Journal of Intelligent Information Systems (2022) 58:43–66
https://doi.org/10.1007/s10844-021-00650-z

# Knowledge graph summarization impacts on movie recommendations

Juarez A. P. Sacenti[1] · Renato Fileto[1] · Roberto Willrich[1]

Received: 30 November 2020 / Revised: 28 May 2021 / Accepted: 31 May 2021 /
Published online: 22 June 2021

## Abstract

A classical problem that frequently compromises Recommender System (RS) accuracy is the sparsity of the data about the interactions of the users with the items to be recommended. The use of side information (e.g. movie domain information) from a Knowledge Graph (KG) has proven effective to circumvent this problem. However, KG growth in terms of size and complexity gives rise to many challenges, including the demand for high-cost algorithms to handle large amounts of partially irrelevant and noisy data. Meanwhile, though Graph Summarization (GS) has become popular to support tasks such as KG visualization and search, it is still relatively unexplored in the KG-based RS domain. In this work, we investigate the potential of GS as a preprocessing step to condense side information in a KG and consequently reduce computational costs of using this information. We propose a GS method that combines embedding based on latent semantics (ComplEx) with nodes clustering (K-Means) in single-view and multi-view approaches for KG summarization, i.e. which act on the whole KG at once or on a separated KG view at a time, respectively. Then, we evaluate the impacts of these alternative GS approaches on several state-of-the-art KG-based RSs, in experiments using the MovieLens 1M dataset and side information gathered from IMDb and DBpedia. Our experimental results show that KG summarization can speed up the recommendation process without significant changes in movie recommendation quality, which vary in accordance with the GS approach, the summarization ratio, and the recommendation method.

**Keywords** Recommender systems · Knowledge graphs · Graph summarization · Entity clustering · Knowledge graph embeddings · Multi-view clustering

✉ Juarez A. P. Sacenti
juarez.sacenti@posgrad.ufsc.br

Renato Fileto
r.fileto@ufsc.br

Roberto Willrich
roberto.willrich@ufsc.br

[1] Graduate Program in Computer Science (PPGCC), Federal University of Santa Catarina (UFSC), Florianópolis, Santa Catarina, Brazil

## 1 Introduction

The use of Recommender Systems (RSs) (Aggarwal et al. 2016) has become part of our daily online experience. They help users to find and choose things (e.g., consumption items (Mesas & Bellogín, 2020), other users (Rodríguez-García et al. 2019), places (Zheng et al. 2018), and tags (Yu et al. 2018a)) based on preference models. Over the years, RSs have addressed information overload on the Web and have contributed to user satisfaction and a revenue increase of many streaming and e-commerce businesses (e.g., Netflix, Amazon, Youtube, Pandora Radio, Last.fm).

A classical problem that causes low RS accuracy is the sparsity of data about interactions of users with items to recommend. To mitigate this problem, a usual solution is the exploitation of side information, i.e. data about items or users, which can be gathered from external sources (e.g., linked data collections) and used to enrich the RS inputs. For example, a movie RS may exploit DBpedia data about genres, directors, actors, keywords, and awards of the movies. Other works exploit social media data, like user profiles, social relations, tags, and posts (Shokeen & Rana, 2020).

Knowledge Graphs (KGs) have proven to be effective to represent side information and improve recommendation (Sun et al. 2018). Usually, a KG used in a RS is a labeled graph whose nodes denote entities of different types (e.g., *User*, *Movie*, *Genre*, *Director*, *Actor*) and edges denote relations between them (e.g, *type*, *rates*, *watches*, *hasGenre*, *hasDirector*, *hasActor*). A KG tailored for an RS can include facts (triples) selected from multiple sources, such as linked data sources (e.g., DBpedia, Freebase, YAGO, Google's KG), and facts extracted from structured datasets (e.g., IMDb, Rotten Tomatoes, Facebook Graph).

In a KG-based RS, traditional user-item interaction models are enriched by linking users and items to entities describing their relevant properties. Then, the recommendations are usually generated by mining meta-paths or by encoding and exploiting low-rank latent representations of the resulting KG. These algorithms are usually complex and their costs are directly related to the KG size. The KG can become quite large as the number of users, items, interactions, and the amount of side information increase. Therefore, computational methods for reducing KG size and eliminating irrelevant and noisy data become crucial for KG-based RS performance. This challenge is very relevant to real RSs because KG-based RS algorithms must be executed as often as possible to consider the latest user interactions in the recommendation task. Therefore, these algorithms must be fast running and robust to frequently and quickly produce updated preference models.

In this work, we investigate the use of Graph Summarization (GS) (Liu et al. 2018) as a way to mitigate some of the problems pointed out above. GS algorithms can transform graphs into more compact representations while preserving properties that are useful to some application or domain. Some recent works like (Sydow et al., 2013; Ali et al. 2020) apply GS to accelerate query processing in graphs, support graph visualization, and analytics. These works point out that GS has several benefits, including reduction of data volume, speedup of algorithms or queries on graphs, and noise reduction. While GS has become popular for these purposes, its application to summarize KGs representing side information in KG-based RS is relatively new and unexplored. To the best of our knowledge, few works (Ragone et al. 2017; Wu et al. 2015) have applied GS to certain tasks of a particular RS, and no studies have proposed GS approaches to reduce computational costs and/or improve recommendation quality in KG-based RS.

This work contributes to narrow this research gap by investigating the use of GS as a preprocessing step for KG-based RSs. In this work, given that GS algorithms are application-specific, we propose a GS method aiming KG size reduction to speed up the training time of the KG-based RS models without significant impact on effectiveness. We adopt the premise that semantic similarity between entities can be used to summarize KGs for RSs. Thus, we propose a latent-based GS method that first translates the entities of a given KG to embeddings (using ComplEx) and then groups similar entities in clusters (using K-Means). Finally, entities belonging to the same cluster are represented as one supernode in the summarized KG.

We propose and evaluate two latent-based GS approaches for KG-based RS: single-view and multi-view. The former applies GS to the whole KG at once and can merge entities of different types (e.g., actor and director) into a single supernode. The multi-view GS approach, on the other hand, first splits the KG into sub-graphs representing different aspects (facets). Then, it summarizes these sub-graphs independently and combines their summaries into an entire summarized KG. In the multi-view approach, only entities of the same sub-graph can be merged in the same supernode. The use of multi-view summarization was motivated by previous works showing that hierarchy customization (i.e., summarization of hierarchical sub-graphs) (Fernandes et al., 2017) and multiple aspects of side information (multi-view perspectives) (Sacenti et al., 2018) influence recommendation quality and tolerance to data sparsity.

Figure 1 illustrates the multi-view GS approach applied to KG-based recommendation of movies. In this approach, the KG content is organized in three distinct sub-graphs (views): *Genres*, *Actors*, and *Directors*. Each sub-graph is summarized independently and the obtained summaries are combined into a single KG in which only entities from the same sub-graph can be merged in the same supernode. For instance, in the genre view, *Action* and *Adventure* are merged into supernode *SN-1* representing these two genres, because of their semantic similarity in the embedding space. Thus, movies belonging to these genres become closer (e.g., *Terminator: Dark Fate*, *The expendables 3*, *Braveheart*). Analogously, in the actors' view *A. Schwarzenegger* and *S. Stallone* are merged into *SN-3*. Finally, in the directors' view, *P. Hughes* and *M. Gibson* are merged into *SN-4*.

Notice that the frequency of KG summarization is lower than that of the re-training of the KG-based RS model. The latest must be performed as often as possible to take into account new user interaction data. The insertion of new facts in the KG and even items to be recommended is usually much less frequent than user interactions. Therefore, KG summarization can be performed just once to produce a summarized KG that can be used

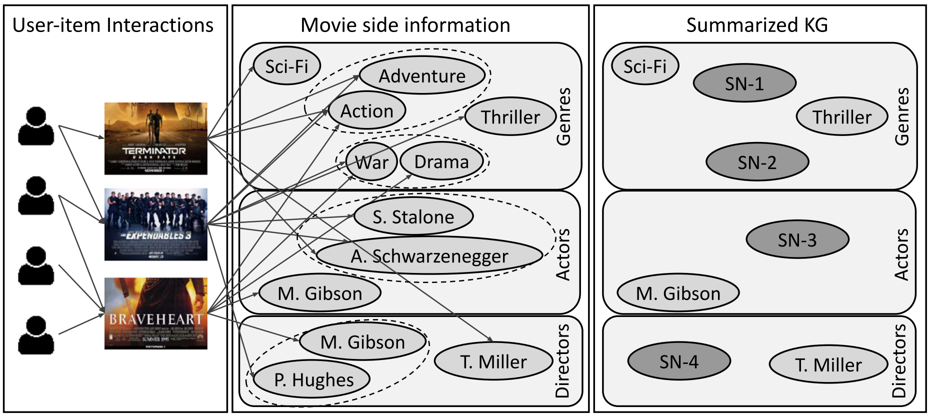

**Fig. 1** Example of multi-view GS for KG-based recommendation of movies

for re-training the KG-based RS model several times to capture new user interactions in the model. Thus, the proposed preprocessing step cannot be considered as an additional overload on the re-training of the KG-based RS model. On the contrary, KG preprocessing is expected to reduce the overall training time of KG-based RS models, while keeping the RS effectiveness or even slightly improving it in some cases.

We performed experiments using different combinations of our GS approaches and KG-based RSs to investigate the impact on effectiveness and efficiency. These experiments are conducted in the movie domain, considering datasets combining MovieLens 1M (Harper & Konstan, 2015) with side information gathered from IMDb[1] and DBpedia[2]. We assessed the summarization impacts in terms of KG reduction, recommendation effectiveness, and training efficiency. The results show that it is possible to reduce RS training costs with the increase of the summarization ratio, without significant losses in recommendation effectiveness. Sometimes, there were even non-significant gains in measures like precision, depending on the GS approach, summarization ratio, recommendation method, and parametrization.

Thus, the main contributions of this work can be stated as follows:

i. a GS method that combines embedding based on latent semantics (ComplEx) and KG node clustering (K-Means) in single-view and multi-view approaches;
ii. a recommendation process that exploits alternative GS strategies as a preprocessing step of KG-based RSs;
iii. the evaluation of the impact of our GS method when generating KGs with increasing summarization ratios on RS model training time and RS effectiveness.

The rest of this paper is structured as follows. Section 2 discusses KG-based RSs. Section 3 discusses related works. Section 4 describes our proposal. Section 5 describes our experiments and Section 6 reports their results. Finally, Section 7 presents conclusions and future work.

## 2 KG-based recommender systems

A KG-based RS suggests items to users based on KG-derived models of user preferences. These models take into account information represented by KG nodes and edges, including user-item interactions and side information about items and users. User-item interactions include user actions and feedback, such as explicit scores (e.g., 1-10 ranges, 5-stars, like-dislike), and implicit traceable actions (e.g., views, buying behavior, clicks). All these diverse data can be expressed in KGs. This section presents basic concepts about KGs and reviews some KG-based RSs.

### 2.1 Knowledge graphs

A Knowledge Graph (KG) has a finite set of nodes (vertices) $N$ and a set of edges $L \subseteq N \times R \times N$, each one linking a pair of nodes taken from $N$ via a particular relation type from $R$. Each node $n \in N$ refers to an entity (object) or concept (class), while

[1]Available in: https://datasets.imdbws.com
[2]Available in: https://dbpedia.org

edges are semantic links of type $r \in R$ between two nodes. The Resource Description Framework (RDF) allows KG representation as a set of subject-property-object triples ($\langle head\_node, relation\_type, tail\_node \rangle$), describing facts (statements), where the $head\_node$ (subject) refers to an entity, the $tail\_node$ (object) refers to an entity or literal and the $relation\_type$ refer to the type of semantic link between the former. Figure 1 exemplifies part of a KG for the movie domain. Its nodes represent users (on the left), movies, genres, actors, and directors. Its edges represent relations among those entities, such as user-item interactions, the classification of movies in one or more genres, the actors that appear in these movies, and their directors.

In KG-based RSs, the side information represented by a KG can be gathered from different data sources:

- Cross-domain data sources: provide information of diverse domains. For example, DBpedia (Lehmann et al. 2015) and Freebase (Bollacker et al. 2008) contain an expressive number of facts about varied things (e.g., people, places, events, books, movies).
- Specific-domain data sources: have their information scope limited to certain domains. For example, IMDb is a movie domain dataset. LinkedMDB (Hassanzadeh & Consens, 2009) was the first open linked dataset connecting web resources about movies from several sources (e.g., IMDb, Rotten Tomatoes, FreeBase, DBpedia).
- Social network APIs: Some KGs include data gathered from social media through APIs like Facebook Graph API and Twitter API.

## 2.2 Models for KG-based RS

KG-based RSs can exploit KGs by using three main strategies (Guo et al. 2020): i. embedding-based methods that enrich item and user representations with embedded encodings of relevant entities that are related to them (Zhang et al. 2016; Zhang & et al. 2018; Piao & Breslin, 2018; Cao et al. 2019); ii. path-based methods that exploit connectivity patterns in the KG to enhance recommendation (Yu et al. 2013; Sun et al. 2018); and iii. unified methods that benefit from both previous strategies (Wang et al. 2018; 2019).

### 2.2.1 Embedding-based Methods

KG-based RSs use KG Embedding (KGE) (Wang et al. 2017) to exploit KG information. The general idea of KGE-based RS methods is to learn latent vectors representing users and items, together with a function that maps a user-item pair into a preference score. There are two main classes of KGE algorithms: i. translation distance models, such as TransE (Bordes et al. 2013) , TransH (Wang et al. 2014), TransR (Lin et al. 2015); and ii. semantic matching models, such as DistMult (Yang & et al. 2015) and ComplEx (Trouillon et al. 2016). These models can be applied to link prediction (also called KG completion), which can potentially enhance recommendation quality.

Some KGE methods consider only a KG representing the side information about recommended items, ignoring user interactions. For example, Zhang et al. (Zhang et al. 2016) propose CKE, which unifies various types of side information in a collaborative filtering framework. They combine item attributes represented in a KG with textual and visual contents (e.g., movie synopsis and posters, respectively). While information about items is encoded with the TransR, the textual feature and the visual feature are extracted with the autoencoder architecture. These representations are aggregated along with the collaborative filtering latent vectors obtained from the user-item matrix.

Other KGE methods exploit KGs representing items, users, their interactions, and side information. For instance, the Collaborative Filtering with Knowledge Graph (CFKG) method proposed by Zhang and et al. (2018) applies TransE on the KG representing user interactions (e.g., review, brand, category, bought-together) and maps the recommendation problem to a $\langle user, r, item\rangle$ triple prediction.

Furthermore, some KGE methods exploit knowledge sharing between the recommendation task and the KG completion task. For instance, Cao et al. (2019) proposed KTUP which consists of two distinct modules for both tasks. In the recommendation module, the loss function is based on a proposed translation-based model, TUP, that models the correctness of embedded vectors representing both observed and unobserved user-item pairs. In the KG completion module, a hinge loss based on TransH is adopted. Moreover, Cao et al. adopt a filtering approach to remove irrelevant instances of the training dataset. CoFM (Piao & Breslin, 2018), by their turn, jointly trains Factorization Machines and TransE by sharing parameters or regularization of aligned items and entities.

### 2.2.2 Path-based methods

Traditionally, path-based KG-based RSs are trained with KGs structured as a Heterogeneous Information Network (HIN) (Guo et al. 2020). These RSs compute the similarity between entities based on meta-paths. HeteRec (Yu et al. 2013) and RKGE (Sun et al. 2018) are examples of path-based RSs. HeteRec leverages the meta-path similarities to enrich the user-item interaction matrix with refined latent vectors of users and items in different meta-paths of the HIN. RKGE automatically mines the path relation between users and items, without manually defining meta-paths. Specifically, RKGE first enumerates user-item paths that connect a given user and item throw a sequence of heterogeneous relations with a length constraint. Then, a recurrent network encodes all user-item paths into a single vector representing the semantic relation between the user and item. Finally, this vector is mapped into a preference score.

### 2.2.3 Unified methods

Unified methods are a recent research trend that combines embedding-based and path-based methods. They are based on embedding propagation algorithms that exploit both the semantic representation of KG entities and relations. One of the first works to introduce embedding propagation on the item graph was RippleNet (Wang et al. 2018). First, it assigns entities in the KG with initial embeddings. Then, it samples multi-hop ripple sets from the KG, i.e. paths from user to neighbor entities satisfying a length constraint. These data are used to refine the embeddings based on the similarities between the head and tail entities of each triple. In this way, RippleNet propagates the user's preference from historical interests, along the path, to unobserved items in the KG.

Finally, Wang et al. (2019) proposed KGAT, which explores the propagation mechanism in the KG. KGAT refines TransR entity embeddings through multi-hop neighbors to directly models the high-order relations between users and items.

## 3 Related work

Dataset size and quality pose challenges to various systems that employ Machine Learning (ML), including many RSs. In particular, the large volumes of data needed to achieve high levels of effectiveness can have a considerable impact on computational costs (Paun, 2020). Meanwhile, irrelevant and noise data may degrade performance (Smyth & Keane, 1995). Thus, lots of research have been conducted to mitigate these problems, resulting in several approaches to reduce input data size and noise prior to ML. In this section, we focus the discussion on instance pruning, case-base editing, dimensionality reduction, and graph summarization approaches. The first three kinds of approaches were proposed in seminal works that influenced many works during the last few decades and already proved to be effective. Nevertheless, further research is still needed to determine the full potential of graph summarization approaches for improving ML by properly reducing and compressing, for instance, facts of a Knowledge Graph (KG) that can be useful as side information.

Instance prunning (or prototype selection) (Wilson, 1972; Wilson & Martinez, 1997; Garcia et al. 2012) cuts down tagged data to alleviate computational costs and improve results of ML tasks like *k*-nearest neighbour classification (Cunningham & Delany, 2020). Similarly, case-base editing (also known as Case-Base Maintenance - CBM) (Leake & Wilson, 1998; Arshadi & Jurisica, 2004) tries to increase the efficiency and accuracy of Case-Based Reasoning by reducing the case base. Such data reduction approaches can be categorized as competence preservation or competence enhancement (Nakhjiri et al., 2020). Competence preservation removes redundant data that do not contribute to the quality of the results, while competence enhancement removes noisy and corrupt data. These approaches have the potential to decrease storage and computation time and improve accuracy. The Edited Nearest Neighbor (ENN) (Wilson, 1972; Wilson & Martinez, 1997) was the first formal competence enhancement method to remove instances from the training dataset when their predicted class does not agree with their k-nearest neighbors. CBM approaches aimed at preserving competence are based on a set of properties of the cases (e.g., the reachability and coverage sets introduced by Smyth (1998)) to identify which cases are selected to compose the knowledge base (the edited set) for reasoning. A recent review of ENN and CBM extensions and variations can be found in Nakhjiri et al. (2020).

Dimension Reducing Techniques (DRTs) (Vlachos et al. 2002) convert high dimensional feature vectors into some lower-dimensional encoding. Several works review DRT for ML (Peluffo-Ordóñez et al., 2014; Reddy et al. 2020; Sorzano et al., 2014; Ayesha et al., 2020). A DRT can rely on feature selection and/or feature transformation (Van Der Maaten et al., 2009). The former reduces dimensionality by selecting a subset of variables (dimensions) that are relevant for a particular problem. The latter transforms the original vectors into more compact ones in which redundant and irrelevant features are removed but relevant properties are preserved. According to (Ayesha et al., 2020), using a DRT can be an efficient way to reduce input variables prior to ML. However, choosing a suitable DRT for a particular situation can be quite intricate, as there is a wide variety of DRTs, with varying purposes and constraints, for distinct kinds of data.

The use of Graph Summarization (GS) (Liu et al. 2018) is another alternative to reduce inputs for ML, when these inputs include graphs, such as (extracts of) KGs representing side information that can help ML. GS can be complementary to other data reduction approaches and more appropriate for certain circumstances. It can take into account aspects as graph topology and nodes/edges similarities. In addition, GS can be less dependent on the ML method as it can consider or not ML model training when summarizing the graph.

Liu et al. (2018) provide a taxonomy of GS methods based on their input types and employed techniques. Čebirić and et al. (2019) provide a taxonomy of existing RDF summarization methods. Some GS methods only consider the connectivity of the KG represented as an unlabelled graph or adjacency matrix. For instance, Fiorucci et al. (2020) propose a GS method based on Szemerédi's Regularity Lemma for separating structural information from noise in large graphs. Other GS methods adopt latent-based techniques to obtain a low-dimensional representation in a latent space. For instance, DeepLENS (Liu et al., 2020) exploits word embedding for encoding triples and generating an optimal summary by selecting the $k$ most salient triples.

In this paper, we investigate the GS potential to condense side information represented in potentially large KGs. The goal is to reduce storage and processing costs while minimizing information loss and preserving patterns and/or properties suitable for KG-based RSs. We propose a GS method that, like some previous ones, adopts graph embedding (ComplEx in our current implementation). However, differently from other works, we cluster KG nodes referring to entities that are similar in the embedded space and merge the nodes participating in each cluster into a single supernode. The node clustering can be done on a single-view (whole KG) or multi-views (a number of separate sub-graphs referring to distinct KG aspects such as movie genres, actors, and directors). We employ $k$-Means with varying values of $k$ to cluster nodes in the KG single-view and multi-views to generate alternative summaries. In the following, we discuss related work about GS, multi-view clustering, and its applications to RS in general and KG-based RS.

In the past decade, several technologies have exploited KG multi-views. The multi-view clustering algorithm was introduced by Bickel and Scheffer (2004) and since then it has been exploited to leverage diversity, accuracy, and robustness of KG partitions. In such methods, graphs representing different viewpoints of the same subjects are first fused and then clustered. Other approaches try to find a way to maximize clustering quality within each view while taking clustering consistency across different views into consideration (Yang & Wang, 2018). For instance, Hussain et al. (2014) proposed a multi-view clustering algorithm for documents that applies a single-view clustering method to each view and generates three similarity matrices based on clustered partitions. Then, it aggregates these matrices to obtain a unified similarity matrix for the final clustering. Xue et al. (2015) proposed a group-aware multi-view fusion method for image clustering, which adopts different weights to characterize the similarity between distinct groups. Nie et al. (2017) propose a multi-view learning model with adaptive neighbors, which is evaluated with image datasets. It performs semi-supervised classification and local manifold learning and clustering, simultaneously, and automatically sets a weight coefficient per view.

While GS recently has become popular in graph visualization and query optimization, its use in KG-based RSs is relatively new and unexplored. Wu et al. (2015) propose GCCR, a media trust-aware RS to recommend news sources to users of heterogeneous social networks. GCCR employs GS and content-based clustering based on K-SNAP (Zhang et al., 2010) to partition a user collection into different interest groups.

Guo et al. (2015) employed a multi-view clustering algorithm that considers user rates and social trust relationships to improve the accuracy and coverage of RSs. First, clusters are generated using the K-medoids algorithm. Then, a Support Vector Regression model is adopted to predict items for users that belong to more than one cluster. Yu et al. (2018b) propose a multi-content clustering collaborative filtering model (MCCCF) for RSs. It applies a multi-view clustering on the traditional content-based recommendation to discover the latent semantic relevance between items or/and users. Then, it uses the kNN algorithm to exploit their similarity. Finally, the similarity matrix can be applied to traditional CF methods.

To the best of our knowledge, no studies have proposed the use of GS algorithms as a data preprocessing step for any KG-based RS yet, neither analyzed the GS impacts on the training time and the effectiveness of KG-based recommendation models. To perform this analysis, we propose a simple but yet effective latent-based GS method that combines latent semantics-based embedding (ComplEx) and clustering (K-Means) in single-view and multi-view approaches.

## 4 Knowledge summarization for KG-based RS

In this work, we propose a new Graph Summarization (GS) method, called KGE-K-Means Summarization, and investigate its use as a preprocessing step for KG-based RSs. Taking into consideration that GS is application-dependent, first (in Section 4.1), we describe our GS method, which has two alternative approaches for merging nodes based on their similarity in a graph embedding space: single-view and multi-view. In Section 4.2, we describe how KG-based RSs can use our GS method for generating alternative KG summaries, based on the single-view or multi-view approaches and with increasing summarization ratios, to speed up the training of KG-based recommendation models. Moreover, we present the methodology adopted to evaluate the impacts of the KGE-K-Means summarization on the efficiency and the effectiveness of different KG-based RSs. Afterward, in Sections 5 and 6, we report experiments using movie datasets and present their results.

### 4.1 KGE-K-Means summarization

Currently, available GS methods usually applied in graph visualization or query optimization may not be suitable for KG-based RS, because summarization is application-specific. Thus, we propose a new GS method, called KGE-K-Means summarization, for KG-based RS. It merges KG nodes based on entity similarity in graph embeddings. Our premise is that it can condense side information to speed up recommendation model training, though still allowing the KG-based RS to capture preferences from the KG summary to generate proper recommendations.

Figure 2 illustrates our GS method. We consider that the *Original KG* provided as input for our method represents side information (entities and relations) related to items to be rec-

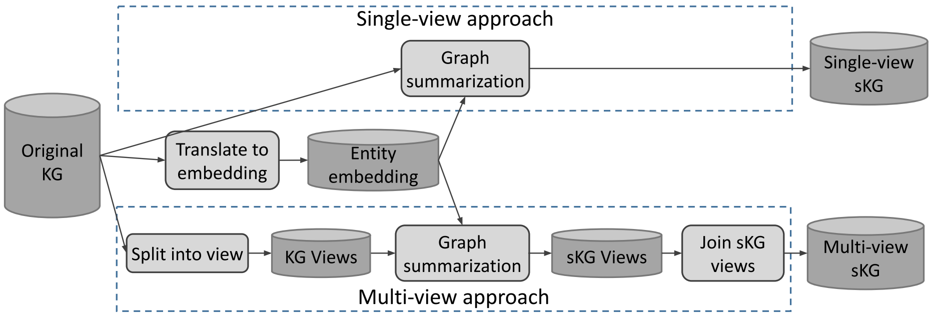

**Fig. 2** KGE-K-Means summarization with the single-view and the multi-view approaches

ommended. This information can be collected from domain-specific sources (e.g., IMDb) and/or general linked data collections (e.g., DBpedia). Our method only summarizes entities of the side information. Users and items are not subject to GS to maintain the independence between the KG preprocessing and the KG-based RS that produces the recommendations.

In the proposed GS method, the *Original KG* is first translated to embedding, by using models such as ComplEx (Trouillon et al. 2016). The K-Means clusterization algorithm relies on the KG embedding to efficiently calculate semantic similarities or distances between entities for clustering them. Nodes in each cluster are merged into a supernode that represents the respective set of entities. Then, triples (facts) from/to summarized entities are replaced by triples from/to the respective supernode instead. These changes can generate duplicate triples which are removed, contributing to reduce the KG size. For example, during the summarization of a movie domain KG, the nodes representing the actors *Anatoli Davydov*, *Julianne Moore*, and *Sylvester Stallone* can be merged into a supernode due to their semantic similarity. Then, if these three actors appear in the same movie (e.g., *Assassins* released in 1995), the *sKG* will have a single link between this movie and the supernode representing the cluster of actors, while the *Original KG* declares three triples, each one linking the movie to an actor.

KGE-K-Means summarization allows distinct summarization ratios by varying the parameter $k$ of the k-Means clustering algorithm. The number of supernodes is the number of clusters found. The summarization ratio is defined as $\alpha = 1 - n/N$, where $N$ is the number of entities in the *Original KG* and $n$ is the number of entities and supernodes in the *Summarized KG*. Therefore, the higher this ratio, the larger the number of entities grouped into supernodes.

As illustrated in Fig. 2, KGE-K-Means summarization supports two approaches for clustering-based GS: single-view and multi-view. The *single-view approach* applies the GS method to the entire *Original KG* and generates a single-view summarized KG (*Single-view sKG*). Therefore, it can merge into the same supernode entities of different types (e.g., actors, directors, genres). For example, a supernode can merge the actor *Sylvester Stallone* and the genre *Action*.

The multi-view GS approach is proposed to evaluate a more restrictive summarization. In this approach, illustrated at the bottom of Fig. 2, the task *Split into views* breaks up the *Original KG* into a set of KG views. Each KG view represents a facet (subset) of the side information that can be used to describe an aspect of the items to be recommended. Examples of aspects for describing movies are *Genre*, *Actor*, and *Director*. In our current implementation, we create a KG view $v_i$ by first choosing one relation type $r_i$, and then collecting triples of the form $\langle e_h, r_i, e_t \rangle \subseteq N \times R \times N$, where $e_h$ or $e_t$ refers to an item to be recommended (e.g. movie), and the relation type $r_i$ restricts the other entity of the triple to a class of interest. For instance, the relations *hasGenre*, *hasActor*, and *hasDirector*, restrict the class of the entity related to the movie to *Genre*, *Actor*, and *Director*, respectively, enabling the generation of three distinct KG views. After creating the views, the task *Graph summarization* summarizes each KG view independently, producing a summary *sKG* for each KG view. Note that, differently from the single-view approach, the generated supernodes here can only group entities that belong to the same KG view.

Finally, the task **Join sKG views** connects the independently generated *sKG views* to make a single summarized KG called *Multi-view sKG*. It is done by considering the facts (edges) of the KG. Remind that, during the graph summarization, the facts to/from entities that were merged into supernodes had these entities substituted by the respective supernodes. Duplicate facts generated by these substitutions were eliminated during the graph summarization task too.

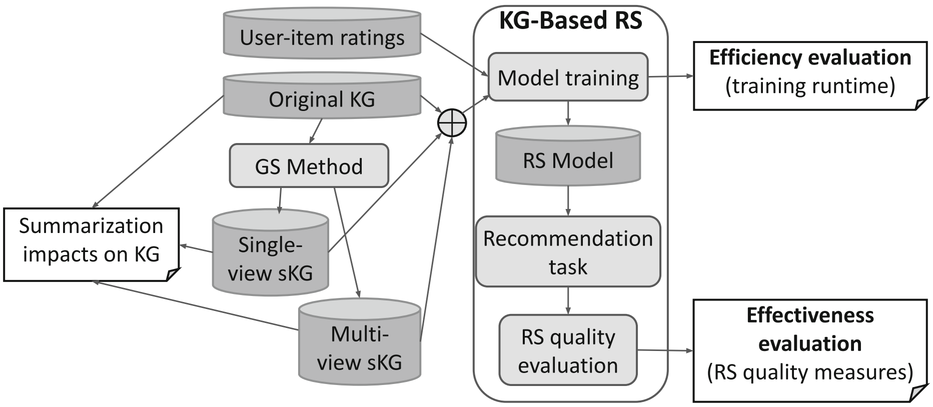

**Fig. 3** Evaluation of the KGE-K-Means summarization impacts on KG-based RS

### 4.2 Evaluating the impacts of GS in KG-based RSs

Due to the lack of a consolidated methodology to evaluate the efficiency and effectiveness of machine learning models for recommendation systems (Paun, 2020), we adopted a simplified method to evaluate the impacts of the KGE-K-Means summarization on the efficiency and the effectiveness of different KG-based RSs. Figure 3 illustrates our evaluation methodology, which addresses GS and then two major tasks of a generic KGE-based RS: *Model Training* and *Recommendation Task*. The *Model Training* learns embeddings from ratings and KG triples to build the RS model. The *Recommendation Task* ranks items based on relevance scores, computed with the learned embeddings of users and items, to return the top-$N$.

Our objective is to evaluate KGE-K-Means summarization impacts on KG size reduction and mainly in the efficiency and effectiveness of KG-based RSs. For *Efficiency Evaluation*, we adopted RS model training time as the efficiency metric because the RS Model Training is the most costly task of KGE-based RS and the most impacted by GS. Reducing this processing time also impacts the efficiency of the recommendation process as a whole. For *Effectiveness Evaluation*, we consider RS quality metrics that are widely adopted to evaluate the effectiveness of KG-based RS, including precision, recall, nDCG, and mAP. Therefore, by assessing the recommendation quality (final result), we are evaluating the impact of the proposed approach on the effectiveness of the whole recommendation process. We also carry out a comparative study to evaluate the efficiency-effectiveness trade-off in different training data scenarios, considering different representations of the side information: the *Original KG* versus *Single-view sKGs* and *Multi-view sKGs*. Different summarization ratios are used in both approaches. The training cost reduction and the effectiveness of the RS models produced by using the Original KG (baseline) are compared with the ones obtained by using its distinct summaries.

## 5 Experiments

This section describes the experiments that we have conducted to evaluate the KGE-K-Means summarization method. We perform a comparative evaluation in that the baselines

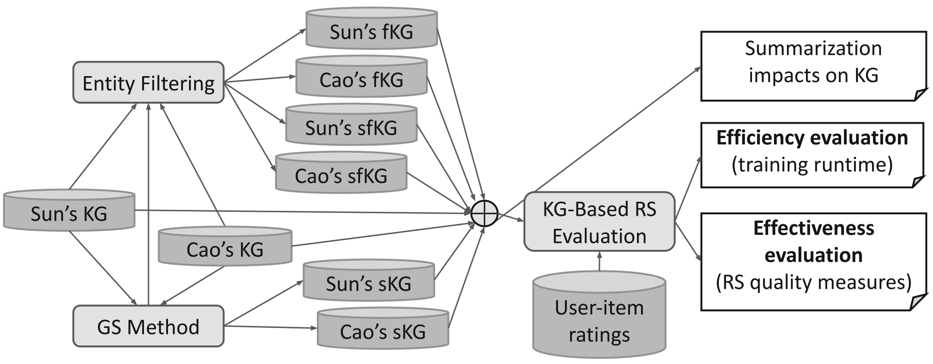

**Fig. 4** Sequence of tasks performed on datasets

are KG-based RSs trained using the original (non-summarized) KG. We compare their efficiency and effectiveness with those of KG-based RSs trained using summarized KGs obtained by KGE-K-Means summarization.

The evaluation method adopted is that described in Section 4.2. Figure 4 outlines the inputs, the sequence of performed tasks, and the outputs of our experiments. The original KGs used as inputs were taken from two datasets in the movie domain (*Sun's KG* (Sun et al. 2018) and *Cao's KG* (Cao et al. 2019)). The combination of the KG preprocessing tasks, namely *GS* and *Entity Filtering*, generates the filtered KGs (*fKGs*), the summarized KGs (*sKGs*), and summarized filtered KGs (*sfKGs*) that are used as alternatives to the original KG for training KG-based RS models.

The *KG-based RS evaluation* task in Fig. 4 is detailed in Fig. 3. The *Summarization impacts on the KG*, as well as training time and effectiveness of RS models, are measured for Sun's and Cao's original KGs and their derivatives (*sKGs*, *fKGs*, and *sfKGs*) with increasing summarization ratios. The following subsections provide more details about the implementation, datasets, KG-Based RS methods, parameters, and evaluation metrics used in the experiments.

## 5.1 Datasets

The *User-item ratings* and the original *Sun' KG* and *Cao's KG* used as inputs in our experiments were obtained from two datasets of the movie domain:

- Sun's dataset (Sun et al. 2018)[3]: contains a small subset of user-item ratings from MovieLens 1M (Harper & Konstan, 2015), with 99,975 ratings made by 943 users for 1,675 movies. The average number of ratings per user is 106 and the rating sparsity ratio is 93.6%. The side information present in this dataset (*Sun's KG* in Fig. 4) was gathered from IMDb data. This KG contains 12,311 subject-property-object triples with 3 property types. Their subjects and objects can be movies or 5,124 other entities (including 24 genres, 3,947 actors, and 1,153 directors). The sparsity ratio of this KG is 99,9%.
- Cao's dataset (Cao et al. 2019)[4]: has a subset of user-item ratings from MovieLens 1M, with 998,539 ratings made by 6,040 users for 3,260 movies. The average number of

[3]Available in: https://github.com/sunzhuntu/Recurrent-Knowledge-Graph-Embedding
[4]Available in: https://github.com/TaoMiner/joint-kg-recommender

**Table 1** Statistics of Sun's and Cao's *Original KG*

| | | Sun's KG | Cao's KG |
|---|---|---|---|
| User-Item Interactions | # Users | 943 | 6,040 |
| | # Items | 1,675 | 3,260 |
| | # Ratings | 99,975 | 998,539 |
| | Avg. Ratings/User | 106 | 165 |
| | Sparsity Ratio | 93,6% | 94,9% |
| Side Information t # | # Entities | 5,124 | 11,443 |
| | Relation types | 3 | 20 |
| | t # Triples | 12,311 | 428,777 |
| | KG Sparsity Ratio | 99,984% | 99,983% |

ratings per user is 165 and the rating sparsity ratio is 94.9%. The side information in *Cao's KG* was reused from LODRecSys[5] and refined to combine MovieLens 1M and DBpedia triples. This KG contains 428,777 triples describing 20 properties of movies (e.g., *cinematography, producers, story, series, basedOn, starring, director, award*), 11,443 entities, and a KG sparsity ratio of 99,9%. Therefore, it covers a greater number of entities than Sun's KG.

Table 1 provides a statistical overview of the contents of these two datasets. Notice that Cao's KG is pretty bigger than Sun's KG. Both took ratings from a well-known dataset of the movie recommendation domain, MovieLens 1M. The sparsity ratio of user-item interactions was calculated as $n/u * i$, where $n$ is the number of ratings, $u$ is the number of users and $i$ is the number of items. The KG sparsity ratio, by its turn, was calculated as $n/r * e^2$, where $n$ is the number of triples, $r$ is the number of relation types and $e$ is the number of entities representing side information that are directly related to entities representing movies.

## 5.2 KG preprocessing methods and parameters

As illustrated in Fig. 4, the following summarization methods were tested during the KG preprocessing step of our experiments:

– *Entity filtering*: removes infrequent entities from the KG. This rudimentary pruning method is adopted in some KG-based RSs, including (Sun et al. 2018) and Cao et al. (2019). In our experiments, as in Cao et al. (2019), entities are considered infrequent when appearing in at most 10 triples.
– *GS method*: is the KGE-K-Means Summarization method proposed in this work. In our experiments, the ComplEx (Trouillon et al. 2016) model is trained with: embedding space dimensionality (k) of 150, eta at 20, epochs at 150, batches at 100, adam optimizer with learning rate at 0.1, multiclass negative loss likelihood, and L2 regularizer with lambda in $10^{-4}$. In the K-means clustering algorithm, the number of clusters to be found is determined in accordance with the summarization ratio ($\alpha$) chosen in each experiment. The number of times the k-means algorithm runs with different centroid seeds is 50, and the maximum number of interactions is 500.

[5] Aval.: https://github.com/sisinflab/LODrecsys-datasets/tree/master/Movielens1M

- *GS method+Entity filtering*: is the combination of two previous methods, exploiting both the merging of similar entities in supernodes and the elimination of infrequent entities.

## 5.3 KG-based RSs and parameters

We evaluated the following state-of-the-art KGE-based RS models trained with the *Original KG* and its derivative summarized versions (fKGs, sKGs or sfKGs): CFKG, which applies TransE in a unified graph with users, items, entities, and relations; CKE, which combines various item embeddings from different sources including TransR on KG; CoFM, which jointly trains FM and TransE; and KTUP, which jointly learns representation and KG completion models.

The hyperparameter settings[6] were empirically found out for each method, departing from the incomplete listing of settings for MovieLens 1M made available in Cao et al. (2019). We have set the parameters considering the features of both datasets (Cao's and Sun's) used in our experiments and the need to establish conditions to carry out a comparative analysis of the training time (i.e., by restricting the number of training epochs in the same way).

For CFKG, CKE, and CoFM we use the pre-trained embeddings BPRMF (Matrix Factorization method) and TransE, while KTUP uses TUP and TransH. The TUP number of preferences (proportional to the number of relation types) was set in 3 for Sun's KGs and 20 for Cao's KGs. For CFKG, CKE, CoFM, and KTUP methods, we set the joint hyperparameter $\lambda$ as 0.5. In BPRMF and TUP, we adopt the $L_2$ regularization coefficient $10^{-5}$ and Adagrad optimizer. In the other methods, we adopted $L_2$ 0 and Adam optimizer. The learning rate is 0.001 for TransE and TransH, and 0.005 for the remaining methods. All methods use batch size at 256 and embedding size at 100. We set the learning rate at 0.001 for TransE and TransH, and 0.005 for the remaining methods. The maximum number of epochs and the use of the early-stopping strategy are set for each experimental scenario described in Section 5.5.

## 5.4 Evaluation metrics

As presented in Section 4.2, we evaluate the impacts of the proposed GS method in terms of KG size reduction, efficiency of the training model, and effectiveness of the KG-based RSs. We adopt training time in seconds as an efficiency metric. In addition, we use the following effectiveness metrics:

- Precision at N (p@N): the fraction of the top-N recommended items that are relevant to the user. The mean for all users is the final precision.
- Recall at N (r@N): the proportion of relevant items in the top-N recommended items. The mean for all users is the final recall.
- Normalized Discounted Cumulative Gain (nDCG@N): the sum of the relevance of the top-N recommended items penalized by the logarithm of the respective rank. This sum is normalized by the ideal DCG, i.e. considering all relevant items for a user in a monotonically decreasing order. The mean for all users is the final nDCG.

[6]The complete list of parameters is available in github.com/juarezsacenti/kg-summ-rec

– Mean Average Precision at N (MAP@N): the sum of the multiplicative inverse of the position (rank) of each relevant item in the top-N recommended items, divided by the number of hits. The mean for all users is the final MAP.

### 5.5 Experimental scenarios

The goal of our experiments is to evaluate the effects of the KGE-K-Means summarization on the cost for training the recommender models and on the recommendation effectiveness. For this purpose, we defined two experimental scenarios. The first one aims to evaluate the impacts of increasing summarization ratios on model training time and recommendation effectiveness. In this scenario, the models based on KG embedding adopted by each KG-based RS are trained with a fixed number of epochs to enable a fair comparison of their training times using distinct summarizations of the KG. This experimental scenario adopts the Sun's dataset and 5-fold cross-validation with random partitioning following the proportions 6:2:2 for training, validation, and test, respectively. We restricted the training of TransE and TransH to 1000 epochs and the training of CFKG, CKE, CoFM, and KTUP to 500 epochs. These parameters were chosen to avoid overfitting, speed up the processing, and provide similar conditions of training. Moreover, as the BPRMF and TUP are item recommendation models that consider only user-item ratings (which are not affected by the GS), we use an early-stopping strategy for training.

In the second experimental scenario, we evaluate the impacts on the effectiveness of the KG-based RSs using an early-stopping strategy for model training that reaches higher numbers of epochs than in the first scenario. Therefore, in this scenario, the number of epochs in each training is variable. The early-stopping strategy stops training before the performance stops improving. This experimental scenario allows comparing the optimal effectiveness of the KG-based RSs. These experiments used the two movie datasets (Cao and Sun) with hold-out validation. We construct training, validation, and test subsets by randomly splitting each dataset based on the proportion 7:1:2.

### 5.6 Implementation

We implemented the KGE-K-Means summarization method using Python 3.7, Ampligraph (Costabello & et al. 2019) 1.3.2 translation methods, and Scikit-learn 0.23.2 clustering methods. The KG-based RSs employed were CFKG, CKE, CoFM, and KTUP from Cao's project[7]. We standardize the evaluation of RS models by adapting them to use CaseRecommender (Da Costa et al. 2018) 1.1.0 evaluation metrics. The experiments were executed in an Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz with 10 physical cores (HT enabled), and two NUMA nodes (20 physical cores + HT). This server has 128 GB of RAM available and an NVIDIA Tesla K40c.

## 6 Experimental results and analysis

In this section, we first evaluate the KG reduction achieved by the proposed GS method, by presenting and comparing various statistics of original and summarized KGs in Section 6.1. Then, in Sections 6.2 and 6.3, we present the results regarding the GS impacts on model

[7] Available in: https://github.com/TaoMiner/joint-kg-recommender

**Table 2** Single-view KGE-K-Means summarization impacts in the KG

| Metrics | Original | KGE-K-Means Summarization ratios | | |
|---|---|---|---|---|
| | KG | $\alpha$=25% | $\alpha$=50% | $\alpha$=75% |
| #Genres | 24 | 24 (100%) | 24 (100%) | 24 (100%) |
| #Directors | 1,153 | 1,104 (95.8%) | 911 (79%) | 211 (18.3%) |
| #Actors | 3,947 | 2,715 (68.8%) | 1,639 (41.5%) | 1,107 (28%) |
| #Entities | 5,124 | 3,843 (75%) | 2,562 (50%) | 1,281 (25%) |
| #hasGenre | 3,974 | 3,974 (100%) | 3,974 (100%) | 3,974 (100%) |
| #hasDirector | 1,800 | 1,746 (97%) | 1,672 (92.9%) | 1,664 (92.4%) |
| #hasActor | 6,537 | 5,289 (80.9%) | 4,207 (64.4%) | 3,806 (58.2%) |
| #Triples | 12,311 | 11,009 (89.4%) | 9,853 (80%) | 9,444 (76.7%) |
| KG Sparsity | 99.984% | 99.975% | 99.950% | 99.808% |

training time and recommendation quality. Finally, in Section 6.4 we discuss these results. The results presented in Section 6.2 were obtained by using 5-fold cross-validation and the other ones with hold-out.

## 6.1 Summarization impacts on KGs reduction

Tables 2 and 3 present the impacts of applying KGE-K-Means summarization on Sun's KG using single-view and multi-view approaches, respectively. Both tables provide a quantitative analysis of these KGs and the summarization results for $\alpha$ of 25%, 50%, and 75%. In these tables, the first and second groups present the number of entities and triples describing the movie side information (disregarding items, users, and associated interactions) before and after the summarization, respectively. The last line of these tables presents the sparsity score estimated in each KG version.

As can be seen in Table 2, in the single-view GS approach, the number of entities after summarization is defined directly by the summarization ratio. However, the ratio in each

**Table 3** Multi-view KGE-K-Means summarization impacts in the KG

| Metrics | Original | KGE-K-Means Summarization ratios | | |
|---|---|---|---|---|
| | KG | $\alpha$=25% | $\alpha$=50% | $\alpha$=75% |
| #Genres | 24 | 18 (75%) | 12 (50%) | 6 (25%) |
| #Directors | 1,153 | 865 (75%) | 577 (50%) | 289 (25%) |
| #Actors | 3,947 | 2,961 (75%) | 1,974 (50%) | 987 (25%) |
| #Entities | 5,124 | 3,844 (75%) | 2,563 (50%) | 1,282 (25%) |
| #hasGenre | 3,974 | 3,873 (97.5%) | 3,588 (90.3%) | 3,271 (82.3%) |
| #hasDirector | 1,800 | 1,673 (92.9%) | 1,669 (92.7%) | 1,667 (92.6%) |
| #hasActor | 6,537 | 5,542 (84.8%) | 4,503 (68.9%) | 3,754 (57.4%) |
| #Triples | 12,311 | 11,088 (90%) | 9,760 (79.3%) | 8,692 (70.6%) |
| KG Sparsity | 99.984% | 99.975% | 99.950% | 99.824% |

entity type (Genre, Director, and Actor view) is different. In general, the greater the number of entities of a type, the greater the size reduction. In its turn, the multi-view GS approach (Table 3) reduces uniformly the number of entities per relation type.

In the summarized KGs, the number of triples was reduced to 89.4%, 80%, and 76.7% when summarized using the single-view approach, and was reduced to 90%, 79.3%, and 70.6% when summarized with the multi-view approach, respectively with summarization ratio at 25%, 50%, and 75%. The summarization impact over the number of triples is determined by the number of duplicate triples after summarization. In the Sun's dataset, we have 1,675 movies (Table 1) and 1,800 *hasDirector* relations. Therefore, *hasDirector* is, in general, an N-1 relation, there is a smaller occurrence of triples duplication. The triple pruning was higher in the multi-view approach. This is justified by the fact that this approach generates genres' supernodes. As the relation *hasGenre* has a number of distinct objects (genres) much smaller than relation *hasActor*, the occurrence of duplication of triples with relation *hasGenre* is greater.

Finally, the use of KGE-K-Means summarization resulted in a small sparsity reduction. Our proposal reduces the number of entities $e$ by grouping them in supernodes, reducing the space of triples combining these entities. Considering KG sparsity ratio such as $n/r * e^2$ (described in Section 5.1), the reduction of the number of triples $n$ (elimination of duplicate triples after replacing entities by supernodes) was lower than the impact of the reduction of the number of entities $e$, causing the observed reduction in KG sparsity.

As presented in Section 5.2, the KGE-K-Means summarization can be combined with the infrequent entity filtering to improve the summarization ratio. Figure 5 presents the entities and triples pruning ratios obtained with Sun's dataset when using only infrequent entity filtering (fKG), the summarization ratios when using the KGE-K-Means summarization (sKGs), and the graph reduction ratio when combining both KG preprocessing methods (summarization and entity pruning, sfKGs). Figure 5a presents pruning ratios obtained by the infrequent entity filtering (appearing in 10 or fewer triples). This filtering pruned 74.7% of the entities and, in consequence, 65.6% triples have been removed from the KG. Figure 5b presents the entities and triples summarization ratios obtained using KGE-K-Means summarization using the single-view and multi-view approaches in different values for $\alpha$

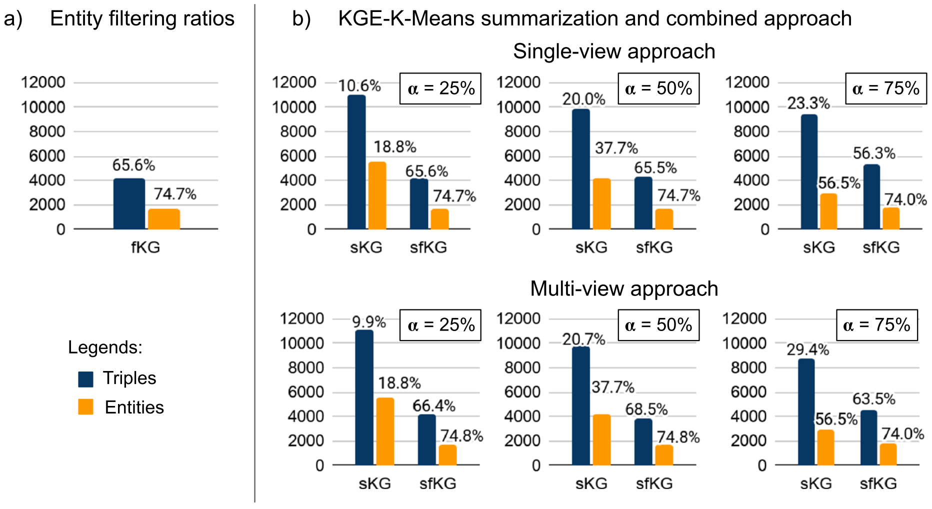

**Fig. 5** Pruning and summarization ratios of entity filtering and KGE-K-Means summarization

(25%, 50%, and 75%). As can be seen in this figure, the summarization ratios observed in the summarized KGs (sKGs) are smaller than the pruning ratios when used the entity filtering method. The entity summarization ratio is smaller than $\alpha$ because our GS approach does not summarize movies. An important observation is that the KG size reduction achieved with the combination of the GS summarization and entity filtering is a little smaller than what was obtained by the entity filtering. Therefore, we can observe that the GS summarization reduces the number of infrequent entities.

## 6.2 Training costs versus recommendation effectiveness

As explained in Section 5.5, in the first experimental scenario, we analyze the impacts of the KGE-K-Means summarization method on training costs and recommendation effectiveness of four KGE-based models in the experiments using Sun's dataset. Table 4 presents the results of these experiments in terms of RS effectiveness metrics (precision, recall, ndcg, and map at top-10 recommendations) and training time in seconds. These results refer to 5-folds cross-validation. The bold entries of Table 4 contain the best efficiency and effectiveness measures (lowest time and highest accuracy measures) for each RS (CFKG, CKE, CoFM, and KTUP). We applied the non-parametric Wilcoxon test (Wilcoxon, 1945) to evaluate the statistical significance of the precision variations presented in Table 4. First, we look at the acceptance of the null hypothesis that there is no difference in precision between models trained with summarized and non-summarized KGs. We observed that none of the summarized KGs rejected the null hypothesis at a confidence level of 5%. However, some p-value results were close to 0.05 (0.0625 of p-value) in the test. Additionally, we tested if the precision measures in the experiments using models trained with the non-summarized KG are greater than those using models trained with summarized KGs. In this test, only the measurements for the CKE method using the single-view summarization approach with $\alpha$=75 rejected the null hypothesis at a confidence level of 5%, showing a p-value of 0.03125. Finally, we tested if models trained with the non-summarized KG have lower precision than models trained with summarized KGs. In this test, only for the CFKG method using the multi-view approach with $\alpha$=25 and for the CoFM method using the single-view approach with $\alpha$=75, the null hypothesis was rejected at a confidence level of 5% (0.03125 of p-value).

Figure 6 allows comparing the average precision at top-10 recommended items (p@10) with training time (in seconds) of each KGE-based RSs. The training time (dashed lines) is clearly reduced with increasing summarization ratios. This trend was observed in all RS models. The least impact on training efficiency was observed in the CKE model with multi-view and $\alpha$=25%: 0.25% reduction (standard deviation = 30.68s). The biggest impact was in the CoFM model with multi-view and $\alpha$=75%: 5.85% reduction (standard deviation=10.31s). As can be seen in Fig. 6, the summarization ratio has a low impact on the precision (solid lines) of all RS models. It was observed with other effectiveness metrics too.

## 6.3 Recommendation effectiveness with optimal training

In the second experimental scenario, we evaluate the impacts on the effectiveness of the KG-based RSs using an early-stopping strategy on their training. It allows comparing the optimal effectiveness of the KG-based models. Figure 7 presents the recommendation effectiveness measures obtained with the KGE-based RS trained with Sun's dataset (original KG, or sKG at $\alpha$=0) preprocessed by infrequent entity filtering (fKG, or sfKG at $\alpha$=0),

**Table 4** Efficiency-effectiveness evaluation

| RS | GS approach | $\alpha$ | p@10 | r@10 | ndcg@10 | map@10 | time |
|---|---|---|---|---|---|---|---|
| CFKG | Original KG | – | **0.3839** | **0.2343** | 0.7033 | 0.6011 | 2264 |
| | | 25% | 0.3818 | 0.2299 | 0.7040 | **0.6017** | 2240.8 |
| | Single-view | 50% | 0.3761 | 0.2277 | 0.7008 | 0.5976 | 2198.6 |
| | | 75% | 0.3711 | 0.2248 | 0.6960 | 0.5912 | 2182.4 |
| | | 25% | 0.3724 | 0.2248 | 0.6894 | 0.5836 | 2249.4 |
| | Multi-view | 50% | 0.3808 | 0.2292 | **0.7053** | 0.6010 | 2219.8 |
| | | 75% | 0.3748 | 0.2278 | 0.6989 | 0.5958 | **2152.2** |
| CKE | Original KG | – | 0.5216 | 0.2987 | 0.8487 | 0.8026 | 4305.4 |
| | | 25% | 0.5265 | 0.3028 | 0.8539 | 0.8094 | 4242.4 |
| | Single-view | 50% | 0.5268 | 0.3040 | 0.8513 | 0.8058 | 4215.4 |
| | | 75% | **0.5334** | 0.3095 | **0.8573** | 0.8110 | 4191.8 |
| | | 25% | 0.5309 | 0.3068 | 0.8537 | 0.8097 | 4294.6 |
| | Multi-view | 50% | 0.5327 | **0.3096** | 0.8565 | **0.8114** | 4255.6 |
| | | 75% | 0.5254 | 0.3015 | 0.8465 | 0.8015 | **4139.4** |
| CoFM | Original KG | – | **0.4715** | **0.2713** | **0.7709** | **0.6917** | 2575.2 |
| | | 25% | 0.4686 | 0.2705 | 0.7666 | 0.6868 | 2468.2 |
| | Single-view | 50% | 0.4700 | 0.2705 | 0.7700 | 0.6889 | 2469.4 |
| | | 75% | 0.4591 | 0.2625 | 0.7642 | 0.6809 | 2442.8 |
| | | 25% | 0.4673 | 0.2678 | 0.7644 | 0.6847 | 2511.2 |
| | Multi-view | 50% | 0.4678 | 0.2701 | 0.7665 | 0.6862 | 2496.8 |
| | | 75% | 0.4647 | 0.2657 | 0.7639 | 0.6826 | **2424.6** |
| KTUP | Original KG | – | 0.5292 | 0.3069 | 0.8442 | 0.7916 | 5126.2 |
| | | 25% | 0.5290 | 0.3079 | 0.8454 | 0.7938 | 5069.2 |
| | Single-view | 50% | 0.5281 | 0.3056 | 0.8418 | 0.7886 | 5046.8 |
| | | 75% | 0.5266 | 0.3037 | 0.8470 | 0.7934 | 4910.2 |
| | | 25% | 0.5282 | 0.3083 | 0.8411 | 0.7881 | 5065.8 |
| | Multi-view | 50% | 0.5180 | 0.3000 | 0.8285 | 0.7746 | 5004.2 |
| | | 75% | **0.5299** | **0.3086** | **0.8478** | **0.7966** | **4908** |

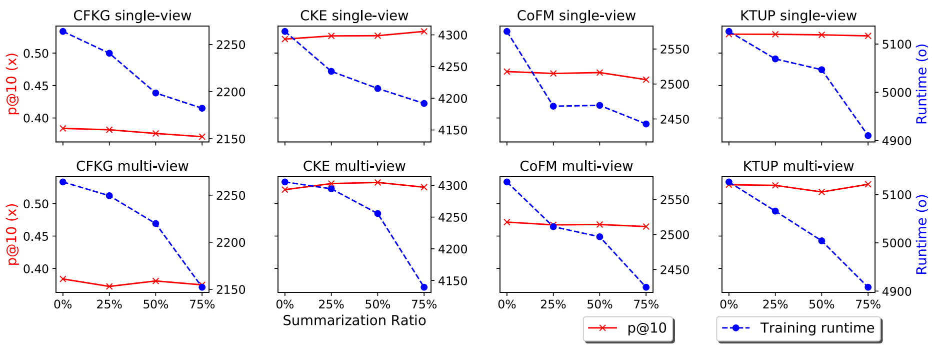

**Fig. 6** Efficiency-effectiveness evaluation

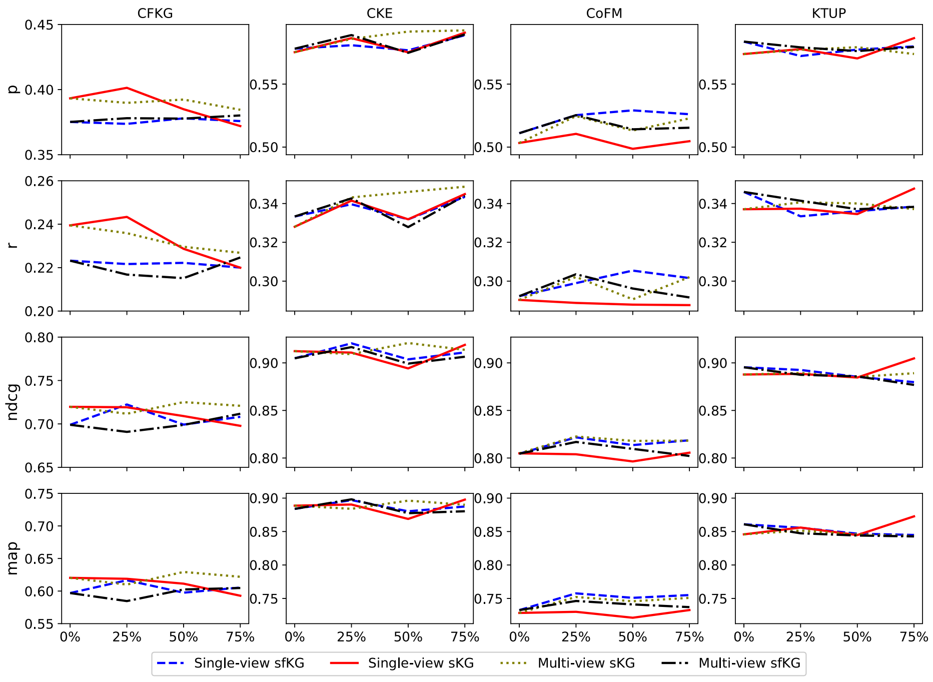

**Fig. 7** Recommendation quality results for each RS model trained using Sun' sKGs and sfKGs

KGE-K-Means summarization (sKGs), and combined methods (sfKGs). The X-axis refers to the KG summarization ratio ($\alpha$), which varies from 0% to 75%.

As can be observed in Fig. 7, the use of KGE-K-Means summarization does not significantly impact the recommendation effectiveness, confirming what was observed in the previous evaluation scenario (Section 6.2). This impact can be better seen in Fig. 8, which allows us to compare the recommendation effectiveness of the evaluated KGE-Based RSs (except CFKG, hidden due to its lower performance) trained with the original KG and its alternative summarizations. CKE trained with multi-view and $\alpha$=75% without entity filtering (sKG-mv-75) was the best KG-based RS concerning precision and recall. The best nDCG result was reached by CKE sKG-mv-75 and the best mAP by CKE sfKG-mv-25 (with infrequent entity filtering).

Finally, we also evaluated the impacts of KGE-K-Means summarization on training time and recommendation effectiveness using Cao's dataset. Due to space limitations, we provide here just some highlights of these experimental results. The single-view 25% approach without filtering (sv-sKG-25) achieved a reduction in training time of 12.82% for CFKG and 22.71% for CoFM. However, the recommendation effectiveness with *Summarized KGs* was slightly worst than the ones obtained with the *Original KG*, in accordance with every metric that we have considered. Further experiments are needed to find configurations that provide the highest effectiveness.

## 6.4 Discussion

Our GS method, KGE-K-Means summarization, proved to be useful for reducing training time, without significant impacts on the effectiveness of state-of-the-art KG-based RSs.

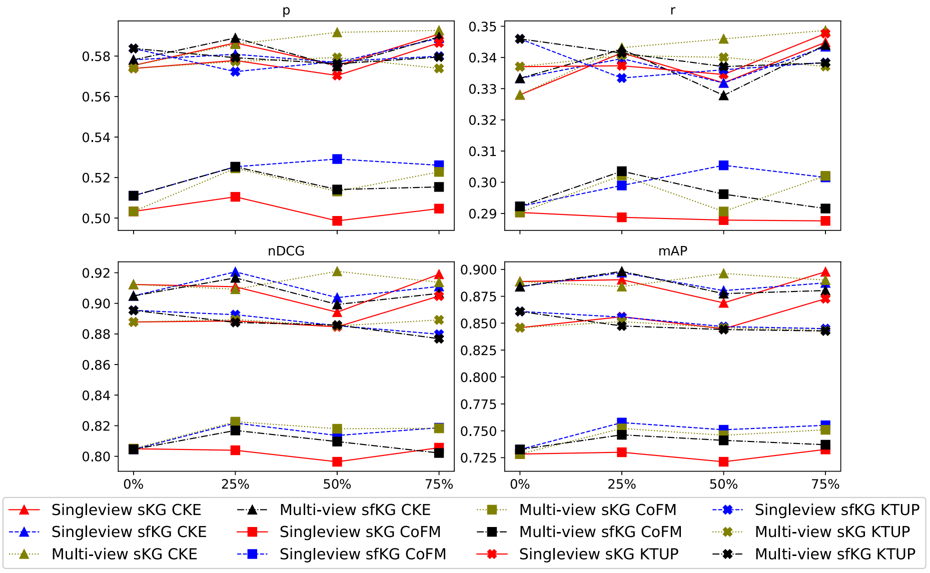

**Fig. 8** Recommendation quality results for Sun' sKGs and sfKGs (without CFKG results)

Since the side information represented in the KG is usually much more stable than user-item interaction data, the KG needs to be summarized much less frequently than the re-training of the KG-based recommendation model must be done. In other words, the same KG summary can be reused for training the RS model again and again with new user-interaction data. In addition, considering the Sun dataset, the speed-up of the training time obtaining by using the summarized KG just once is around half of the time spent by our method to summarize the KG. Consequently, the overall time spent on keeping the recommendation model up-to-date can be significantly reduced.

Though the experiments with the Sun dataset showed positive results, despite the relatively small size of this dataset, the gains can be more significant for bigger KGs. Experiments with the Cao dataset resulted in higher gains due to the larger amount of side information. This suggests that summarization can be more beneficial for bigger datasets.

The multi-view summarization approach enabled more speed-up of the training of most KG-based recommendation models used in our experiments than the multi-view approach for most RSs, notably for a summarization ratio of 75%. On the other hand, there was no significant change in the recommendation effectiveness with distinct summarization approaches (single-view and multi-view). CKE and KTUP were the RS models that achieved the best recommendation quality results for both summarization approaches and distinct summarization ratios, including no summarization (baseline).

## 7 Conclusions and future work

In this work, we investigated the use of Graph Summarization (GS) as a Knowledge Graph (KG) preprocessing step of KG-based Recommender Systems (RS). Moreover, as GS algorithms are application-dependent, we proposed a new GS method applied to RSs. This method, named KGE-K-Means summarization, combines latent semantics-based embedding (ComplEx) and KG node clustering (K-Means) in single-view and multi-view GS approaches. Our experiments using Sun's and Cao's datasets and several state-of-the-art RS models showed that GS realized by our method can reduce training time and the overall time spent for keeping the RS model up-to-date, without significant changes or a clear tendency for gains or losses in terms of RS effectiveness.

Currently, we are performing further experiments with other summarization approaches seeking to improve the noise reduction performance. Additionally, we are analyzing an alternative multi-view approach in which different GS methods can be applied in each graph view. As future work, we intend to evaluate the proposed method in other domains than movies (e.g., amazon-books, last-fm, yelp). Finally, we aim to investigate the possible roles of the summarized side information in the explainability of the recommendation models.

**Acknowledgements** The authors also acknowledge the members of LaPeSD and the GBD laboratories from UFSC, particularly to Prof. Márcio B. Castro, Ph.D and Prof. Luís P. F. Garcia, Ph.D.

**Author Contributions** Related work was carried out by all the authors. The implementation of the proposal and experiments was carried out by J.A.P. Sacenti. All authors drafted, revised and approved the manuscript.

**Funding** This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 and by the Print CAPES-UFSC "Automation 4.0".

**Availability of data and material** The datasets generated during and/or analysed during the current study are available in the sacenti-jiis-2021 repository, https://github.com/juarezsacenti/sacenti-jiis-2021

**Code Availability** The code implemented during the current study are available in the kg-summ-rec repository, https://github.com/juarezsacenti/kg-summ-rec

## Declarations

**Conflict of Interests** The authors have no conflicts of interest to declare that are relevant to the content of this article.

## References

Aggarwal, C. C. et al. (2016). *Recommender Systems* Vol. 1. Berlin: Springer.

Ali, S. M. et al. (2020). Topic and sentiment aware microblog summarization for twitter. *Journal of Intelligent Information System*, 54(1), 129–156.

Arshadi, N., & Jurisica, I. (2004). Maintaining case-based reasoning systems: a machine learning approach. In *Adv in case-based reason* (pp. 17–31). Berlin: Springer.

Ayesha, S., Hanif, M. K., & Talib, R. (2020). Overview and comparative study of dimensionality reduction techniques for high dimensional data. *Information Fusion*, 59, 44–58. https://doi.org/10.1016/j.inffus.2020.01.005.

Bickel, S., & Scheffer, T. (2004). Multi-view clustering. In *ICDM*, (Vol. 4 pp. 19–26).

Bollacker, K. et al. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *Proc ACM SIGMOD int conf manag of data*.

Bordes, A. et al. (2013). Translating embeddings for modeling multi-relational data. In *Adv in neural inf processing syst,* pp 2787–2795.

Cao, Y. et al. (2019). Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conf,* pp 151–161.

Čebirić, Š., et al. (2019). Summarizing semantic graphs: a survey. *VLDB J*, 28(3), 295–327. https://doi.org/10.1007/s00778-018-0528-3.

Costabello, L., et al. (2019). AmpliGraph: a library for representation learning on knowledge graphs. https://doi.org/10.5281/zenodo.2595043.

Cunningham, P., & Delany, S. J. (2020). k-Nearest neighbour classifiers: 2nd edition (with python examples). arXiv:2004.04523.

Da Costa, A. et al. (2018). Case recommender: a flexible and extensible python framework for recommender systems. In *Proc 12th ACM conf recomm syst, recsys '18,* pp. 494–495*, ACM, NY, USA.* https://doi.org/10.1145/3240323.3241611.

Fernandes, B. B., Sacenti, J. A. P., & Willrich, R. (2017). Using implicit feedback for neighbors selection: Alleviating the sparsity problem in collaborative recommendation systems. In *Proc 23rd braz symp multimed and the web, webmedia 2017, gramado, Brazil,* pp. 341–348*, ACM.* https://doi.org/10.1145/3126858.3126896.

Fiorucci, M., Pelosin, F., & Pelillo, M. (2020). Separating structure from noise in large graphs using the regularity lemma. *Pattern Recognition*, 98, 107,070.

Garcia, S. et al. (2012). Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3), 417–435. https://doi.org/10.1109/TPAMI.2011.142.

Guo, G., Zhang, J., & Yorke-Smith, N. (2015). Leveraging multiviews of trust and similarity to enhance clustering-based recommender systems. *Knowledge-Based Systems*, 74, 14–27. https://doi.org/10.1016/j.knosys.2014.10.016.

Guo, Q. et al. (2020). A survey on knowledge graph-based recommender systems. IEEE Transactions on Knowledge and Data 1–1. https://doi.org/10.1109/TKDE.2020.3028705.

Harper, F. M., & Konstan, J. A. (2015). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TIIS)*, 5(4), 1–19.

Hassanzadeh, O., & Consens, M. P. (2009). Linked movie data base. In *LDOW*.

Hussain, S. F., Mushtaq, M., & Halim, Z. (2014). Multi-view document clustering via ensemble method. *Journal of Intelligent Information System*, 43(1), 81–99.

Leake, D. B., & Wilson, D. C. (1998). Categorizing case-base maintenance: Dimensions and directions. In *Adv in case-based reason* (pp. 196-207). Berlin: Springer.

Lehmann, J. et al. (2015). Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semant web*, 6(2), 167–195.

Lin, Y. et al. (2015). Learning entity and relation embeddings for knowledge graph completion. In *9th AAAI Conf Artif Intell* pp 2181–2187.

Liu, Q., Cheng, G., & Qu, Y. (2020). Deeplens: Deep learning for entity summarization.

Liu, Y. et al. (2018). Graph summarization methods and applications: A survey. ACM Computing Surveys 51(3):1–34. https://doi.org/10.1145/3186727. article number 62.

Mesas, R. M., & Bellogín, A. (2020). Exploiting recommendation confidence in decision-aware recommender systems. *Journal of Intelligent Information System*, 54(1), 45–78.

Nakhjiri, N., Salamó, M., & Sànchez-marrè, M. (2020). Reputation-based maintenance in case-based reasoning. Knowledge-Based Systems 193(105283):1–11. https://doi.org/10.1016/j.knosys.2019.105283.

Nie, F., Cai, G., & Li, X. (2017). Multi-view clustering and semi-supervised classification with adaptive neighbours. In *Proc 35th AAAI conf artif intell,* pp 2408–2414.

Paun, I. (2020). Efficiency-effectiveness trade-offs in recommendation systems. In *14Th ACM conf recomm syst,* pp 770–775. *ACM, NY, USA.* https://doi.org/10.1145/3383313.3411452.

Peluffo-Ordóñez, D. H., Lee, J. A., & Verleysen, M. (2014). Recent methods for dimensionality reduction: a brief comparative analysis. In *22Th eur symp artif neural netw, ESANN 2014, bruges, Belgium, April 23-25, 2014*.

Piao, G., & Breslin, J. G. (2018). Transfer learning for item recommendations and knowledge graph completion in item related domains via a co-factorization model. In *Eur semant web conf,* pp 496–511. *Springer*.

Ragone, A. et al. (2017). Schema-summarization in linked-data-based feature selection for recommender systems. In *Proc symp appl comput, SAC '17,* pp 330–335. *ACM, NY, USA*. https://doi.org/10.1145/3019612.3019837.

Reddy, G. T. et al. (2020). Analysis of dimensionality reduction techniques on big data. *IEEE Access*, 8, 54,776–54,788. https://doi.org/10.1109/ACCESS.2020.2980942.

Rodríguez-García, M. A. et al. (2019). Blinddate recommender: a context-aware ontology-based dating recommendation platform. *Journal of Information Science*, 45(5), 573–591. https://doi.org/10.1177/0165551518806114.

Sacenti, J. A. P., Willrich, R., & Fileto, R. (2018). Hybrid recommender system based on multi-hierarchical ontologies. In *Proc 24th braz symp multimed and the web, webmedia 2018, Salvador, Brazil,* pp 149–156. *ACM*. https://doi.org/10.1145/3243082.3243106.

Shokeen, J., & Rana, C. (2020). Social recommender systems: techniques, domains, metrics, datasets and future scope. Journal of Intelligent Information System. 54:633–667. https://doi.org/10.1007/s10844-019-00578-5.

Smyth, B. (1998). Case-base maintenance. In *Tasks and methods in applied artif intell* (pp. 507–516). Berlin: Springer.

Smyth, B., & Keane, M. T. (1995). Remembering to forget. In *Proc 14th IJCAI. Citeseer*.

Sorzano, C. O. S., Vargas, J., & Montano, A.P. (2014). A survey of dimensionality reduction techniques.

Sun, Z. et al. (2018). Recurrent knowledge graph embedding for effective recommendation. In *Proc 12th ACM conf recomm syst, recsys '18,* pp 297–305, ACM*, NY, USA*. https://doi.org/10.1145/3240323.3240361.

Sydow, M., Pikuła, M., & Schenkel, R. (2013). The notion of diversity in graphical entity summarisation on semantic knowledge graphs. *Journal of Intelligent Information Systems*, 41(2), 109–149.

Trouillon, T. et al. (2016). Complex embeddings for simple link prediction. In *Proc 33rd int conf mach learn - volume 48, ICML'16,* pp 2071–2080*. JMLR.org*.

Van Der Maaten, L., Postma, E., & Van den Herik, J. (2009). Dimensionality reduction: a comparative review. *Journal of Machine Learning Research*, 10, 66–71.

Vlachos, M. et al. (2002). Non-linear dimensionality reduction techniques for classification and visualization. In *Proc 8th ACM SIGKDD int conf knowl discov and data min, KDD '02,* pp 645–651*, ACM, NY, USA*. https://doi.org/10.1145/775047.775143.

Wang, H. et al. (2018). Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proc 27th ACM int conf inf and knowl manag,* pp 417–426.

Wang, Q. et al. (2017). Knowledge graph embedding: a survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12), 2724–2743.

Wang, X. et al. (2019). KGAT: Knowledge graph attention network for recommendation. In *KDD,* pp 950–958.

Wang, Z. et al. (2014). Knowledge graph embedding by translating on hyperplanes. In *AAAI, vol 14,* pp 1112–1119*. Citeseer*.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometric Bulletin*, 1(6), 80–83.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-2(3), 408–421.

Wilson, D. R., & Martinez, T. R. (1997). Instance pruning techniques. In *Mach learn: Proc 14th int conf ICML'97,* pp. 404–411*. Morgan Kaufmann*.

Wu, J. et al. (2015). Trust-aware media recommendation in heterogeneous social networks. *World Wide Web*, 18(1), 139–157. https://doi.org/10.1007/s11280-013-0243-3.

Xue, Z. et al. (2015). Gomes: a group-aware multi-view fusion approach towards real-world image clustering. In *2015 IEEE Int conf multimed and expo (ICME),* pp 1–6*. IEEE*.

Yang, B., et al. (2015). Embedding entities and relations for learning and inference in knowledge bases.

Yang, Y., & Wang, H. (2018). Multi-view clustering: a survey. *Big Data Mining and Analytics*, 1(2), 83–107. https://doi.org/10.26599/BDMA.2018.9020003.

Yu, H. et al. (2018). Tag recommendation method in folksonomy based on user tagging status. *Journal of Intelligent Information System*, 50(3), 479–500.

Yu, H. et al. (2018). Web items recommendation based on multi-view clustering. In *2018 IEEE 42Nd annual comput softw and appl conf (COMPSAC), vol 01,* pp 420–425. https://doi.org/10.1109/COMPSAC.2018.00064.

Yu, X. et al. (2013). Recommendation in heterogeneous information networks with implicit user feedback. In *Proc 7th ACM conf recomm syst, recsys '13,* pp 347–350*, ACM, NY, USA*. https://doi.org/10.1145/2507157.2507230.

Zhang, F. et al. (2016). Collaborative knowledge base embedding for recomm systems. In *Proc 22nd ACM SIGKDD int conf knowl discov and data min,* pp 353–362.

Zhang, N., Tian, Y., & Patel, J.M. (2010). Discovery-driven graph summarization. In *2010 IEEE 26Th int conf data eng (ICDE 2010),* pp 880–891*. IEEE*.

Zhang, Y., et al. (2018). Learning over knowledge-base embeddings for recommendation. In *SIGIR*.

Zheng, X. et al. (2018). A tourism destination recommender system using users' sentiment and temporal dynamics. *Journal of Intelligent Information System*, 51(3), 557–578.