Semantically-enhanced topic recommendation systems for software projects

Izadi, Maliheh; Nejati, Mahtab; Heydarnoori, Abbas

DOI
10.1007/s10664-022-10272-w

Publication date
2023

Document Version
Final published version

Published in
Empirical Software Engineering

Empirical Software Engineering (2023) 28:50
https://doi.org/10.1007/s10664-022-10272-w

Semantically-enhanced topic recommendation systems for software projects

Maliheh Izadi[1] · **Mahtab Nejati**[2] · **Abbas Heydarnoori**[3]

Accepted: 30 November 2022

Abstract

Software-related platforms such as GitHub and Stack Overflow, have enabled their users to collaboratively label software entities with a form of metadata called topics. Tagging software repositories with relevant topics can be exploited for facilitating various downstream tasks. For instance, a correct and complete set of topics assigned to a repository can increase its visibility. Consequently, this improves the outcome of tasks such as browsing, searching, navigation, and organization of repositories. Unfortunately, assigned topics are usually highly noisy, and some repositories do not have well-assigned topics. Thus, there have been efforts on recommending topics for software projects, however, the semantic relationships among these topics have not been exploited so far.

In this work, we propose two recommender models for tagging software projects that incorporate the semantic relationship among topics. Our approach has two main phases; (1) we first take a collaborative approach to curate a dataset of quality topics specifically for the domain of software engineering and development. We also enrich this data with the semantic relationships among these topics and encapsulate them in a knowledge graph we call *SED-KGraph*. Then, (2) we build two recommender systems; The first one operates only based on the list of original topics assigned to a repository and the relationships specified in our knowledge graph. The second predictive model, however, assumes there are no topics available for a repository, hence it proceeds to predict the relevant topics based on both textual information of a software project (such as its README file), and SED-KGraph.

Communicated by: Sousuke Amasaki, Xin Xia, Shane McIntosh

This article belongs to the Topical Collection: *Predictive Models and Data Analytics in Software Engineering (PROMISE)*

✉ Maliheh Izadi
m.izadi@tudelft.nl

✉ Abbas Heydarnoori
aheydar@bgsu.edu

Mahtab Nejati
mahtab.nejati@uwaterloo.ca

[1] TU Delft, Delft, Netherlands

[2] University of Waterloo, Waterloo, Ontario, Canada

[3] Bowling Green State University, Bowling Green, Ohio, USA

Published online: 24 February 2023

We built SED-KGraph in a crowd-sourced project with 170 contributors from both academia and industry. Through their contributions, we constructed SED-KGraph with 2,234 carefully evaluated relationships among 863 community-curated topics. Regarding the recommenders' performance, the experiment results indicate that our solutions outperform baselines that neglect the semantic relationships among topics by at least 25% and 23% in terms of Average Success Rate and Mean Average Precision metrics, respectively. We share SED-KGraph, as a rich form of knowledge for the community to re-use and build upon. We also release the source code of our two recommender models, KGRec and KGRec+ (https://github.com/mahtab-nejati/KGRec).

Keywords Recommender system · Topics · Tags · Semantic relationships · Knowledge graph · Software projects · GitHub

1 Introduction

Software engineers and developers explore Software Information Sites such as GitHub and Stack Overflow to find interesting software components tailored to their needs, to reuse source code, to find answers to their programming questions and many more. However, the sheer number of software entities (projects, questions, etc.) hosted on these sites hinders efficient searching, retrieving, navigating, and categorizing said entities. For instance, GitHub currently hosts more than 240 million software projects.[1] Many of these projects share common characteristics such as similar objectives and functionalities. With the continuous growth of these platforms, more advanced automatic solutions are needed to improve the retrieval of relevant software projects. Existing techniques for better organization, documentation or and retrieval of software entities include various types of recommender systems (Xia et al. 2013; Xin-Yu Wang and Xia 2015; Vargas-Baldrich et al. 2015; Zhou et al. 2017; Wang et al. 2018; Liu et al. 2018; Izadi et al. 2021; Izadi et al. 2022; Mazrae et al. 2021), similar repository retrieval (McMillan et al. 2012; Thung et al. 2012; Zhang et al. 2017), and project clustering (Escobar-Avila et al. 2015; Zhang et al. 2019; Reyes et al. 2016; Yang et al. 2016). Topics, also known as tags, are a form of concise yet highly valuable metadata, that enrich software entities with human knowledge. Topics annotate an entity based on its core concepts. A software topic encompasses the key features of a repository including which category it belongs to, its main programming language, its intended audience, its user interface, and more. Topics complement textual descriptions of repositories as they highlight their main aspects with explicit and short tokens. Thus, topics can greatly help with the visibility of relevant entities to user queries. Consequently, they are used for improving the organization and retrieval of software repositories.

Topics can convey information in two ways; *explicitly* as stand-alone sources of information, and *implicitly* through their semantic connections to one other. The former has been extensively exploited to build topic recommendation systems (Xia et al. 2013; Di Sipio et al. 2020; Di Rocco et al. 2020; Izadi et al. 2021). For the latter, consider the topic `angular` which refers to an open-source web application framework.[2] As Angular provides functionality for front-end development, a programmer can almost immediately relate this topic to the `frontend` or `web-development` topic. Thus, there exist implicit links between

[1] January 2022, https://github.com/search
[2] https://angular.io/

topics `angular`, `frontend`, and `web-development`. In practice, repository owners -probably due to a lack of motivation- neglect tagging their projects with sufficient topics. Implicit connections mentioned above, can be utilized to track missing information, complement such incomplete topic sets, and consequently, improve the visibility of a given repository. They can also help recommendation systems suggest more accurate topic lists. However, the semantic relationships among software engineering topics and their impact on the performance of such predictive models are not properly explored yet. In this study, we strive to build more advanced recommendation systems for predicting key topics of software repositories through exploiting these relationships.

As much as topics and the relationships among them appear enticing for improving automated information retrieval-based tasks, there are several challenges for employing them in real-world scenarios, including the well-known *tag explosion* phenomenon (Golder and Huberman 2006) and the problem we call *tangled topics*. As users are free to define topics in the free-format text, they can create differently-written yet synonymous topics for any given concept, as well as compound and personal topics. This freedom results in an explosion of tags. That is when the set of topics exceedingly grows in number to the point that the sheer multitude of topics defeats their intended purposes. Furthermore, inspecting topics assigned by users, we came upon many tangled topics. These are compound topics that bundle multiple atomic concepts into a single tag and treat this tag as a distinct concept. Note that compound topics which communicate an atomic concept do exist, e.g., `single-page-application` is an atomic topic which should not be further dissected into multiple topics. However, a compound topic such as `java-library` can easily be broken down into its constituent atomic concepts without losing any semantic content. The same can happen by adding adjectives to the existing atomic topics, such as `small-library` or `big-library` and the situation exacerbates when `small-java-library` is also considered a unique concept. Unfortunately, some models redundantly recommend such topics together for a given repository. We believe that learning and recommending tangled topics adds little value when an entity is already assigned their atomic constituent topics while increasing the size and complexity of the topic set. As a result, to build an enhanced recommender system, we need to address both tag explosion and tangled topics problems through carefully assessing the input set of topics. In an attempt to resolve the above challenges and to collect a set of quality topics, GitHub has commenced a crowd-sourced project to feature a set of community-curated Software Engineering and Development (SED) topics.[3] At the time of commencing this study, this project curated 389 topics over the course of almost three years. This set of GitHub's featured topics contains valuable explicit information, however, semantic relationships among topics are missing. In this study, we take this seed as an initial set for topic collection and build upon it by acquiring more high-quality topics, and annotating them with semantic information based on human knowledge.

The next challenge is to properly store the high-quality SED topics along with their semantic connections. Knowledge graphs (KG) are a viable solution to this problem. More specifically, such relationships can be modeled in the form of relation triples ⟨ *subject, verb-phrase, object* ⟩. Take the previous example, we can store two types of relations as ⟨`angular`, is-a, `framework`⟩, and ⟨`angular`, provides-functionality, `frontend`⟩. KGs have been shown useful in different tasks such as information retrieval, recommendation, question answering, and search results ranking (Zou 2020). As a prominent example,

[3] https://github.com/github/explore

Google's KG is used to enhance its search engine results. They have also been widely used in domain-specific applications for medical, financial, news, social networks, etc. purposes (Zou 2020) as well as software engineering (Li et al. 2018; Chen et al. 2019). A KG of SED topics can improve the performance of topic-dependent tasks based on the topics assigned to the entities. In addition, such a KG can also be utilized as a structured knowledge base for the community to query, navigate, and perform an exploratory search. Hence, we aim to store the semantic information along with our topics in a KG, which we then feed into our predictive model to recommend better topics.

A domain-specific KG can be built automatically through processing domain knowledge, manually with the help of domain experts, or in a hybrid manner. In the software engineering domain, there exist a few semi- or fully-automatic approaches to build KGs (Zhao et al. 2017; Li et al. 2018; Chen et al. 2019; Sun et al. 2019; Sun et al. 2021). Zhao et al. propose *HDSKG* using a semi-automatic approach to construct a KG of SED topics (Zhao et al. 2017). Although HDSKG aims to minimize the manual effort that goes into the construction of a KG, it obsessively chunks noun phrases. This leads to the introduction of numerous tangled topics in this KG. Moreover, the knowledge scope acquired using fully-automatic approaches tends to be restricted to specific aspects/technologies such that the concepts can be predetermined or easily extracted. Construction of a KG of SED topics at the scope of this study is much more challenging due to the diversity of topics, their types of relationships, and the different abstraction levels of topics. Not to mention the data sparsity on particular topics, and data scatteredness across the web and multiple sources which cause duplicate, incomplete, and incorrect data (Fathalla and Lange 2018). Consequently, we take a mostly manual approach in conjunction with automation techniques for facilitating knowledge acquisition and evaluation.

In the first phase of our approach, using the contributions of 170 SE experts from both academia and industry, we acquire high-quality SED topics, extract their semantic relationships, evaluate this information, and store them in a domain-specific KG we call *SED-KGraph*. We developed an online platform on which we partially automated the growth of SED-KGraph using the help of our contributors in multiple snapshots. We expand the set of GitHub's featured topics to a more comprehensive and inclusive one. To guarantee the consistency of SED-KGraph, we centrally coordinate the expansion of this KG. By capturing the semantic relationships among topics in a KG, utilizing SED-KGraph can potentially improve the performance of solutions to numerous software community problems such as software entity classification, automated labeling, navigation, search, etc. While our approach to KG construction is a manual one, similar to `FreeBase` (Bollacker et al. 2008) which has been utilized in automated tasks in other studies (Dong et al. 2015; Xu et al. 2016; Yao and B. Van Durme. 2014), SED-KGraph too can pave the way for numerous automated topic-dependent tasks, while it continues to grow further over time.

In the second phase, we propose recommendation systems for two scenarios; (1) *KGRec*, a topic recommender system to predict *missing topics*, the topics relevant to the entities but not assigned to them by users. Correctly predicting the missing topics improves the completeness of the set of topics assigned to each project, which has been shown as an important factor in performance of solutions to topic-dependent tasks (Held et al. 2012). We build KGRec purely based on SED-KGraph through the application of spreading activation techniques. (2) Next, we build upon KGRec by adding a Machine Learning-based (ML) component to the model and proposing *KGRec+*, a fully automated topic prediction model. KGRec+ works based on both the projects' textual data and the knowledge captured in SED-KGraph.

We demonstrate that the recommender systems based on KGRec outperform the ones based on *TopFilter*, the state-of-the-art technique for relevant topic prediction (Di Rocco et al. 2020), especially when the set of initial topics assigned to the project is limited in number. Our contributions are as follows:

- We develop and evaluate two topic recommenders that outperform the competing approaches by 23% to 151% in terms of Mean Average Precision (MAP) score.
- We collaboratively augment the set of GitHub featured topics with 393 community-suggested topics. Furthermore, we present SED-KGraph to capture the semantic relationships among atomic and semantically unique SED topics to improve the performance of topic recommenders. We engage 170 practitioners and researchers from 16 technology-based companies and 11 universities in the expansion and validation of SED-KGraph. The resultant KG consists of 863 topics, 2,234 verified relationships, and 13 relation types.
- We publicly share our two main software artifacts; the data component (SED-KGraph), and the model component (KGRec, KGRec+) along with their source code for use by the SE community.[4]

In the following, we first define the problem formally. Next, we present the approach in Section 3, experiments' settings in Section 4, and our results in Section 5. We then discuss the results, lessons learned, and possible applications and implications of this work. Finally, we discuss the threats to the validity of this study and review the related work around the study.

2 Problem definition

GitHub hosts millions of repositories $S = \{r_1, r_2, .., r_n\}$, where r_i denotes a single software project. Each repository may contain various types of information such as a description, README files, wiki pages, and source code files. Each project may include a set of topics $T = \{t_1, t_2, ..., t_m\}$, where m is the number of assigned topics to a repository. Our goal is to (1) augment the initial set of topics assigned to a given repository r_i, or (2) recommend a set of topics from scratch for a given topic-less repository r_j. In both cases, we aim to enhance recommender models using semantic relationships among high-quality topics.

3 Approach

Our approach consists of two main phases; (1) acquire and store high-quality topics and the semantic relationships among them, (2) build stronger recommenders exploiting the semantic source of information. In the first phase, we exploit explicit human knowledge in the domain to procure rich input data for our topic prediction models. In the next phase, we propose two recommenders; *KGRec* is a topic *augmentation* model which is used when an initial set of topics is already assigned to a given project which we aim to extend, i.e., predict the missing topics only based on the original set. Finally, we stack this model on top of a ML-based component, building *KGRec+*, an automated *topic set* recommender system. KGRec+ eliminates the need for the initial set of topics and takes the textual data on the

[4] https://github.com/mahtab-nejati/KGRec

project and SED-KGraph as input. Figure 1 depicts the overall workflow of our approach. In the following, we provide more details on the proposed approach.

3.1 Phase 1: KG construction

As part of our approach for building better recommender systems, we need to utilize high quality software engineering topics and their semantic relationships. KGs are a viable solution to store such information in a structured format. In this section, we lay out the methodology through which we construct and evaluate such KG. We utilize a crowd-sourcing technique to build the SED-KGraph in a two-step process with the second step being an on-going and continuous expansion phase. To this end, we design an online platform for SED-KGraph's growth through community contributions. Throughout the process, individuals are involved in one of the two roles of *maintainer* or *contributor*. The first two authors take on the role of maintainers and the participants of the second step are the contributors. Figure 2 demonstrates the overall process of KG construction.

3.1.1 Initialization

Maintainers initialize SED-KGraph with a set of topics, relation types, and relationships in a *manual coding* process (Wagner and Fernández 2015). They incorporate the *triangular validation* (Wagner and Fernández 2015) process in which coders cross-evaluate the coding results. To initialize SED-KGraph, maintainers studied the 389 topics featured provided by GitHub and used it as the seed set. For each topic, maintainers studied the information available on GitHub about it, as well as the top projects on GitHub (based on the number of stargazers) labeled with it. Moreover, maintainers also searched each topic online to glean more knowledge on them. They referenced these projects to make sure their understanding of the topic matched with the usage of the topic in the community.

After acquiring an overall insight into the topics, maintainers defined the relationships among them in a manual coding process. They first discussed the possible types of relationships among the topics and decided on four basic, yet strongly effective ones as a primary set of relation types. Note that this decision was made with the *conciseness* feature in mind,

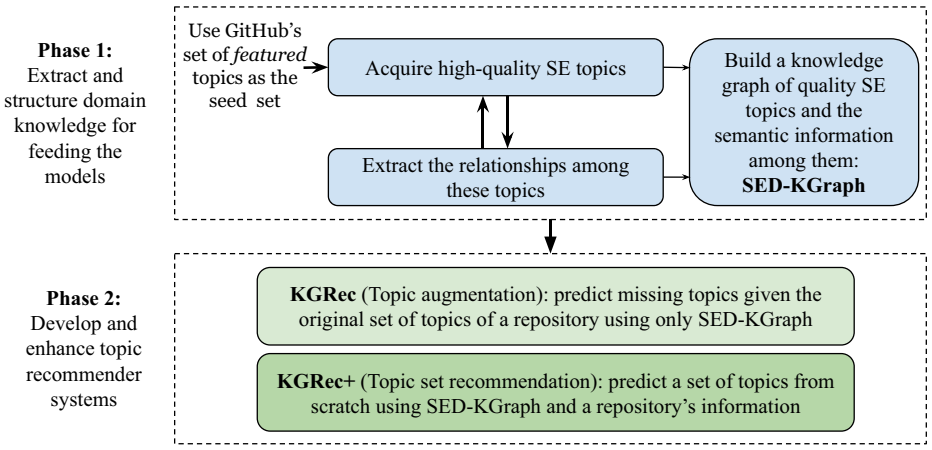


Fig. 1 Overall workflow of our proposed approach

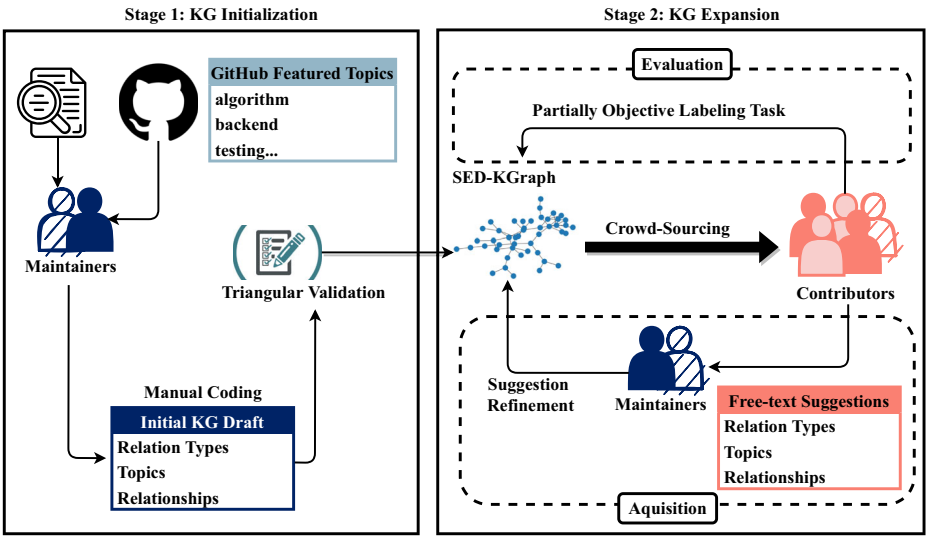


Fig. 2 KG construction

knowing that the primary set of relation types is not comprehensive. To minimize the impact of the maintainers' subjectiveness on the resultant KG, they take note to keep the initial structure minimal. by making the initial draft as concise as possible. Yet, the maintainers look to include as many distinct atomic topics as possible to cover the diverse range of SED topics. Then, they defined the relationships in an iterative manual coding process. In this process, each maintainer iterated over the set of topics several times, each time defining and/or correcting the relationships. Maintainers also incorporated triangular validation into the process to validate the relationships. They reviewed the relationships defined by each other to validate their correctness and effectiveness. In case of a disagreement, maintainers discussed their reasons for approving/disapproving of a relationship and made the final decision on the relationship's correctness together. If a consensus was not made, they included the relationships of conflict in the initial KG to allow the contributors to deliver the final verdict on the correctness of these relationships as a third jury.

3.1.2 Expansion

After establishing the first draft of SED-KGraph, the defined relationships needed to be evaluated. Furthermore, acquiring the community's knowledge on the topics and the relationships among them would help expand SED-KGraph into a more comprehensive KG. During the Expansion step, two separate tasks run simultaneously: *a) Evaluation* of the previously defined relationships, and *b) Acquisition* of new community knowledge on topics, relation types, and relationships among the topics. To facilitate this step, we deployed an online platform through which contributors engage in evaluation and expansion of the KG. Contributors review the relationships among topics and submit their suggestions in free-text forms. In the online platform, contributors are allowed to skip unfamiliar topics and choose topics of their interest to contribute to. This measure is taken to avoid mandating contributions when contributors are not familiar with a concept.

Finally, for this KG to be valid at all times, a *continuous expansion* method is required to maintain and update it with new knowledge. This is because the SED fields are dynamic, i.e., new topics frequently emerge and/or evolve such that the previously defined relationships can be affected. Moreover, we can not obviate the probability of unnoticed topics, i.e., topics that are insignificant or unknown to the contributors at the moment but grow into popularity over time. Therefore, the platform later evolved for maintenance purposes.

Evaluation In the KG Expansion step, contributors validate the correctness of the already defined relationships in SED-KGraph. They evaluate all the relationships defined during the Initialization step and Acquisition task. We achieve this through a *partially objective labeling task* (Alonso et al. 2014), a type of crowd-sourced labeling task in which the label of the subject to the task is determined based on inter-rater agreement. That is, the subject $\langle relationship \rangle$ is assigned the label ("True" or "False") which the majority of raters (contributors) have given the subject. Contributors validate the correctness of each relationship by labeling it with "True" and disapprove of the relationship by labeling it with "False". In the end, we consider the relationships as "approved" and add them to SED-KGraph only if the majority of the contributors who reviewed the relationship have labeled it with "True". Otherwise, the relationship is disposed of. Therefore, to determine the objective label for each relationship, we need votes from at least three contributors.

Acquisition The expansion of SED-KGraph solely depends on community contributions. To expand SED-KGraph such that it covers the currently missing SED topics and the relationships among them, we ask the contributors to provide relationship suggestions for each of the topics, they review through free-text forms. We also allow for new topic and relation type definitions As contributors submit their suggestions in the free-format text, there is a chance for tag explosion and/or tangled topics. Moreover, the semantic uniqueness of the relation types and topics is also at risk, leading to redundancy/duplication. To mitigate such occurrences, we implement policies and functionalities, some of which are (described in Section 4.6) into the online platform over which SED-KGraph is maintained. These policies and functionalities are designed such that they give the contributors autonomy in expanding the KG while ensuring that the integrity and consistency of the KG are preserved. Through initial snapshots, maintainers inspected the suggestions to mitigate the possibility of tag explosion and tangled topics, as well as to ensure the semantic uniqueness of the relation types. Once a suggestion is made, it must be validated by at least three other contributors before it is integrated into the KG. The refined relationships and topics amassed through the Acquisition phase will be the ones subject to evaluation in the next snapshot. Later, we added automation measures and restriction policies into the evolved version of the online platform to minimize the manual work required of the maintainers.

3.2 Phase 2: automated topic recommendation

In this phase, we propose two topic prediction models for two different scenarios; topic augmentation and topic set recommendation. We build these recommenders using the semantic information obtained in the previous phase.

3.2.1 Topic augmentation

We first propose KGRec, a model that takes the topics already assigned to a software project as input and recommends *missing* but relevant topics. Note that the input to this

model is only the seed set topics and no extra information about the repository. Using the SED-KGraph we aim to expand this initial set. We apply a *spreading activation* technique (Crestani 1997) on SED-KGraph to depth one. Spreading activation techniques operate on semantic networks based on the Spreading Activation theory in semantic networks. Figure 3 displays an overview of the spreading activation effect. When a set of nodes are activated, in the graph spreading activation computes the activation score of other nodes.

We first annotate SED-KGraph with node weights computed based on the popularity of the topic in the community and the degree of the topic node in SED-KGraph. Our intuitions are; (1) if a topic is used frequently in the community, its a more useful and valuable topic, and (2) a topic with a higher degree in SED-KGraph is related to more topics, and consequently has a better chance of being relevant to more projects which might be assigned the related topic. The weights are calculated as

$$W_t = \alpha \times P_t + \beta \times D_t, \alpha + \beta = 1 \tag{1}$$

in which W_t denotes the weight of the node corresponding to topic t. P_t and D_t are defined as below as a measure of popularity and degree of the topic t, respectively

$$\begin{aligned} P_t &= \frac{\log(n_t + 1)}{\max\limits_{t_i \in T} \log(n_{t_i} + 1)} \\ D_t &= \frac{\log(d_t + 1)}{\max\limits_{t_i \in T} \log(d_{t_i} + 1)} \end{aligned} \tag{2}$$

where n_t is the number of projects in the platform labeled with topic t, d_t is the total degree of topic t in SED-KGraph, and T is the set of topics in SED-KGraph. Note that α and β are coefficients to scale the scores. Note that the results can vary for different values of α and β. For instance, for $\alpha > \beta$, the results will mostly rely on the set of most popular GitHub-featured topics. With $\alpha < \beta$, one can emphasize topics that are not widely used by the community yet. To account equally for both popularity and node degree, we set

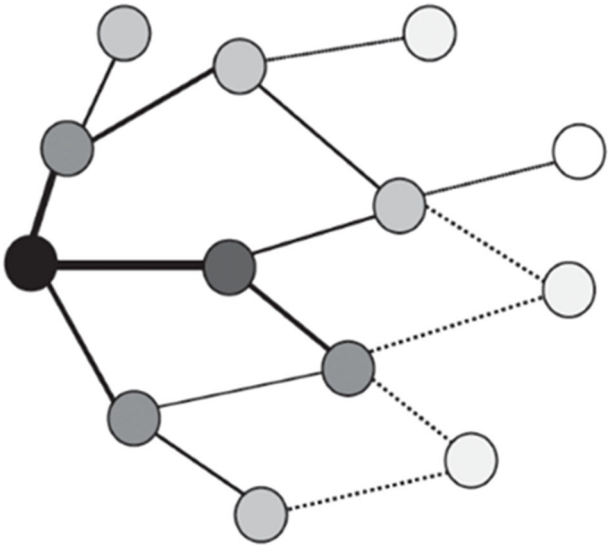


Fig. 3 Spreading Activation

$\alpha = \beta = 0.5$. Future researchers and practitioners can set these values according to their needs and use cases.

To generalize the formulation of the approach such that it can apply to the KGRec+ model as well, consider that topic t is relevant to project p with the probability Pr_t^p (here $Pr_t^p = 1$ since topics are already assigned to the projects). I is the set of initial topics assigned to project p and $N(I)$ denotes the set of topics that are immediate neighbors to at least one of the topics in I. For all t in $N(I)$, we spread this probability to compute the relevance score of topic t to project p along SED-KGraph edges using (3). Then, the model sorts the list of candidate topics by the S_t^ps and return the top-k ones as a list of recommendations.

$$S_t^p = W_t \times \sum_{t_k \in I} Pr_{t_k}^p; \forall t \in N(I) \tag{3}$$

Note that in our current model, we do not take the relation type into account when spreading the activation along the edges of the KG. Doing so requires extensive analysis of whether certain types of relations result in augmented topics that are a better fit for the repository. As the evaluation of the recommendations is a human-intensive task, verifying the effect of the relation types on the quality of the recommendation can only be achieved in the long run when the recommender system has been used enough times to provide adequate data for the analysis. We leave this as a future direction for this research.

3.2.2 Topic set recommendation

In this application, we assume that repositories are not labeled with any initial topic. We build upon KGRec and propose KGRec+ as a stand-alone topic set recommender model. We feed the model with the available textual data on the projects. This textual data include README files, repository description, and wiki pages. We concatenate and then transform these pieces of textual data to their respective TF-IDF vectors for consumption by the classifiers. Note that we use the *preprocessed* dataset provided by Izadi et al. (2021) in their recent work on the topic recommendation. [5] After the model is trained on repositories' textual data and assigned topics, it can predict a list of relevant topics for a given project. Finally, we take this set predicted by the ML-based component and complement it using KGRec.

As the ML-based component in our proposed model, we use two classifiers employed in our baselines. Based on Di Sipio et al.'s approach, we train a *Multinomial Naive Bayes* (MNB) text classifier that takes as input the textual data on the software project and predicts the Pr_t^ps for the GitHub featured topics. We also take Izadi et al.'s approach and train a multi-class multi-label *Logistic Regression* (LR) classifier which operates similarly to the baseline and yet significantly outperforms it Izadi et al. (2021). Note that we follow the instructions provided by the baselines to build these classifiers.

The ML-based component, trained with textual data of projects, predicts the Pr_t^p for each topic. We take the top-m topics with the highest Pr_t^p and feed them to the KGRec component as the set I. The KGRec component operates on this set of topics as input and augments the list of m topics with g more topics to return a list of $k = m + g$ topics as the recommendations. We propose LR+KGRec as our main approach and show that it outperforms all models in terms of all the considered metrics.

[5] https://github.com/MalihehIzadi/SoftwareTagRecommender

4 Experiment settings

In this section, we present our experimental setting. We first state the research questions of this study, then review the dataset, evaluation metrics, and model setting. Next, we provide an overview of our baselines, contributors' information engaged in the construction of SED-KGRaph, and the platform using which the graph was built.

4.1 Research questions

We aim to augment topic recommender models for software projects semantically using a high-quality KG tailored to SE topics. Thus, we answer the following research questions:

- **RQ1: What are the characteristics of our collaboratively-constructed KG, SED-KGraph?** First, as the dataset containing the KG is one of our main contributions, we describe the KG characteristics including its size, entities quantities or types, contributors' agreement rates during each snapshot, and more. As with any other dataset contributed to the research or industry community, we aim to report these characteristics to clarify details for the readers and future users of the KG. Moreover, as the KG is susceptible to expand over time, we would like to report our contribution at the time of conducting the research.
- **RQ2: How accurately can we augment a set of initial topics assigned to a repository utilizing SED-KGraph?** Missing topics recommendation is one of the main applications of SED-KGraph. Hence, with RQ2 we assess the accuracy of recommenders which solely augment missing topic sets using a seed of initial topics.
- **RQ3: How accurate is our topic set recommender, KGRec+, as a stand-alone predictive model?** Finally, we would like to build upon the previous application and propose a stand-alone recommender that takes a repository's textual information and predicts relevant topics. This is different from the previous question in which we only feed the model the initial topic seeds and not the repository's textual information.

With these three questions, we hope to clarify our contribution regarding the KG itself and also the two semantically-enhanced predictive models built upon such a graph.

4.2 Dataset

We use the dataset from Izadi et al.'s study (Izadi et al. 2021)[6], which contains cleaned textual data on about 152K projects, along with their cleaned topics. Our set of topics in the dataset contains 236 of GitHub's featured topics, eight more topics compared to the baseline (Izadi et al. 2021). This small difference (eight topics) comes from a mapping of synonymous topics in the dataset to their corresponding GitHub-featured ones based on the golden mapping provided by the baseline. This resulted in a strengthened number of samples for the few topics which qualified them for the training of the ML-based component. Note that all repositories have owner-assigned topics. In this dataset, all repositories have at least one topic, and on average, 2.46 topics are assigned to each repository by their owners. The textual data we use to train the models include repositories' descriptions, README files, and wiki pages. We do not use source file names and project names. In the provided preprocessed dataset (Izadi et al. 2021), about 12% of repositories have wiki pages. Considering

[6] https://github.com/MalihehIzadi/SoftwareTagRecommender

that all repositories in the data have at least one README file, we have ample data even for those that do not have wiki pages. Hence, we do not perform additional preprocessing tasks on the data. The baseline study provides more details on the statistics of data (Izadi et al. 2021). Similar to the common standard, we take 80% of the 152K projects as the training set for our ML-based component and leave the remaining 20% as the test set.

4.3 Evaluation metrics

We report the *Success Rate* (SR) of the relationships in the partially objective labeling task as a measure of the quality of the relationships defined by the maintainers during KG Initialization or suggested by the contributors during Acquisition. SR is the ratio of the relationships labeled "True" over the set of all the relationships under evaluation. The relationships labeled with "True" are considered successfully defined. Thus, with N_T as the number of relations labeled as "True" and N_F as the number of relations labeled as "False", the metric is defined as

$$SR = \frac{N_T}{(N_T + N_F)}. \tag{4}$$

We also report the *Absolute Agreement Ratio over True Relationships* (AARTR) as a measure of the quality of the community-approved relationships. A relationship is absolutely agreed upon if all the contributors label it with the same label ("True" or "False"). Respectively, a true relationship is absolutely agreed upon if all the contributors label it as "True". Considering N_T^{AA} as the number of absolutely agreed true relationships and N_T as the total number of true relationships, we define AARTR as

$$AARTR = \frac{N_T^{AA}}{N_T}. \tag{5}$$

Finally, to quantify the reliability of the contributors in the Evaluation step, we report *Average Rater-Objective Conformance Rate* (AROCR). *Rater-Objective Conformance Rate* (ROCR) is how the reliability of raters (contributors) is measured in a crowd-sourced partially objective labeling task (Alonso et al. 2014). Considering $N_{R_i}^C$ as the number of votes from R_i that conform with the final objective label of the items (relationships) and N_{R_i} as the total number of votes from R_i, the ROCR for rater R_i is calculated as

$$ROCR_{R_i} = \frac{N_{R_i}^C}{N_{R_i}}, \tag{6}$$

AROCR for the labeling task is defined as the average of $ROCR_{R_i}$ over all raters who contributed to the task. We select this metric since we engage a large number of contributors. Moreover, not all of our contributors review all the relationships, which makes metrics such as Cohen's Kappa an inadequate measure of reliability for our case.

Please note that Cohen's Kappa coefficient is a statistic that is used to measure inter-rater reliability for qualitative items. However, in this study, we have *multiple* and *varying* raters per relationship, hence, we cannot incorporate Cohen's Kappa. Moreover, in most cases, our raters share only a few relationships they have voted for with each of the other contributors. This incurs a huge amount of missing data and makes the results even more sensitive to human error in votes. As previous crowd-sourced studies have also exemplified (Zhang and Banovic 2021), even Krippendorf's alpha which accounts for missing data is not a well-suited metric for crowd-sourced studies with large numbers of participants. Hence, we used AARTR.

To evaluate the recommender systems, we run all approaches on the test set (20% of the projects in the dataset). However, since KGRec recommends "missing" topics, both as a stand-alone model and as a component of KGRec+, automatically calculated classification metrics such as Recall and Precision are irrelevant to the purpose of this study (Izadi and Ahmadabadi 2022). We support this statement by evaluating two baseline approaches (Di Sipio et al. 2020; Izadi et al. 2021) both automatically against the ground truth and manually through human evaluation, and list the results in Table 1. The results show that since the ground truth does not contain the missing topics, the automated evaluation of the recommendation lists does not serve justice to the merits of each model in recommending missing topics. Therefore, we randomly sample 50 projects from the test set and evaluate the results through a human evaluation process with five experts. Three evaluators validate the results for each project. We engage five Computer Science experts from both academia and industry in this human evaluation process. For each sampled project, we ask three evaluators to go through the content of the project and then determine whether the recommended topics were relevant to the project. To avoid biasing our evaluators towards any one of the approaches, we shuffle the results from different approaches before anonymously presenting them to the evaluators. KGRec is evaluated against TopFilter, and the ML+KGRec is evaluated against the ML+TopFilter as their goals are aligned. We also include evaluations on the ML-based components to investigate how KGRec can improve a ML-based solution.

As a measure of the practicality of the missing topic recommendation task, we report the percentage of the test cases in the test set and in the test sample set for which each of the approaches fails to recommend any topics. For this purpose, with N_{FC} as the number of test cases a model fails to return any recommendations for and N_C as the total number of test cases in the set, we define *Failed Case Ratio* (FCR) as

$$FCR = \frac{N_{FC}}{N_C}. \tag{7}$$

To quantify the quality of the recommendations, we report *Average Success Rate @k* (ASR@k) to evaluate the performance of the topic recommendation approaches. We also report *Mean Average Precision @k* (MAP@K) metric, a commonly-used measure for evaluating recommender systems that returns a ranked list of results. It captures how high successful suggestions are positioned in the recommendation list as well as the number of successful suggestions. Note that as FCR is already reported in the missing topic recommendation task's results, we only report the ASR and MAP over the set of test case samples for which the KGRec or TopFilter components do return a list of recommendations. However, reporting FCR for the automated topic recommendation task is irrelevant since the recommendation list is never empty.

Table 1 Automated versus manual evaluation

Model	ASR@5 Evaluation Method	
	Automated	Manual
Di Sipio et al. (2020)	24.60%	**30.80%**
Izadi et al. (2021)	30.00%	**50.80%**

4.4 Model settings

First, to calculate the topic weights (W_t), we get the number of public repositories labeled with each of the 863 topics in SED-KGraph through GitHub API calls. We then construct the weighted graph and implement KGRec in a tailored Python script. As we use the dataset from the baseline study (Izadi et al. 2021), the mean and maximum number of tokens in the concatenated input data is 235 and 650 tokens. Similar to the baseline paper, we set the maximum input length to 512 tokens and the maximum number of features to $20K$ for TF-IDF embedding vectors. The input data to the ML-based components is repositories' descriptions, README files, and wiki pages. We train the MNB (Di Sipio et al. 2020) and LR (Izadi et al. 2021) models from the python library, `scikit-learn`, for 236 of the GitHub featured topics, as these are the topics with enough supporting instances for training in the dataset. We set k to 5 as the length of recommendation lists. When evaluating the KGRec+ models, as the average number of topics assigned to the projects is 2.46, we set m to three. Then, feeding these m topics to the KGRec component, we take top-$f = 2$ topics from KGRec to make a full list of top-5 recommendations.

4.5 Baselines

Baselines for topic augmentation TopFilter, the state-of-the-art approach for augmenting a repository's topics list based on its initial set, is an item-based collaborative filtering approach (Di Rocco et al. 2020). In this approach, each project is represented with a set of assigned topics using a project-topic matrix. For each project, taking the set of topics assigned to the project, the model computes the similarity of this topic set with the topic sets of all other projects in the dataset and takes the topics assigned to the top-25 most similar projects (in terms of the set of topics assigned to the project) as the candidate set of topics. Calculating a ranking metric defined by the authors, the model returns the top-k topics as recommendations. While the authors do not evaluate TopFilter as a single component, we use this model as the baseline for our first task (topic augmentation).

Baselines for topic set recommendation For the second task, we use *MNB+TopFilter* proposed by Di Rocco et al. (2020) as one of the baselines. We also compare our method against Izadi et al. (2021) and Di Sipio et al. (2020) proposed methods for the topic recommendation. The third baseline uses an LR classifier based on our previous work. LR takes a repository's textual information including README files, description, and available wiki pages, concatenate them, and transforms them into TF-IDF vectors. Then the classifier is trained on the TF-IDF vectors. The labels for the classifier are the assigned topics. To provide a more comprehensive evaluation, we also stack our model (KGRec) on Di Sipio et al. (2020) proposed model. Moreover, we combine Di Rocco et al. (2020) and Izadi et al. (2021) approaches together and introduce it as another baseline.

4.6 Platform

We designed and deployed an online platform to collect contributions from the SE community and use their help in building and evaluating topics, relation types, and relationships (collectively called KG entities). First to construct the KG, we automatically retrieved GitHub's featured topics and presented them to contributors for knowledge acquisition in our platform. Contributors performed CRUD operations on KG entities independently.

Then, to minimize human effort and facilitate the KG Expansion step, we added more functionalities to the online platform. In the following, we describe our platform and it features with more details. Generally, after logging in, contributors can access and modify their previous contributions through their dashboard. [7] For snapshots of the platform please refer to the appendix.

Platform functionalities Inspired by similar crowed-sourced projects in code hosting and information websites such as GitHub and Stack overflow we provide a set of functionalities to help maintain the KG as it grows. These functionalities provide a level of autonomy for contributors while keeping the aforementioned challenges under check. They are used/performed by contributors, maintainers, or the platform itself. More specifically, users can contribute to the expansion and maintenance of the KG in several ways, including

- vote to approve or disapprove an already defined relationship,
- suggest new relationships for each of the topics they review through free-text forms,
- introduce new KG entity,
- edit their previously suggested KG entity,
- remove their own suggestion (before they are featured on the platform),
- request edit for existing KG entities so that SED-KGraph is modifiable upon evolution and emergence of topics,
- report spams, and finally
- report duplicates or aliases manually. Such reports are then brought to the entity owners' and maintainers' attention to address. This solution can also help with semantically-similar topics or relationships that are written with different lexicons.

The platform automatically runs the following actions.

- verify new entities based on our policies described in the following,
- run reliability check for contributors,
- check for possible aliases/redundancies. This is done automatically to check whether the topics that are being introduced by the contributors are already defined in the KG. We detect aliases and redundancies based on the topics' names, and alias lists. Specifically, we use the NLTK library's edit distance functionality.[8] Through this mechanism, the edit distance between all names (full name and display name) and aliases of each pair of topics are calculated. Thus, for a pair of topics t_1 and t_2 with m_1 and m_2 names (full name, display name, and aliases) respectively, $m_1 \times m_2$ similarities are computed. The pair is marked as potential redundancy if at least one of these similarities is above a certain threshold, here 80%. We choose this threshold to retrieve all potential duplicates and minimize the False Negative error. However, this value can be tuned. Once a topic is detected as potentially duplicate, it is listed along with the pair causing the duplicate for maintainers to check on them.

Finally, maintainers perform the below tasks to keep the KG intact. The platform automatically runs the following actions.

- random checks,
- verify edits,

[7] To access the platform, please refer to our public GitHub repository at https://github.com/mahtab-nejati/KGRec.

[8] https://tedboy.github.io/nlps/generated/generated/nltk.edit_distance.html

– check reported spams, and
– resolve reported redundancies. The potential redundant pairs identified either manually by contributors or automatically by the system are brought to the entity owners' and maintainers' attention to address.

Platform policies We also establish policies to guarantee contributors' eligibility for evaluation and expansion of SED-KGraph. The policies include tutorials before granting permission to perform CRUD operations, random checks on the suggestions and evaluations by maintainers, and reliability checks based on the conformance of contributors' answers with the objective labels of relationships. This helps identify issues in the KG and potentially detecting unreliable users. More specifically, we have different policies in place for user permissions and entity acceptance as explained in the following.

1. User Permission Policies

– Only users with at least three years of academic experience or one year of industrial experience and Computer Science-related fields shall contribute to the study. Once a user meets the minimum requirement, they are considered reliable to start contributing.
– We constantly check for the reliability of the users by comparing whether the majority of their votes conform with the majority agreement for each relationship they have voted for. If the portion of their conforming votes falls under a set threshold, here 50%, their reliability is revoked and all their previous votes are nullified. Note that this threshold can be configured to other values as well.
– We provide background information on topics (if any). For instance, several topics already have definitions in the GitHub featured set.
– Reliable contributors can up/down-vote the relationships. To vote for a relationship, contributors need to first read the definition of the verb in the relationship and mark it as read.
– Contributors are also allowed to skip assessing a relationship in case they do not have enough knowledge to evaluate it or simply prefer not to comment.
– Reliable contributors get creator-level permissions if they have marked all the verbs as read and voted for a total of 50 relationships involving 20 topics. Similarly, the parameter values here can also be tuned. As a creator, contributors can define new topics, verbs, and relationships.
– Take note that once the reliability is revoked, creator-level permissions are revoked too.
– Contributors can edit or delete the entities they have created unless the entity is accepted (featured in the platform).

2. Entity Acceptance Policies

– Relationships are accepted through a partially objective labeling task.
– In this task, contributors up-vote or down-vote each of the relationships. They can also declare that they do not know whether the relationship is correct, in which case the vote is considered a null vote and does not affect the acceptance criteria.
– For a relationship to be accepted, at least three non-null votes from the contributors are required.
– If all first three voters agree that a relationship is correct (absolute agreement, 100% acceptance), the relationship is accepted and featured. Otherwise, we gradually lower the threshold for acceptance down to 65% among at least nine contributors. We do not lower the threshold any further to make sure that suspicious relationships are not accepted. Note that we tried lowering it down to 50% and the graph did not change a

lot. However, we chose not to decrease the threshold to have higher confidence in the result.
- A topic/verb is accepted and featured if it is in at least one accepted relationship.
- GitHub-featured topics are also accepted by default as they have already been assessed by the GitHub community and the project's maintainers.

4.7 Contributors' overview

In this section, we provide an overview of contributors. To engage contributors and ensure diversity, we sent out invitations to the technical teams of 30 local technology-based companies active in a variety of SED-related fields including software engineering, cloud computing, data science, network, blockchain, security, social media, e-commerce, digital advertisement, entertainment, etc. We also invited students of related programs including Computer Engineering, Computer Science, Data Science, Software Engineering, IT, etc. from 20 top local and international universities. The invitation was open for a total of seven months, over which individuals could apply to participate in the study.

To improve reliability, we disqualified (1) students with less than three years of academic experience and no industrial experience, and (2) practitioners with less than three years of industrial experience who did not have at least three years of prior academic experience. Since the first six months of KG Expansion required central control over the suggestions, the maintainers thoroughly refined the suggestions every two months and issued refined suggestions for evaluation. Therefore, the first three snapshots of the SED-KGraph were captured. We engaged the first 50 applicants during the first, the next 40 applicants in the second, and the next 30 applicants in the third two-months period of KG Expansion, resulting in the three snapshots. Finally, the last 50 applicants were engaged in a long-term (six months) snapshot for expansion of the KG (fourth snapshot). Throughout the KG Expansion step, we eliminated and replaced unreliable contributors, i.e., contributors with *Rater-Objective Conformance Rate* (ROCR, defined in Section 3.1.2) lower than 50%. The reliability checker functionality of the online platform automatically applies this policy among others to assure the reliability of the contributors.

In the end, 170 individuals have made contributions to the study from 16 companies and 11 universities. The diversity of the contributors' experience and expertise matched our requirements, considering the wide range of topics in the KG, and allowed for a fair evaluation and expansion process. Table 2 presents more details, including the average years of experience in both industry and academia, on the contributors.

5 Results

In this section, we provided the results of our approach. We first review the characteristics of the constructed KG to answer RQ1. Then, we proceed to present the evaluation results of the two recommender models to address RQ2 and RQ3.

5.1 SED-KGraph: data characteristics

This section first summarizes the results of each step in the KG construction process, followed by the characteristics of SED-KGraph. Tables 3 and 4 summarize the results captured in each snapshot.

Table 2 Overview of contributors' information

Snapshot	Duration	BG	All	Gender		Experience (Academia, years)			Experience (Industry, years)		
				M	F	Avg	Min	Max	Avg	Min	Max
#1	2M	Academia	25	15	10	3.24	3	5	0	0	0
		Industry	25	21	4	3.92	3	10	1.92	1	9
		All	50	36	14	3.58	3	10	0.96	0	9
#2	2M	Academia	13	10	3	5.08	3	10	0	0	0
		Industry	27	13	14	7.81	0	16	5.85	1	10
		All	40	23	17	6.73	0	16	3.95	0	10
#3	2M	Academia	12	5	7	5.83	4	10	0	0	0
		Industry	18	17	1	7.94	4	15	4.56	2	14
		All	30	22	8	7.1	4	15	2.74	0	14
#4	6M	Academia	18	7	11	4.21	3	9	0	0	0
		Industry	32	15	17	5.46	2	8	3.11	1	6
		All	50	22	28	5.01	2	9	1.99	0	6

Initialization The seed set of topics from GitHub's project contained 389 topics from a wide variety of areas in SED, and from different levels of abstraction, i.e., topics could be as coarse-grained as `ai` or as fine-grained as `django`. However, this set proved to be relatively inadequate in representing the final goal of the study since many proper topics such as `web-development`, and `ui-ux` were missing. Moreover, such an occurrence is inevitable due to the constant emergence of new topics in the community. The maintainers identified this issue while defining the relationships and as a solution, augmented the seed set with 72 more topics in the process of constructing the first draft of SED-KGraph, enlarging the set to include 461 distinct topics. This was done with the aim to provide a more comprehensive set and to facilitate the contributors' task. Having conciseness in mind, maintainers defined four primary relation types, namely *is-a*, *is-used-in-field*,

Table 3 Statistics for different snapshots

Snapshot	Topics	Relationship Types	Verified Relationships
#1	461	4	0
#2	640	12	982
#3	716	13	1,548
#4	812	13	1,864
All	863	13	2,234

Table 4 Results of expansion and maintenance tasks on the graph

	Evaluation (Relationships)					Acquisition (Suggestions)		
Snapshot	TL	FL	SR	AARTR	AROCR	Relationship Types	Topics	Rels
#1	982	101	0.907	0.887	0.791	8	179	635
#2	566	69	0.891	0.744	0.726	1	76	322
#3	316	6	0.981	0.877	0.891	0	15	39
#4	370	41	0.900	0.754	0.780	0	51**	372**
All	2,234	217	0.911	0.828	0.789	9	321	1368

(*) TL and FL denote the number of True Labels and False Labels. (**) These topics and relationships are gradually added to and evaluated in the snapshot#4

provides-functionality, and *works-with* described and exemplified in Table 5.[9] The first three relation types capture three determinative characteristics of a topic regarding the topic's scope. The last relation type connects the most closely intertwined yet differently categorized topics together. In the end, the maintainers agreed on 995 relationships and disagreed over a total of 88 of them (8.13%). This yielded 1,083 relationships with the four primary types among the 389 featured topics and the 72 augmented topics in the initial draft of SED-KGraph.

Expansion Figure 4 illustrates a sample node and its relationships' evolution over the KG Expansion process in the SED-KGraph.

In the first snapshot, we evaluated the initial draft of SED-KGraph. From the 995 relationships in the initial KG, (excluding the 88 that the maintainers disagreed on) contributors labeled 963 as "True" but disapproved 32 relationships. Contributors also rejected 69 of relationships already disapproved by the maintainers and accepted only 19 of them. This yielded a success rate of 96.78% for the set of 995 relationships, 21.60% for the set of 88 relationships, and 90.67% in total. Among the 982 approved relationships, 88.68% were unanimously labeled as true relationships by the contributors and the AROCR was 79.12%. Contributors also contributed to the expansion of the KG by providing 838 new suggestions in total. Through refinement of these suggestions, the maintainers yielded 635 new and distinct relationships, introducing 179 topics and eight relation types that were not previously defined in SED-KGraph. This made the set of topics grow in size up to 640.

In the second snapshot, the 635 new relationships acquired during the first snapshot were subject to evaluation, from which 566 were approved and 69 relationships were deemed ineffective by the contributors. This yielded a success rate of 89.13% for the Evaluation step. The AARTR dropped to 74.38% and the AROCR was 72.56%. The reason for such a drop can be explained by the number of relation types and their granularity. The number of relation types reaches 12, which adds to the complexity of the KG and might confuse the contributors to some extent. In this snapshot, the contributors made a total of 642 relationship suggestions. Upon inspection, the maintainers identified 322 of these as distinct and new ones. The 322 new relationships introduced 76 new topics to the KG, enlarging

[9]For more samples please refer to Appendix B

Table 5 Relation types' descriptions and examples

Step	Relation Type	Description	Example
#1	*Is-a*	This is the most basic relation type that allows for the categorization of the topics of the same type together.	(`django`, *is-a*, `framework`)
	Is-used-in-field	Relationships of this type map the topic to the field or area it is used in and allow for categorization based on the application field.	(`django`, *is-used-in-field*, `web-development`)
	Provides-functionality	Relationships of this type map the topic to the functionality (i.e., the functional purpose of the topic) it provides and allow for categorization based on the functionality of topics.	(`django`, *provides-functionality*, `backend`)
	Works-with	Relations of this type map the topic to its dependencies or compatibility constraints. This relation is a bidirectional one, i.e., it matches the topics that work together.	(`django`, *works-with*, `python`)
#2	*Is-subset-of*	This type of relation allows for hierarchical categorization of topics, putting the subject topic under a broader concept (object topic).	(`deep-learning`, *is-subset-of*, `neural-network`)
	Is-based-on	Relationships of this type indicate that the creation or development of the subject topic was achieved through use of the object topic.	(`archlinux`, *is-based-on*, `linux`)
	Is-focused-on	Relationships of this type emphasize the concepts that the subject topic is concerned with.	(`agile`, *is-focused-on*, `flexibility`)
	Has-property	This type of relation connects the subject topic to meta-data topics. The meta-data topics only include well-known and widely used ones.	(`mysql`, *has-property*, `open-source`)
	Overlaps-with	This is a bidirectional relation that links two topics that share some common grounds but are not necessarily interdependent.	(`robotics`, *overlaps-with*, `ai`)
	Provides-product	This relation type connects the subject topic as a provider to the products it provides. The provider could be a company, a software system, a tool, or any other entity that creates and provides another entity as a product.	(`google`, *provides-product*, `flutter`)
	Provided-by	This is the inverse of the *"provides-product"* relation type and keeps the provider and the product connected when the provider is the topic of the user's interest.	(`atom`, *provided-by*, `github`)
	Maintained-by	Relationships of this type connect the subject topic to the authorities that maintain the subject topic.	(`html`, *maintained-by*, `w3c`)
#3	*Has-license*	This relation type maps connects the subject topic to its corresponding license.	(`backbonejs`, *has-license*, `mit-license`)

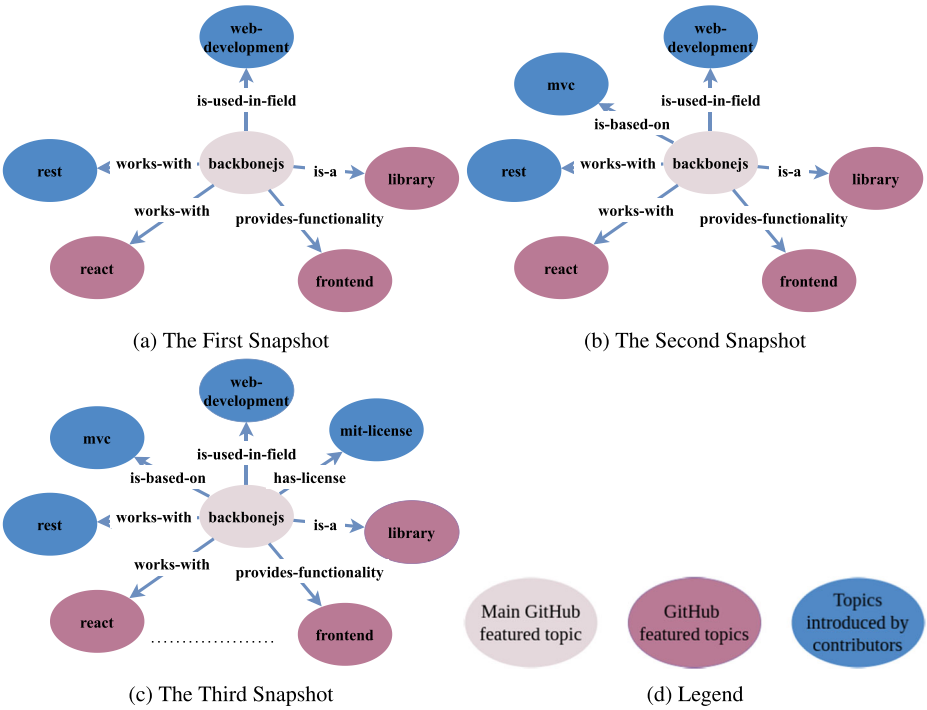


Fig. 4 Sample Node Expansion

the set of topics to 716 distinct ones. Moreover, one new relation type was introduced. Rejected *relation types* (verbs) were either too specific (e.g., included versioning) or they were synonymous to other verbs. The latter, rather than getting rejected, were merged. For instance, the two relation types `used-for` and `provides-functionality` were merged. Moreover, rejected *relationships* mainly were rejected due to the granularity of the object topic (and in some cases the subject topic). For example, if the object topic contained a version such as 3 in `python3`. In such cases, relationships sometimes became practically a duplication of another relationship and were rejected or merged.

In the third snapshot, we evaluated the new 322 relationships acquired in the previous snapshot from which only six were disapproved. Contributors verified the remaining 316 relationships, resulting in a success rate of 98.14%. As the KG became more stable and the number of new suggestions dropped, the AARTR grows back to an 87.66% and the AROCR is 89.02%. The contributors made a total of 53 relationship suggestions in this two-months period. After refinement, the maintainers acquired 39 new and distinct relationship suggestions. These suggestions introduced 15 new topics to the KG.

For the final snapshot, maintainers identified 17 more topics, and acquired 81 new topics added to GitHub's feature topic list during the past few months. Maintainers gradually injected these topics, without initializing their relationship set, into the KG. We asked contributors to define new relationships for these topics. As a result of this extra knowledge acquisition step, contributors defined 532 more relationships. Refinement of these suggested relationships resulted in a set of 372 relationships, introducing 51 more topics. These acquired relationships were also gradually injected into the KG and evaluated over the same

six months. Therefore, for the fourth snapshot, a total of 411 relationships (acquired over the previous step and during the fourth step) were evaluated. This resulted in 370 of the relationships under review getting accepted and 41 of them getting rejected, yielding a success rate of 90.02%. We terminated this long-term phase as the number of suggested relationships and topics by the contributors gradually diminished.

5.1.1 Resultant KG

SED-KGraph consists of 2,234 relationships of 13 relation types among 863 distinct software topics. Topics appear as both the subject and the object topic in relationships. The topic `web-development` has the maximum number of appearances (78 relationships), while the minimum number of appearances is one. While one might argue that such rare topics should be eliminated from the KG, they can be among the useful topics frequently used by the community. Examples of such topics are `awesome`, `authorization`, and `augmented-reality`, each assigned to 3,863, 1,847, and 1,628 projects on GitHub, making them well-known topics in the community. They are also evidently important topics in the SED domain. Not to mention the topics denoting programming languages that fall under the same circumstances are important topics when used as labels for software entities. Moreover, we believe that dropping such topics hinders the effective expansion of SED-KGraph. For SED-KGraph to remain valid and correct, it should be continuously expanded as new fields and technologies are always at emergence. Such rarely present topics might rise to popularity or be newly emergent ones that need to be well-established in the KG through future contributions. Thus, we address this issue by assigning weights to topics and relationships. Figure 5 presents the long-tail plot of the number of relationship appearances per topic, for the 25 most recurrent ones. Moreover, Table 6 details the number of relationships in the KG per type of relations. Notice how the four primary relation types, *is-a*, *is-used-in-field*, *provides-functionality*, and *works-with*, are the most common relationships in the KG.

5.2 KGRec: topic augmentation model

Table 7 summarizes the results from the missing topic recommendation task. As the FCR values indicate, TopFilter (Di Rocco et al. 2020) fails to make any recommendations for

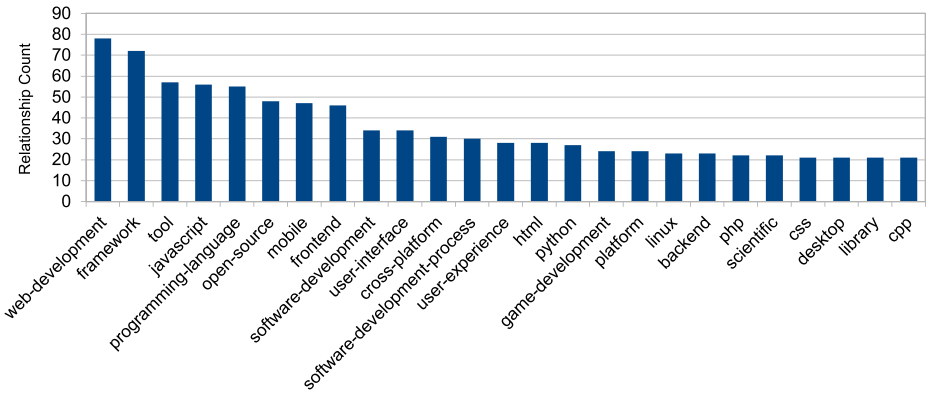


Fig. 5 Top 25 most frequent topics

Table 6 Relation type frequency

Relation Type	Count	Relation Type	Count
Has-license	30	*maintained-by*	6
Has-property	134	*overlaps-with*	7
Is-a	578	*provided-by*	25
Is-based-on	55	*provides-functionality*	429
Is-focused-on	43	*provides-product*	18
Is-subset-of	19	*works-with*	450
Is-used-in-field	440		

almost 50% of the test cases, no matter the correctness of the recommendations. The reason behind this is mainly the limited number of topics assigned to the projects in the dataset (2.46 topics on average which is closer to reality). Any collaborative filtering method suffers from the cold start problem (Wang et al. 2018). An average of 2.26 topics indicates the data sparsity, i.e., there are limited items (topics) assigned to projects, which in turn results in the cold start problem. This shortcoming of TopFilter is also pointed out as a limitation of the approach by Di Rocco et al. (2020). Taking into account that the dataset is captured from a real-world setting, this raises questions about the practicality of TopFilter. However, KGRec overcomes this limitation and manages to make recommendations under such circumstances. To better understand the magnitude of the improvements that KGRec brings forth, one must take the FCR into account. The ASR measure for this task is only calculated over the set of test cases for which the approach under evaluation has managed to make recommendations. That is, for TopFilter, the ASR is calculated over 54% of the test cases, while for KGRec, it is calculated over 98% of the test cases. Regardless, KGRec outperforms the baselines by +67% and +218% in terms of ASR and MAP respectively. That is, KGRec performs considerably better over a wider set of test cases, while the baseline has a high ratio of FCR.

5.3 KGRec+: topic set recommendation model

To further evaluate KGRec+, we improve the baseline approaches by combining them or stacking KGRec on top of the approach. We include the resultant models as modified baselines. Table 8 presents the automated topic set recommendation results. Notice that reporting FCR for this task is irrelevant since the ML-based components of the approaches always manage to make a list of recommendations, which compose part of the final recommendation lists. Therefore, FCR for each of the approaches values at zero in such circumstances.

Table 7 Topic augmentation results

Model	Over Test Set	Over Sampled Test Set		
	FCR	FCR	ASR@5	MAP@5
Di Rocco et al. (2020)	50.15%	46.00%	28.33%	10.31%
KGRec (proposed)	**2.06%**	**2.00%**	**47.32%**	**32.86%**

Table 8 Automated topic set recommendation results

	Model	ASR@5	MAP@5
Baselines	MNB (Di Sipio et al. 2020)	30.80%	27.07%
	TopFilter (Di Rocco et al. 2020)	41.20%	33.47%
	LR (Izadi et al. 2021)	50.80%	48.93%
Modified Baselines	KGRec plus MNB (Di Sipio et al. 2020)	54.80%	42.94%
	TopFilter (Di Rocco et al. 2020) plus LR (Izadi et al. 2021)	58.40%	55.05%
Proposed	KGRec+ (KGRec plus LR (Izadi et al. 2021))	**72.80%**	**67.87%**
	Outperforming baselines by	**+25% to +136%**	**+23% to +151%**

The results indicate that not only does KGRec+ outperforms all the previously proposed baselines by at least 43.31% and 38.71% in terms of ASR and MAP respectively, but it also yields better results than the modified and improved versions of the baseline approaches. To be exact, KGRec+ outperforms all the approaches, including the modified ones by at least 25% and 23% in terms of ASR and MAP, respectively. As mentioned before, LR outperforms MNB as the ML-based component. As we use LR in our approach and to understand the impact of KGRec as part of the proposed approach (KGRec+), one can compare the LR classifier performance with and without the KGRec component. According to this table, LR alone achieves 50.8% and 48.93% regarding ASR@5 and MAP@5, respectively. However, KGRec+ achieves 72.8% and 67.8% for ASR@5 and MAP@5, respectively. This outperformance (43% ASR@5 and 39% MAP@5) highlights the contribution of KGRec as part of KGRec+. We believe our unique advantage lies in the fact that current recommenders are restricted to the training data which suffers from missing topics. Machine learning techniques' performance is usually limited by the quality of the data they consume. However, using the knowledge graph built on human expertise, we provide the recommender model with missing blocks of information that it can exploit to complement and enhance the recommendation list. In other words, the combination helps us utilize both the strengths of the ML-based component and expert knowledge.

5.4 Discussion

In the following, we discuss various aspects of our approach and its settings, our results, lessons learned, and possible applications of this work.

KGRec+ parameter setting When recommending the final top k topics per repository, we take m topics from the ML-based component and g topics from the KGRec component, where $k = m+g$. We can use arbitrary values of m and g that fit the above criterion. Increasing m translates to taking more topics from the ML-based component. But one should also consider the number of available topics in the ground truth set on which the model is trained to avoid going beyond the characteristics/limitations of the classifier. Moreover, in a fixed-length recommendation list, increasing m leaves fewer places for missing topics to be predicted, hence hindering effective evaluation of the missing topic recommender

component. On the other hand, decreasing m causes the approach to rely on fewer topics discovered by the ML-based component. Note that providing very few initial topics may cause the missing topic recommender component to struggle for finding sufficient relevant topics. Hence, all this should be taken into account while tuning these parameters' values in different use cases. For our application, as the average number of topics per repository is 2.4, and the model is trained on that data, we choose $m = 3$ to be fair to the ML-based component. Then, as we aim to complement the predicted topic set, we set $g = 2$ to construct a set of 5 topics per repository. This setting considering the dataset characteristics seemed more logical. Regarding the total size of the recommended list, we also experimented with other settings. As expected, as the number of predicted tags goes up, the recall score increases and precision decreases. This means while we are becoming more confident that the ground truth tags are being retrieved by the recommender, more unrelated or missed tags can be also added to the recommendation list. This highlights the need for the manual inspection of these missing topics to determine whether they are relevant to the repository or not. Hence, this number needs to be customized based on the dataset and the problem domain at hand in other use cases.

Tangled topics In this study, we mainly focused on constructing the KG using the help of experts to avoid tangled topics and tag explosion. Hence, we set a number of policies in our platform to prevent tangled topics polluting the data as much as possible. More specifically, in our platform, we have redundancy detection, spam report, and edit suggestion capabilities to mitigate such risks. Please refer to Section 4.6 for the platform functionalities. Redundancy detection can potentially detect tangled topics. When someone defines a topic, they can see the list of the redundancies with that new topic and either delete their topic to resolve the redundancy or edit it to clarify the differences if needed. Maintainers can also check reported redundancies and resolve them. The same goes for the tag explosion problem. Finally, we would like to point out that there are more advanced features to detect semantically-similar topics such as ML techniques using contextual embeddings. This is indeed a possible future feature for our platform.

Lessons learned Through our experiments, we learned a few lessons; some helped us better understand and adjust our process, and some are worthy of further investigation by future studies. Next, we will review them. Based on our experience in this study, we realized that topic sets can easily enlarge and become irrelevant or useless for practical downstream tasks. Hence, it was evident to us that to capture a high-quality set of topics, manual inspection by experts is highly recommended. However, one should also mind the cost. Here, semi-automatic features can be helpful. Another clear lesson was the fact that the KG should be maintained and updated as the SE community grows and expands. Hence, we improved our platform to include several automatic and semi-automatic functionalities to help maintain the KG. Moreover, we observed that SE practitioners seem to be more familiar with SE topics and how they work or relate to each other compared to SE researchers without any industrial background. This may be due to the fact that practitioners interact with these technologies on a daily basis and *may* be more up-to-date. This notion can indicate that topics in our KG may be more practical in nature. This is of course not verified and a controlled experiment and more qualitative studies are required to confirm such assumptions. Based on the assessments, we also suggest that contributors limit their contributions to topics related to their specialty rather than trying to contribute to the whole graph. We emphasize this notion to our contributors on the platform. An interesting remark is that some concepts or fields may be lacking initial seed topics to begin with. For instance, a popular concept such

as software security does not have well-curated seed topics in GitHub's featured set yet. This makes it more difficult for contributors to expand the KG for this specific field as they rarely come across topics hinting at security. Subsequently, recommenders based on this may under-recommend such topics. This is an existing challenge for such applications and is an interesting line of research to pursue. Finally, we observed that as the topic set matures and becomes more stable over time, the growth in the average performance of recommenders over all topics slows down gradually. This can indicate that while the KG helps build better recommenders (compared to recommenders that do not utilize the KG), individual repositories associated with newer or rarely-used topics may benefit more from the growth of the KG over time.

During the KG construction and expansion phases, we encountered a few controversial aspects of topics/relationships including: A main controversial issue was to whether use abbreviated or long forms of topics. For instance, we can use `SE` instead of `Software Engineering`. Currently, our policy is to include both under the display and full names for a topic to avoid confusion or redundancy. Moreover, some topics have different meanings. This is exasperated in the case of abbreviated topics. In these cases, one can refer to tagged repositories to determine the difference. Users who try to introduce such topics will be notified of duplicates. Moreover, topics that are a single topic in nature may seem to be compound at first look. For example, `material-design-for-bootstrap` sounds compound but it is a single topic. Another controversy may arise from the experience or familiarity level of users with different SE fields. Consider the topic `less` which stands for Leaner Style Sheets in the SE domain and refers to a dynamic preprocessor style sheet language that can be compiled into CSS. However, an inexperienced user may confuse it with the "less" determiner in the English language. Finally, bidirectional relationships such as `works-with` may create duplicate pairs. Nonetheless, the multiple-rater policy we have in place on the platform helps avoid mistakes.

Implications and potential applications We curated SED-KGraph to help build better topic recommenders for software repositories. Through experiments, we demonstrated that utilizing SED-KGraph in the topic recommendation task improves the quality of recommendation lists. Topic sets assigned to repositories can be enlarged twice their original size using only cleaned and high-quality topics. Moreover, the accuracy of complemented topic sets can be increased up to 151% which is a notable improvement. Moreover, it has been shown that the correctness and completeness of the set of topics assigned to entities improve their visibility, and in turn, impact the performance of any topic-based solution to information retrieval problems (Held et al. 2012). Hence, SED-KGraph can also be beneficial in other settings. Our work paves the way for future research with many possible directions to improve both industrial and academic problems. Practitioners and researchers can tailor our KG to other applications including (but not limited to) categorizing SE entities, enhancing retrieval and searching, improving exploratory navigation in information websites, recommending similar repositories, discovering duplicate QA posts on Stack Overflow, and many more. Furthermore, SED-KGraph, as a structured knowledge base, can serve as a rich source of information on SED topics themselves. In SED-KGraph, topics are stored in the form of info boxes, i.e., topics are saved along with a list of their aliases, links to the informative pages on the topics, and short descriptions of the topics. Therefore, information on SED topics is easily accessible through queries, especially since the semantics of relation types can be easily injected into the queries. We hope the SE community utilizes this carefully curated knowledge base for numerous downstream tasks.

5.5 Threats to validity

In this section, we discuss the possible threats to the validity of this study and how we have addressed these threats.

Internal validity These threats correspond to the correctness of the relationships and the subjectiveness of contributors and maintainers. We address the former by evaluating every relationship through a partially objective labeling task in which at least three and up to nine contributors validate the correctness of the relationships. As for the latter, aside from the KG Initialization stage, the effect of maintainers' personal experience and knowledge was minimized by limiting their role to fixing consistency issues in the suggestions from the community in the continuous second stage. The subjectiveness of the contributors was also mitigated by engaging 170 experts as contributors. Although we invited people with relevant and sufficient background in SE and development, it is possible that some participants may not be familiar with some topics. To avoid inaccurate votes, we make it possible for contributors to only contribute to the topics related to their expertise and skip unfamiliar topics. Moreover, we provided background information on topics so that the participants can make more informed contributions. Another factor can be errors in our source code. We have double-checked the source code to decrease this threat,. However, there could be experimental errors in the setup that we did not notice. Therefore, we have publicly released our source code and dataset, to enable the community to use and/or replicate our work[10].

External validity These threats correspond to the generalizability and effectiveness of our graph and recommenders. Through crowd-sourcing and the expansion of SED-KGraph, we address the generalizability and effectiveness concerns. We also validated the relationships in multiple snapshots with the community, assuring their correctness and effectiveness. Although we use GitHub's featured set as the initial seed set, the resultant KG is not restricted to any platform and can be reused in other software-related platforms. Contributors were indeed instructed to incorporate their knowledge of Software Engineering while assessing/suggesting relationships and topics, irrespective of any specific SED platform. As for the topic recommendation, the KGRec component only takes the initial set of topics assigned to software projects as the input. Thus, it can be easily adapted for use on any software-related platform and any software entity. For the KGRec+ model, as long as proper textual information is available for a project, the model is able to recommend relevant topics. Moreover, the ML-based component can be re-trained with textual information of other software entities, using the model to recommend topics for those entities as well. Also for training, datasets were randomly split to avoid introducing bias. Finally, when assessing recommenders' performance, we use 50 repositories. Incorporating more repositories can improve the generalizability of our approach. However, assessing the correctness of the assigned missing tags requires *manual* inspection per repository and approach. Not to mention that each sample is examined by multiple evaluators to avoid introducing bias. This accumulates to a large number of evaluations and takes a lengthy time. Due to this, we take at most 50 repositories and could not afford to increase the number of repository samples.

Construct validity These threats correspond to the features and capabilities of the online platform used for KG expansion, and the sensitivity of KGRec to its input. We allowed for

[10] https://github.com/mahtab-nejati/KGRec

free-text topics and relation types in the suggestion forms to allow for the expansion of the KG. This resulted in consistency issues, which maintainers handled by refining the suggestions. This concern is further handled in the final version of the platform through the use of specially designed features and policies. Moreover, KGRec is sensitive to the correctness of the initial set of topics assigned to the project, such that the inaccuracy misleads the model on SED-KGraph. This limitation calls for an accurate ML-based component to be combined with KGRec. To address this threat, we used a multi-class multi-label LR classifier, which has been shown to exhibit the best performance among similar ML-based approaches (Izadi et al. 2021).

6 Related work

We organize the related work as approaches on (1) topic recommendation for software projects, (2) for other software entities, and finally studies on (3) KGs for software engineering.

Topic recommendation for software projects There are several studies with a focus on the topic recommendation for software projects (Xia et al. 2013; Xin-Yu Wang and Xia 2015; Vargas-Baldrich et al. 2015; Cai et al. 2016; Zhou et al. 2017; Wang et al. 2018; Wang et al. 2014; Liu et al. 2018; Di Sipio et al. 2020; Izadi et al. 2021). *Sally* presented by Vargas-Baldrich et al. (Vargas-Baldrich et al. 2015), is a tool for generating topics for Maven-based software projects by analyzing their bytecode and the dependency relations among them. Their approach, unlike ours, is limited in application due to being dependent on the programming language. Cai et al. (2016) proposed a graph-based cross-community approach called*GRETA*, for assigning topics to repositories. Their approach is to first construct an Entity-Tag Graph and for each queried project, take a random walk on a subset of the graph around the most similar entities to the queried one to assign tags to the project. While they do propose a graph-based approach, their graph fundamentally differs from ours in nature. Di Sipio et al. (2020), proposed using an MNB classifier for the classification of about 134 GitHub topics. In each top-k recommendation list, authors predict $k-1$ topics using text analysis and one topic using a code analysis tool called *GuessLang*. TopFilter (Di Rocco et al. 2020), the state-of-the-art for missing topic recommendation, is the most similar study to ours in terms of purpose (topic augmentation task). Authors take an item-based collaborative approach for recommending missing topics. Our experiments prove that their approach suffers from practicality issues, which our proposed approach overcomes. Most recently, Izadi et al. (2021), demonstrated the impact of clean topics and proposed a multi-label Logistic Regression classifier for recommending topics. Our approach is orthogonal to this study, thus we incorporate their approach in this work.

Topic recommendation for other software entities There are several pieces of research on tag recommendation for other types of software entities such as questions on Stack Overflow, Ask Ubuntu, and Ask Different (Wang et al. 2018; Wang et al. 2014; Zhou et al. 2017; Xia et al. 2013; Liu et al. 2018; Maity et al. 2019). The discussion around these tags and their usability in the SE community have been so fortified that the Stack Overflow platform has also developed a tag recommendation system of its own. These approaches mostly employ word similarity-based and semantic similarity-based techniques. Xia et al. (2013) focused on calculating the similarity based on the textual description. The authors propose *TagCombine* for predicting tags for questions using a multi-label ranking method based

on OneVsRest Naive Bayes classifiers. Semantic similarity-based techniques (Wang et al. 2018; Wang et al. 2014; Liu et al. 2018) consider text semantic information and perform significantly better than the former approach. Wang et al. (2018) and Wang et al. (2014), proposed *ENTAGREC* and *ENTAGREC++* which uses a mixture model based on LLDA to consider all tags together. Liu et al. (2018), proposed *FastTagRec*, for tag recommendation using a neural-network-based classification algorithm and bags of n-grams (bag-of-words with word order). As a possible future direction, our SE-based KG and approach can also be utilized for recommending topics for these types of software entities.

KGs for software engineering KGs have been utilized in numerous studies to address different software engineering problems. In some cases, researchers strive for a fully automated KG extraction approach from the available textual data. However, more often than not, these studies narrow the scope of their KG's content down to very specific aspects. In such cases, concept extraction from the textual data can be achieved through use case-specific tailored solutions, especially when the input data for KG construction is in a semi-structured or expected format (Li et al. 2018; Chen et al. 2019; Sun et al. 2019; Sun et al. 2021; Wang et al. 2017) or in some extreme cases, the concepts are predefined (Zhao et al. 2019). Some studies recognize that a fully-automated approach does not suffice their purpose, even in the limited scope of knowledge they intend to model and opt for semi-automated solutions (Karthik and Medvidovic 2019; Cao et al. 2019). HDSKG (Zhao et al. 2017) is a framework for mostly-automated KG construction. The authors applied their approach to the *tagWiki* pages on Stack Overflow in an attempt to construct a domain-specific KG of SED topics. The authors claim HDSKG includes 4,4800 unique concepts (topics) and 3,5279 relation triples of 9,660 unique verb phrases (relation types). While HDSKG can guarantee the lexical uniqueness of concepts and verb phrases through applying text processing techniques, the semantic uniqueness can not be promised. The lack of semantic uniqueness leads to tag explosion and redundant/duplicate verb phrases and relationships, which can be well hidden since neither the concepts nor the verb phrases are mapped to their semantically equivalent terms. Unfortunately, neither the resultant KG nor the code base of this work is publicly available. However, based on the sample nodes of HDSKG, its automatic method of extracting noun phrases results in tangled topics such as `small-java-library`. This justifies the enormous number of extracted topics and relationships. While HDSKG can be used in conjunction with our approach and replace the Acquisition process, the sheer multitude of the concepts and verb phrases works to the detriment of integrity and consistency concerns. The applicability of a KG of SED topics is highly sensitive to tag explosion and tangle topics problems, a threat that semi-automated and fully-automated approaches fail to mitigate. Especially with tangled topics as a concern while there are compound topics that convey atomic concepts as SED topics, each of the approaches leans towards detecting one and missing the other. Finally, some studies resort to fully-manual construction approaches due to data scatteredness and sparsity (Fathalla and Lange 2018) or make use of pre-constructed community-defined KGs for their purposes (Han et al. 2018). Consequently, we opted for a hybrid approach to avoid the pitfalls of each method described above, while obtaining high-quality topics and relationships as much as possible.

7 Future work

The main contributions of this study namely the SED-KGraph, the KGRec+ topic recommender, and also the platform all can be improved with future work. In the following,

we present possible directions for extending this work. In the future, we aim to maintain both SED-KGraph and the platform through which is expanded by the community. We can further improve different aspects of our platform and add more automated solutions. For instance, we aim to enhance our redundancy checker using ML techniques to discover semantically-similar topics that are written differently. The KG itself and the contributions received from the community can be further investigated. For instance, several factors including participants' numbers, duration of the snapshot, gender or background of participants, etc. differ for different snapshots. Such factors may impact the results of various snapshots. Further studies are required to explore the possible effects. The ML-based component for recommending topics can also be improved. For instance, one can also process the source files associated to repositories to enhance the ability of the classifiers and suggest more relevant topics. Moreover, one can invest in training contextual-based models to further improve the performance of the recommenders. Currently, our missing topic recommender does not utilize the relation types. Different relation types and their frequencies, if properly investigated, can potentially help build stronger recommenders. This is also a possible direction to investigate in future research. Finally, we aim to study other applications of SED-KGraph in different contexts such as search engines and information retrieval as well as other information websites such as Stack Overflow.

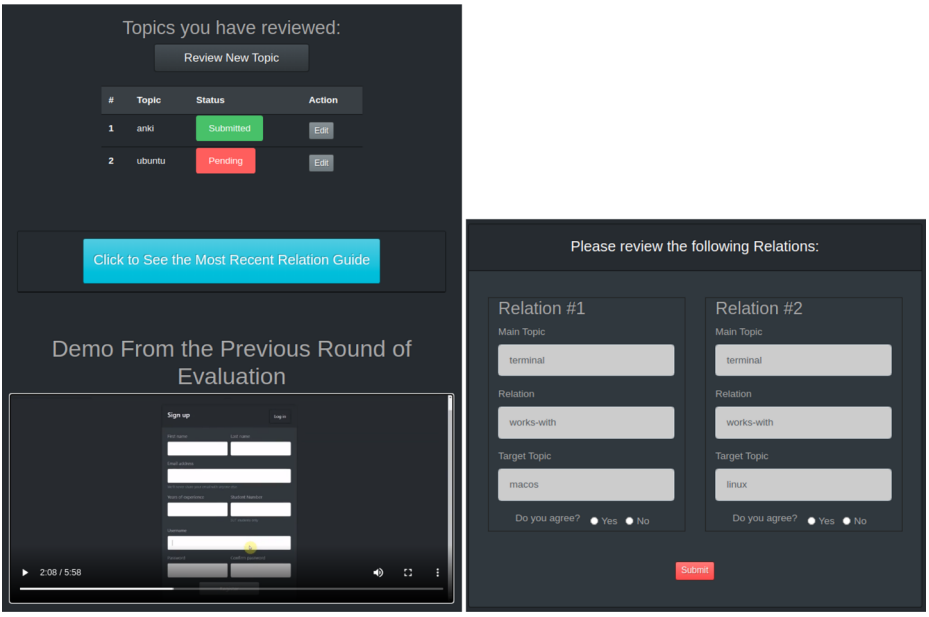
8 Conclusions

To discover the semantic relationships among SED-related topics, we engaged 170 researchers and practitioners to collaboratively construct a KG we call SED-KGraph. To initialize the KG, we first drafted a primary version, taking GitHub's featured topics as the seed topic set. Through their contributions, we constructed SED-KGraph with 2,234 carefully evaluated relationships among 863 community-curated topics. We also developed a platform through which we evaluated and expanded SED-KGraph in a crowd-sourced continuous method. Knowing that the KG will keep growing as do the SED technologies, we maintained the platform to sustain the continuous expansion of SED-KGraph in a more continuous and semi-automated manner requiring less human effort.

In the second stage of the proposed approach, we propose two recommender systems, KGRec and KGRec+ for tagging software projects augmented by the semantic relationship among their topics. We developed KGRec to predict the missing topics of software projects in GitHub based on SED-KGraph. The second recommender, however, assumes there is no topic available for a repository, and proceeds to predict the relevant topics based on both textual information of a software project (such as its README file), SED-KGraph, and its ML-based component. Our experiments yield that this model achieves $1.7X$ and $3.2X$ higher scores regarding ASR@5 and MAP@5, respectively. We also built upon the missing topic recommender (KGRec) and added an ML-based component to the approach to develop a stand-alone automated topic recommender system, KGRec+. The results show that KGRec+ outperforms the state-of-the-art baseline approaches as well as the modified and improved ones by at least $+25\%$ and $+23\%$ regarding ASR@5 and MAP@5 measures, respectively. Finally, we publicly share SED-KGraph, as a rich form of knowledge for the community to reuse and build upon. Furthermore, we release the source code of our two recommender models.

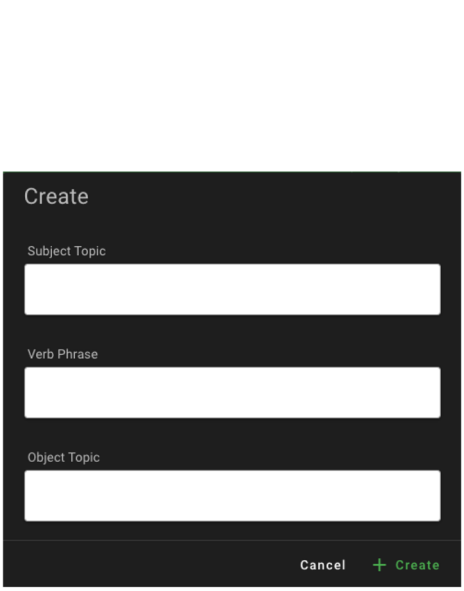
Appendix A: Platform Screenshots

Figures 6 and 7 present multiple screenshots of our online platform for both the construction and maintenance phases.

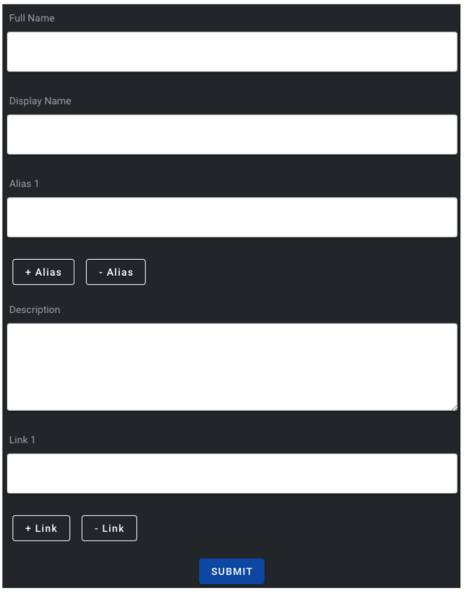


(a) Dashboard (b) Review a relationship

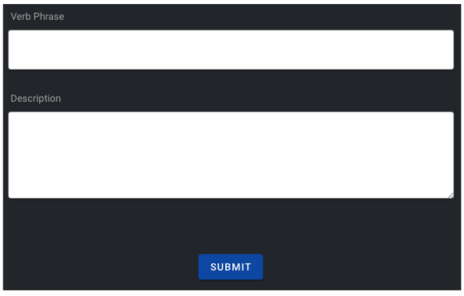
Fig. 6 Platform dashboard and review panel for KG construction phase



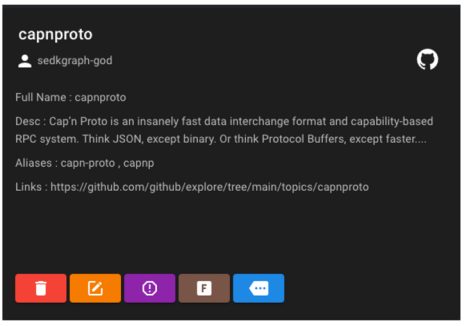
(a) Define new relationship



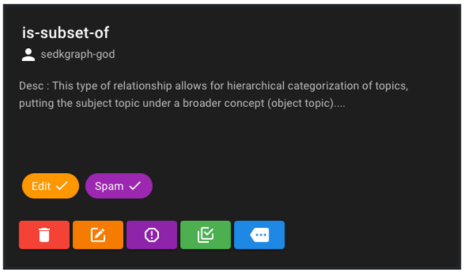
(b) Define new topic



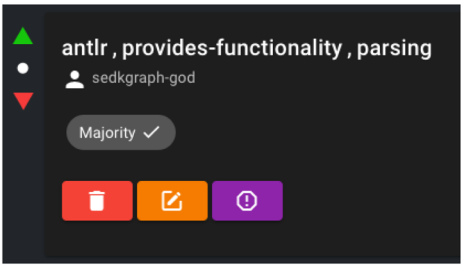
(c) Define new relation type



(d) A sample topic



(e) A sample relation type



(f) A sample relationship

Fig. 7 KG entities in the maintenance phase

Appendix B: Samples

Table 9 presents several samples per relation type from SED-KGraph.

Table 9 Samples per relation types

Relation Type	Samples
Is-a	(`django`, *is-a*, `framework`) (`android`, *is-a*, `operating-system`) (`atom`, *is-a*, `text-editor`)
Is-used-in-field	(`django`, *is-used-in-field*, `web-development`) (`3d`, *is-used-in-field*, `graphics`) (`azure`, *is-used-in-field*, `cloud-computing`)
Provides-functionality	(`django`, *provides-functionality*, `backend`) (`auth0`, *provides-functionality*, `authentication`) (`blockchain`, *provides-functionality*, `decentralization`)
Works-with	(`django`, *works-with*, `python`) (`blockchain`, *works-with*, `cryptography`) (`kubernetes`, *works-with*, `docker`)
Is-subset-of	(`image-processing`, *is-subset-of*, `machine-learning`) (`continuous-deployment`, *is-subset-of*, `cicd`) (`user-experience`, *is-subset-of*, `ui-ux`)
Is-based-on	(`archlinux`, *is-based-on*,`linux`) (`xmake`, *is-based-on*,`lua`) (`reactiveui`, *is-based-on*,`mvvm`)
Is-focused-on	(`agile`, *is-focused-on*, `speed`) (`end-to-end-encryption`, *is-focused-on*, `privacy`) (`neo4j`, *is-focused-on*, `scalability`)
Has-property	(`mysql`, *has-property*, `open-source`) (`anki`, *has-property*, `cross-platform`) (`elite-dangerous`, *has-property*, `multiplayer`)
Overlaps-with	(`robotics`, *overlaps-with*, `ai`) (`data-science`, *overlaps-with*, `ai`) (`nlp`, *overlaps-with*, `machine-learning`)
Provides-product	(`google`, *provides-product*, `flutter`) (`amazon`, *provides-product*, `aws`) (`mediawiki`, *provides-product*, `wikipedia`)
Provided-by	(`atom`, *provided-by*, `github`) (`flutter`, *provided-by*, `google`) (`macos`, *provided-by*, `apple`)
Maintained-by	(`html`, *maintained-by*, `w3c`) (`symfony`, *maintained-by*, `sensiolabs-sas`) (`uportal`, *maintained-by*, `apereo`)
Has-license	(`backbonejs`, *has-license*, `mit-license`) (`ansible`, *has-license*, `gnu-gpl-license`) (`robotframework`, *has-license*, `apache-license`)

Acknowledgements We would like to thank all the participants for helping us with constructing and evaluating our KG, as well as for assessing our recommender model.

Data Availability The dataset generated during the current study is available in the authors' public GitHub repository.[11] The dataset used for training the ML-based components and comparing approaches is also available in the baseline paper's public GitHub repository.[12]

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

[11] https://github.com/mahtab-nejati/KGRec

[12] https://github.com/MalihehIzadi/SoftwareTagRecommender

References

Alonso O, Marshall C, Najork M (2014) Crowdsourcing a subjective labeling task: a human-centered framework to ensure reliable results. Microsoft Res, Redmond, WA, USA, Tech Rep MSR-TR:2014–91

Bollacker K, Evans C, Paritosh P, Sturge T, Taylor J (2008) Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD international conference on management of data, pp 1247–1250

Cai X, Zhu J, Shen B, Chen Y (2016) Greta: graph-based tag assignment for github repositories. In: In proceedings of the 40th annual computer software and applications conference (COMPSAC). IEEE, vol 1, pp 63–72

Cao J, Du T, Shen B, Li W, Wu Q, Chen Y (2019) Constructing a knowledge base of coding conventions from online resources. In: The international conference on software engineering and knowledge engineering (SEKE), pp 5–14

Chen D, Li B, Zhou C, Zhu X (2019) Automatically identifying bug entities and relations for bug analysis. In: 2019 IEEE 1st international workshop on intelligent bug fixing (IBF), pp 39–43

Crestani F (1997) Application of spreading activation techniques in information retrieval. Artif Intell Rev 11(6):453–482

Di Rocco J, Di Ruscio D, Di Sipio C, Nguyen P, Rubei R (2020) Topfilter: an approach to recommend relevant github topics. In: In proceedings of the 14th international symposium on empirical software engineering and measurement (ESEM). ACM, ESEM '20, New York

Di Sipio C, Rubei R, Di Ruscio D, Nguyen PT (2020) A multinomial naïve bayesian (mnb) network to automatically recommend topics for github repositories. In: In proceedings of the 24th international conference on evaluation and assessment in software engineering (EASE). ACM, pp 71–80

Dong L, Wei F, Zhou M, Xu K (2015) Question answering over freebase with multi-column convolutional neural networks. In: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (vol 1: long papers), pp 260–269

Escobar-Avila J, Linares-Vásquez M, Haiduc S (2015) Unsupervised software categorization using bytecode, pp In proceedings of the 23rd international conference on program comprehension (ICPC). IEEE, pp 229–239

Fathalla S, Lange C (2018) Eventskg: a knowledge graph representation for top-prestigious computer science events metadata. In: In proceedings of the 10th international conference on computational collective intelligence (ICCCI). Springer, pp 53–63

Golder SA, Huberman BA (2006) Usage patterns of collaborative tagging systems. J Inf Sci 32(2):198–208

Han Z, Li X, Liu H, Xing Z, Feng Z (2018) Deepweak: reasoning common software weaknesses via knowledge graph embedding. In: In proceedings of the 25th international conference on software analysis, evolution and reengineering (SANER). IEEE, pp 456–466

Held C, Kimmerle J, Cress U (2012) Learning by foraging: the impact of individual knowledge and social tags on web navigation processes. Comput Hum Behav 28(1):34–40

Izadi M, Ahmadabadi MN (2022) On the evaluation of nlp-based models for software engineering. In: 2022 IEEE/ACM 1st international workshop on natural language-based software engineering (NLBSE). IEEE computer society, USA, pp 48–50

Izadi M, Akbari K, Heydarnoori A (2022) Predicting the objective and priority of issue reports in software repositories. Empir Softw Eng 27(2):1–37

Izadi M, Heydarnoori A, Gousios G (2021) Topic recommendation for software repositories using multi-label classification algorithms. Empir Softw Eng 26(5):1–33

Karthik S, Medvidovic N (2019) Automatic detection of latent software component relationships from online qa sites. In: Proceedings of the 7th international workshop on realizing artificial intelligence synergies in software engineering (RAISE). IEEE Press, pp 15–21

Li H, Li S, Sun J, Xing Z, Peng X, Liu M, Zhao X (2018) Improving api caveats accessibility by mining api caveats knowledge graph. In: In proceedings of the 34th international conference on software maintenance and evolution (ICSME), pp 183–193

Liu J, Zhou P, Yang Z, Liu X, Grundy J (2018) Fasttagrec: fast tag recommendation for software information sites. Autom Softw Eng 25(4):675–701

Maity SK, Panigrahi A, Ghosh S, Banerjee A, Goyal P, Mukherjee A (2019) Deeptagrec: a content-cum-user based tag recommendation framework for stack overflow. In: In proceedings of the 41st european conference on information retrieval (ECIR). Springer, pp 125–131

Mazrae PR, Izadi M, Heydarnoori A (2021) Automated recovery of issue-commit links leveraging both textual and non-textual data. In: 2021 IEEE international conference on software maintenance and evolution (ICSME). IEEE computer society, USA, pp 263–273

McMillan C, Grechanik M, Poshyvanyk D (2012) Detecting similar software applications. In: In proceedings of the 34th international conference on software engineering (ICSE). IEEE, pp 364–374

Reyes J, Ramírez D, Paciello J (2016) Automatic classification of source code archives by programming language: a deep learning approach. In: 2016 International conference on computational science and computational intelligence (CSCI), pp 514–519

Sun J, Xing Z, Chu R, Bai H, Wang J, Peng X (2019) Know-how in programming tasks: from textual tutorials to task-oriented knowledge graph. In: IEEE international conference on software maintenance and evolution (ICSME), pp 257–268, 09

Sun J, Xing Z, Peng X, Xu X, Zhu L (2021) Task-oriented api usage examples prompting powered by programming task knowledge graph. In: 2021 IEEE international conference on software maintenance and evolution (ICSME). IEEE, pp 448–459

Thung F, Lo D, Jiang L (2012) Detecting similar applications with collaborative tagging. In: In proceedings of the 28th international conference on software maintenance (ICSM). IEEE, pp 600–603

Vargas-Baldrich S, Linares-Vásquez M, Poshyvanyk D (2015) Automated tagging of software projects using bytecode and dependencies (n). In: In proceedings of the 30th international conference on automated software engineering (ASE). IEEE, pp 289–294

Wagner S, Fernández DM (2015) Chapter 3 - analyzing text in software projects. In: Bird C, Menzies T, Zimmermann T (eds) The art and science of analyzing software data. Morgan Kaufmann, Boston, pp 39–72

Wang H, Zhang F, Wang J, Zhao M, Li W, Xie X, Guo M (2018) Ripplenet: propagating user preferences on the knowledge graph for recommender systems. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM). ACM, New York, pp 417–426

Wang L, Sun X, Wang J, Duan Y, Li B (2017) Construct bug knowledge graph for bug resolution. In: In proceedings of the 39th international conference on software engineering companion (ICSE-C). IEEE, pp 189–191

Wang S, Lo D, Vasilescu B, Serebrenik A (2018) Entagrec++: an enhanced tag recommendation system for software information sites. Empir Softw Eng 23(2):800–832

Wang T, Wang H, Yin G, Ling CX, Li X, Zou P (2014) Tag recommendation for open source software. Frontiers Comput Sci (FCS) 8(1):69–82

Xia X, Lo D, Wang X, Zhou B (2013) Tag recommendation in software information sites. In: 2013 10th Working conference on mining software repositories (MSR). IEEE, pp 287–296

Xin-Yu Wang DL, Xia X (2015) Tagcombine: recommending tags to contents in software information sites. J Comput Sci Technol 30(5):1017

Xu K, Reddy S, Feng Y, Huang S, Zhao D (2016) Question answering on freebase via relation extraction and textual evidence

Yang Y, Li Y, Yue Y, Wu Z, Shao W (2016) Cut: a combined approach for tag recommendation in software information sites. In: Lehner F, Fteimi N (eds) Knowledge science, engineering and management. Springer, Cham, pp 599–612

Yao X, B. Van Durme. (2014) Information extraction over structured data: question answering with freebase. In: Proceedings of the 52nd annual meeting of the association for computational linguistics (vol 1: long papers), pp 956–966

Zhang E, Banovic N (2021) Method for exploring generative adversarial networks (gans) via automatically generated image galleries. In: Proceedings of the conference on human factors in computing systems (CHI), pp 1–15

Zhang Y, Lo D, Kochhar PS, Xia X, Li Q, Sun J (2017) Detecting similar repositories on github. In: In proceedings of the 24th international conference on software analysis, evolution and reengineering (SANER). IEEE, pp 13–23

Zhang Y, Xu FF, Li S, Meng Y, Wang X, Li Q, Han J (2019) Higitclass: keyword-driven hierarchical classification of github repositories. In: 2019 IEEE international conference on data mining (ICDM). IEEE, pp 876–885

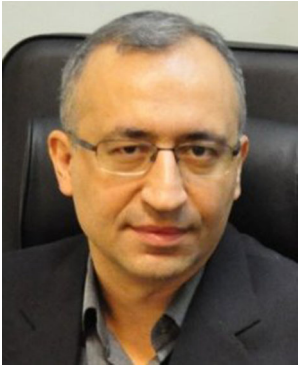
Zhao X, Xing Z, Kabir MA, Sawada N, Li J, Lin S (2017) Hdskg: harvesting domain specific knowledge graph from content of webpages. In: In proceedings of the 24th international conference on software analysis, evolution and reengineering (SANER), pp 56–67

Zhao Y, Wang H, Ma L, Liu Y, Li L, Grundy J (2019) Knowledge graphing git repositories: a preliminary study. In: 2019 IEEE 26th international conference on software analysis, evolution and reengineering (SANER), pp 599–603

Zhou P, Liu J, Yang Z, Zhou G (2017) Scalable tag recommendation for software information sites. In: In proceedings of the 24th international conference on software analysis, evolution and reengineering (SANER). IEEE, pp 272–282
Zou X (2020) A survey on application of knowledge graph. J Phys Conf Ser 1487(03):012016

Maliheh Izadi is currently a post-doctoral researcher and lecturer in the Software Engineering Research Group at the Delft University of Technology, Netherlands. Her research lies in the intersection of software engineering and machine learning. Specifically, she uses natural and programming language processing techniques to build recommender systems for various software development tasks.

Mahtab Nejati is currently a PhD candidate in the Software Analysis Group at University of Waterloo, where she studies and analyzes software artifacts to improve release pipelines.



Abbas Heydarnoori is a faculty member in the Department of Computer Science at Bowling Green State University, USA. Dr. Heydarnoori has also worked as a faculty member in the Department of Computer Engineering at the Sharif University of Technology, directing the Intelligent Software Engineering Lab. Before that, he was a post-doctoral fellow in the Faculty of Informatics at the University of Lugano, Switzerland. Dr. Heydarnoori finished his Ph.D. studies in the School of Computer Science at the University of Waterloo, Canada, in 2009. His research interests focus on Software Analytics, Empirical Software Engineering, and Intelligent Software Engineering.