: The integration of the proposed methodology for dashboard recommendation was unanimously experienced as a significant improvement of their day-to-day workflow and the quality of service they could provide ($\mu = 4.8$, $\sigma = 0.42$). Particularly the context-aware content recommendation based on the medical background of the patient was considered to be the most important feature in reducing the time required to process the incoming call and derive the root cause of an alarm.

Transparency and explainability: The majority of the participants agreed that the output of the recommender system is easy to understand and transparent ($\mu = 4.0$, $\sigma = 0.67$). The most relevant information provided about the context for which

a dashboard has been recommended (e.g. "the patient has diabetes" as visualized in Fig. 6(a)) was considered to be sufficient in terms of explainability for why the recommender system suggested to visualize a certain sensor property (e.g. "Glucose Level"). Additionally, a short explanation was included in the survey with regard to the recommended visualization type based on their similarity between users in the dataset. It appeared obvious and correct that they were considered similar to users with the same expertise (call operator versus nurse).

Accuracy and reliability: VisCARS received overwhelmingly positive feedback with regard to its accuracy ($\mu = 4.7$, $\sigma = 0.48$). The participants compared the dashboard manually created during S1 with the recommended dashboard in S2. The recommendations were perceived as complete and reliable ($\mu = 4.5$, $\sigma = 0.71$). This result was also highlighted by the fact that only one out of ten participants felt the need to create an additional visualization to confidently make a decision about the root cause of the alarm.

3) Suggestions for Improvement: Apart from the positive feedback, some suggestions for future work were also obtained through the user survey. The most impactful suggestions are discussed.

Adaptive thresholds: One participant, a call operator with a medical background and experience as a nurse in the field, suggested an improvement on the visualized colors. In the current approach, the thresholds are considered expert knowledge and included in the ontology and thus as contextual features in the dataset. However, usage of fixed threshold values per sensor property (e.g. healthy heart rate values reside between 60 and 100 bpm) is considered to be a significant constraint of the proposed methodology. Thresholds, especially in this use-case, can vary based on the context (e.g. the patient) and even in time (e.g. sleeping versus exercising). This phenomenon can be seen in the dashboard generated for context 2 as shown in Fig. 6(b) where the heart rate is indicated as problematic while the patient appears to be sleeping. An improvement would be to use analysis of the historical data to estimate these thresholds per context and the current state thereof.

Dashboard composition: Two remarks were given on the dashboard composition. Design rules such as the location of widgets (e.g. always visualizing sensor properties in a specific order across dashboards, or displaying correlated sensor properties in a close proximity) could improve the time required for the operator to find the relevant information as quickly as possible. Multiple-view visualization composition is a relevant and actively researched field within the domain of visualization recommender systems [17], but has not been considered the core focus of this contribution. Extension of the proposed methodology with regard to dashboard composition and lay-out is considered future work.

VII. Discussion and Future Work

The proposed solution was integrated in the Dynamic Dashboard [1] and evaluated as a proof-of-concept. In this section, the proposed recommender system, VisCARS, is critically analyzed, and its limitations and open challenges for future work are discussed.

Importance of contextual awareness: Evaluation results of the proposed methodology highlight the benefits of including contextual features and domain knowledge to drive the automated generation of dashboards. The content recommendation requires awareness of the context in which the recommender system is applied in order to generate a complete dashboard consisting of multiple visualizations without intervention of the end-user.

Performance on limited datasets: Contextual features obtained through domain expert knowledge establishes links in the dataset between different contexts, items and users which would otherwise not be available (e.g. patients with similar medical backgrounds, users with similar professional expertise). These additional connections reduce the sparsity in the dataset which ensures high performance with reduced data samples. This proves to be particularly important in use-cases where domain experts are limited, and obtaining large datasets is costly and challenging.

Comparative evaluation: VisCARS has been evaluated towards performance and scalability. The considered baseline models include personalized neighborhood-based and latent factor-based recommendation techniques such as CF and MF, and an additional context-aware, graph-based recommendation technique, Personalized PageRank [23]. While comparing the proposed methodology with these baselines highlights the importance of contextual awareness, it does not directly evaluate VisCARS against related work in the domain of visualization recommender systems. VisCARS introduces the application of visualization recommender systems for real-time monitoring and root cause analysis in critical domains where dashboards have a strong dependency to the monitored system, i.e., context. This application domain has not thoroughly been researched despite high demand by industry. Content recommendation solutions in literature, e.g. MultiVision [14], Dashbot [16], focus on analytical tasks in which the end-user aims to gain insights into primarily tabular datasets. Preliminary evaluation results when applying MultiVision to the dataset collected in this research, which consists of time-series sensor data, have shown that the gap in application domains causes an unfair comparison between both solutions. Dashboards recommended by MultiVision contain visualizations portraying general statistics of the dataset, e.g., number of data samples per patient or the distribution of the data over time. These dashboards are not comparable to the real-time monitoring dashboards requested by the end-users in the presented use-case. To stimulate comparative evaluation of context-aware recommendations for visualization, the modular VisCARS evaluation framework and dataset have been made publicly available.

Context-aware datasets: The absence of contextual features in popular benchmark datasets for VisRS [60] makes it infeasible to evaluate the performance of the proposed solution on large, mature datasets. Without contextual information to calculate the neighborhood, VisCARS would revert back to a classic CF approach where similarity between contexts, or users, are solely based on the user-item ratings and would therefore not accurately represent the proposed solution, resulting in an unfair

performance evaluation. While the proposed methodology has been evaluated on a representative smart healthcare use-case, the obtained dataset during the research project is limited to the scope of the use-case. To objectively evaluate VisCARS towards generalization across different use-cases and domains, the context-aware dataset should be extended upon and additional use-cases should be included, e.g., predictive maintenance applications in Industry 4.0. Even though this would introduce distinct concepts of monitored systems and end-users, it is possible to find contextual similarities across use-cases. As more research is conducted in the domain of context-aware visualization recommender systems, more VisRS datasets that include contextual features will become available.

Extensive user study: Performance evaluation of context-aware visualization recommendation requires domain experts in the target domain who have sufficient knowledge of the monitored system, and the available data. Acquiring these domain experts to create context-aware datasets and the evaluation thereof is challenging. During the research project, access to 25 domain experts was obtained. 15 of these experts were used during data collection and the remaining 10 experts for evaluation of the proposed methodology through a user study. As more application domains are added to the context-aware dataset, a more extensive user study can be conducted indicative of the generalization of the proposed solution.

VIII. Conclusion

This paper introduces VisCARS, a Knowledge Graph-based Context-Aware Recommender System for Visualizations focused on dashboards to monitor systems in time-critical applications. The proposed methodology is based on Context-Aware Collaborative Filtering, an extension of the classic neighborhood-based CF. It utilizes contextual metadata to accurately determine the neighborhood of the target context, or user, and improve the recommendation process. Two approaches to extract these contextual features from a knowledge graph are researched and evaluated: a knowledge graph embedding-based approach and a graph kernel-based approach. In order to construct the KG containing all contextual information about the monitored system, a dashboard ontology is designed and presented. It semantically describes key concepts concerning the system, the available data and the different components in a dashboard ecosystem. Through this ontology-based approach, the VisCARS can be easily applied across different domains.

The presented recommender system is implemented and integrated in a state-of-the-art dashboard application, namely the Dynamic Dashboard, and has been evaluated for a representative smart healthcare use-case. To include different approaches and baseline models during the evaluation, a framework is implemented. It is designed towards extensibility and modularity, enabling the introduction of new models, evaluation techniques and metrics. The resulting framework and its variety of implemented models was evaluated towards performance and scalability. In addition to the quantitative evaluation of VisCARS, a qualitative user study was conducted which yielded positive feedback from domain experts. The results highlight the potential of VisCARS for sparse datasets with contextual information. The context-aware dataset for dashboard recommendations captured during the data collection campaign and user study is made publicly available.

Acknowledgment

The authors would also like to personally thank Emile Deman and Stef Pletinck for their support during the integration of VisCARS into the Dynamic Dashboard.

References

[1] P. Moens et al., "Event-driven dashboarding and feedback for improved event detection in predictive maintenance applications," *Appl. Sci.*, vol. 11, no. 21, Nov. 2021, Art. no. 10371. [Online]. Available: http://dx.doi.org/10.3390/app112110371

[2] M. Chakraborty and A. P. Kundan, "Grafana," in *Monitoring Cloud-Native Applications*. Berlin, Germany: Springer, 2021, pp. 187–240.

[3] D. G. Murray, *Tableau Your Data!: Fast and Easy Visual Analysis With Tableau Software*. Hoboken, NJ, USA: Wiley, 2013.

[4] Y. Gao, Y.-F. Li, Y. Lin, H. Gao, and L. Khan, "Deep learning on knowledge graph for recommender system: A survey," 2020, *arXiv: 2004.00387*.

[5] A. Wu et al., "AI4VIS: Survey on artificial intelligence approaches for data visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 28, no. 12, pp. 5049–5070, Dec. 2021.

[6] M. Vartak, S. Rahman, S. Madden, A. Parameswaran, and N. Polyzotis, "SeeDB: Efficient data-driven visualization recommendations to support visual analytics," in *Proc. VLDB Endowment Int. Conf. Very Large Data Bases*, NIH Public Access, 2015, pp. 2182–2193.

[7] T. Siddiqui, A. Kim, J. Lee, K. Karahalios, and A. Parameswaran, "Effortless data exploration with zenvisage: An expressive and interactive visual analytics system," 2016, *arXiv:1604.03583*.

[8] Y. Luo, X. Qin, N. Tang, and G. Li, "DeepEye: Towards automatic data visualization," in *Proc. IEEE 34th Int. Conf. Data Eng.*, 2018, pp. 101–112.

[9] K. Hu, M. A. Bakker, S. Li, T. Kraska, and C. Hidalgo, "VizML: A machine learning approach to visualization recommendation," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, 2019, pp. 1–12.

[10] "Plotly: Open source graphing library," 2012. Accessed: Oct. 02, 2023. [Online]. Available: https://plotly.com/

[11] Q. Guo et al., "A survey on knowledge graph-based recommender systems," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3549–3568, Aug. 2022.

[12] H. Li, Y. Wang, S. Zhang, Y. Song, and H. Qu, "KG4Vis: A knowledge graph-based approach for visualization recommendation," *IEEE Trans. Vis. Comput. Graph.*, vol. 28, no. 1, pp. 195–205, Jan. 2022.

[13] A. Pandey, A. Srinivasan, and V. Setlur, "MEDLEY: Intent-based recommendations to support dashboard composition," *IEEE Trans. Vis. Comput. Graph.*, vol. 29, no. 1, pp. 1135–1145, Jan. 2023.

[14] A. Wu et al., "Multivision: Designing analytical dashboards with deep learning based recommendation," *IEEE Trans. Vis. Comput. Graph.*, vol. 28, no. 1, pp. 162–172, Jan. 2022.

[15] M. Zhou et al., "Table2charts: Recommending charts by learning shared table representations," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2021, pp. 2389–2399.

[16] D. Deng, A. Wu, H. Qu, and Y. Wu, "DashBot: Insight-driven dashboard generation based on deep reinforcement learning," *IEEE Trans. Vis. Comput. Graph.*, vol. 29, no. 1, pp. 690–700, Jan. 2023.

[17] Y. Lin, H. Li, A. Wu, Y. Wang, and H. Qu, "Dashboard design mining and recommendation," *IEEE Trans. Vis. Comput. Graph.*, to be published, doi: 10.1109/TVCG.2023.3251344.

[18] B. Mutlu, E. Veas, and C. Trattner, "VizRec: Recommending personalized visualizations," *ACM Trans. Interactive Intell. Syst.*, vol. 6, no. 4, pp. 1–39, 2016.

[19] X. Qian et al., "Personalized visualization recommendation," *ACM Trans. Web*, vol. 16, no. 3, pp. 1–47, 2022.

[20] F. Ojo et al., "VisGNN: Personalized visualization recommendationvia graph neural networks," in *Proc. ACM Web Conf.*, 2022, pp. 2810–2818.

[21] L. Baltrunas, B. Ludwig, and F. Ricci, "Matrix factorization techniques for context aware recommendation," in *Proc. 5th ACM Conf. Recommender Syst.*, 2011, pp. 301–304.

[22] S. Raza and C. Ding, "Progress in context-aware recommender systems—an overview," *Comput. Sci. Rev.*, vol. 31, pp. 84–97, 2019.

[23] C. Musto, P. Lops, M. de Gemmis, and G. Semeraro, "Context-aware graph-based recommendations exploiting personalized pagerank," *Knowl.-Based Syst.*, vol. 216, 2021, Art. no. 106806.
[24] A. Sarikaya, M. Correll, L. Bartram, M. Tory, and D. Fisher, "What do we talk about when we talk about dashboards?," *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 1, pp. 682–692, Jan. 2019.
[25] Y. Koren, S. Rendle, and R. Bell, "Advances in collaborative filtering," in *Recommender Systems Handbook*. Berlin, Germany: Springer, 2022, pp. 91–142.
[26] C. C. Aggarwal, "Neighborhood-based collaborative filtering," in *Recommender Systems*. Berlin, Germany: Springer, 2016, pp. 29–70.
[27] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Matrix factorization and neighbor based algorithms for the netflix prize problem," in *Proc. ACM Conf. Recommender Syst.*, 2008, pp. 267–274.
[28] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.
[29] X. N. Lam, T. Vu, T. D. Le, and A. D. Duong, "Addressing cold-start problem in recommendation systems," in *Proc. 2nd Int. Conf. Ubiquitous Inf. Manage. Commun.*, 2008, pp. 208–211.
[30] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, "Facing the cold start problem in recommender systems," *Expert Syst. Appl.*, vol. 41, no. 4, pp. 2065–2073, 2014.
[31] S. Zhang, Z. Sun, and W. Zhang, "Improve the translational distance models for knowledge graph embedding," *J. Intell. Inf. Syst.*, vol. 55, no. 3, pp. 445–467, 2020.
[32] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2787–2795.
[33] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proc. AAAI Conf. Artif. Intell.*, 2014, pp. 1112–1119.
[34] O. Lassila et al., "Resource description framework (RDF) model and syntax specification," 1998. [Online]. Available: http://www.w3.org/TR/REC-rdf-syntax/
[35] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.
[36] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning distributed representations of graphs," 2017, *arXiv: 1707.05005*.
[37] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. Int. Conf. Mach. Learn.*. PMLR, 2014, pp. 1188–1196.
[38] G. Antoniou and F. V. Harmelen, "Web ontology language: OWL," in *Handbook on Ontologies*. Berlin, Germany: Springer, 2004, pp. 67–92.
[39] S. Decker et al., "The Semantic Web: The roles of XML and RDF," *IEEE Internet Comput.*, vol. 4, no. 5, pp. 63–73, Sep./Oct. 2000.
[40] A. Gyrard, M. Serrano, and G. A. Atemezing, "Semantic Web methodologies, best practices and ontology engineering applied to Internet of Things," in *Proc. IEEE 2nd World Forum Internet Things*, 2015, pp. 412–417.
[41] H. Neuhaus and M. Compton, "The semantic sensor network ontology," in *Proc. AGILE Workshop Challenges Geospatial Data Harmonisation*, Hannover, Germany, 2009, pp. 1–33.
[42] K. Janowicz, A. Haller, S. J. Cox, D. Le Phuoc, and M. Lefrançois, "SOSA: A lightweight ontology for sensors, observations, samples, and actuators," *J. Web Semantics*, vol. 56, pp. 1–10, 2019.
[43] H. Rijgersberg, M. Van Assem, and J. Top, "Ontology of units of measure and related concepts," *Semantic Web*, vol. 4, no. 1, pp. 3–13, 2013.
[44] M. Sugiyama, M. E. Ghisu, F. Llinares-López, and K. Borgwardt, "graphkernels: R and python packages for graph comparison," *Bioinformatics*, vol. 34, no. 3, pp. 530–532, 2017. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btx602
[45] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels.," *J. Mach. Learn. Res.*, vol. 12, no. 9, pp. 2539–2561, 2011.
[46] A. Adadi and M. Berrada, "Explainable AI for healthcare: From black box to interpretable models," in *Proc. Embedded Syst. Artif. Intell.*, Springer, 2020, pp. 327–337.
[47] J. M. Rožanec, P. Zajec, K. Kenda, I. Novalija, B. Fortuna, and D. Mladenić, "XAI-KG: Knowledge graph to support XAI and decision-making in manufacturing," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.*, Springer, 2021, pp. 167–172.
[48] A. Vultureanu-Albişi and C. Bădică, "Recommender systems: An explainable AI perspective," in *Proc. Int. Conf. Innovations Intell. Syst. Appl.*, 2021, pp. 1–6.
[49] M. Ali et al., "PyKEEN 1.0: A python library for training and evaluating knowledge graph embeddings," *J. Mach. Learn. Res.*, vol. 22, no. 1, pp. 3723–3728, 2021.
[50] G. Vandewiele, B. Steenwinckel, T. Agozzino, and F. Ongenae, "pyRDF2Vec: A python implementation and extension of RDF2Vec," 2022, *arXiv:2205.02283*.
[51] G. Siglidis, G. Nikolentzos, S. Limnios, C. Giatsidis, K. Skianis, and M. Vazirgiannis, "GraKel: A graph kernel library in python," *J. Mach. Learn. Res.*, vol. 21, no. 54, pp. 1–5, 2020.
[52] D. Krech, "RDFLib: A python library for working with RDF," 2006. [Online]. Available: https://github.com/RDFLib/rdflib
[53] J. Pérez, M. Arenas, and C. Gutierrez, "Semantics and complexity of SPARQL," *ACM Trans. Database Syst.*, vol. 34, no. 3, pp. 1–45, 2009.
[54] H. Valizadegan, R. Jin, R. Zhang, and J. Mao, "Learning to rank by optimizing NDCG measure," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1883–1891.
[55] A. S. Highsoft, "Highcharts," 2015. Accessed: Oct. 2, 2023. [Online]. Available: http://www. highcharts.com/products/highcharts
[56] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer, "Vega: A visualization grammar," *IEEE Trans. Visualization Comput. Graph.*, vol. 23, no. 1, pp. 341–350, 2016.
[57] B. Steenwinckel et al., "Data analytics for health and connected care: Ontology, knowledge graph and applications," 2022. [Online]. Available: https://dahcc.idlab.ugent.be
[58] D. F. Gleich, "Pagerank beyond the web," *SIAM Rev.*, vol. 57, no. 3, pp. 321–363, 2015.
[59] P. A. Lofgren, S. Banerjee, A. Goel, and C. Seshadhri, "Fast-ppr: Scaling personalized pagerank estimation for large graphs," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 1436–1445.
[60] L. Podo, B. Prenkaj, and P. Velardi, "Machine learning for visualization recommendation systems: Open challenges and future directions," 2023, *arXiv:2302.00569*.