A knowledge graph empowered online learning framework for access control decision-making

World Wide Web
https://doi.org/10.1007/s11280-022-01076-5

A knowledge graph empowered online learning framework for access control decision-making

Mingshan You[1] · Jiao Yin[1] · Hua Wang[1] · Jinli Cao[2] · Kate Wang[3] · Yuan Miao[4] · Elisa Bertino[5]

Received: 20 January 2022 / Revised: 20 May 2022 / Accepted: 9 June 2022

Abstract
Knowledge graph, as an extension of graph data structure, is being used in a wide range of areas as it can store interrelated data and reveal interlinked relationships between different objects within a large system. This paper proposes an algorithm to construct an access control knowledge graph from user and resource attributes. Furthermore, an online learning framework for access control decision-making is proposed based on the constructed knowledge graph. Within the framework, we extract topological features to represent high cardinality categorical user and resource attributes. Experimental results show that topological features extracted from knowledge graph can improve the access control performance in both offline learning and online learning scenarios with different degrees of class imbalance status.

Keywords Knowledge graph · Online learning · Access control · High cardinality categorical data

1 Introduction

With the popularization of information systems and digital devices, enterprises and organizations accumulate a large amount of valuable or sensitive data locally or in the cloud [14, 23, 33]. Once these data are leaked or used maliciously, it will cause

This article belongs to the Topical Collection: *Special Issue on Web Information Systems Engineering 2021*
Guest Editors: Hua Wang, Wenjie Zhang, Lei Zou, and Zakaria Maamar

✉ Jiao Yin
j.yin@latrobe.edu.au

[1] Institute for Sustainable Industries, Liveable Cities, Victoria University, Melbourne, VIC 3083, Australia

[2] Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC 3083, Australia

[3] School of Health and Biomedical Sciences, RMIT University, Melbourne, Australia

[4] College of Engineering and Science, Victoria University, Melbourne, VIC 3083, Australia

[5] Department of Computer Science, Purdue University, West Lafayette, IN, USA

significant economic losses or pose a great threat to users' privacy [7, 25, 29]. Secure sensitive information is an important issue to protect customers and then attract users [8, 43]. Access control is recognised as the first defence to guarantee that only authorized users can gain access to sensitive data and thus prevent data leakage [22, 25, 30, 32].

The two main categories of the most widely used access control strategies are role-based access control (RBAC) strategies and attribute-based access control (ABAC) strategies [25, 28, 31]. The former assigns permissions only based on user's roles, which makes it simple to implement and thus widely used in the past [4, 24]. However, with the expansion of the information system scale and the proliferation of users, RBAC strategies are too coarse-grained to meet the needs of sensitive data protection [27, 36]. By contrast, ABAC strategies adopt carefully crafted policies based on multiple attributes from users, environment and resources to assign data access permission. ABAC strategies have become more popular nowadays because they are more fine-grained and flexible than RBAC strategies [17, 34]. For example, the work in [9] proposed an ABAC mining algorithm named Rhapsody to mine ABAC rules from sparse logs.

However, the evolving new information technologies and changes in users' behaviours bring new challenges [13, 15]. One of the biggest problems is the policy explosion, which means the scale of the policy has increased dramatically [18, 35]. The main reason for the policy explosion is that users' roles in organizations are becoming more diverse and people are using more different devices to access data in different places. Policy explosion brings two consequences directly, i.e., decreased efficiency of the system and increased misconfiguration [12, 16, 20].

To overcome these problems, more and more researchers are beginning to explore machine learning based access control strategies, which treat access control decision-making as a binary classification problem. Sample features for machine learning classifier training come from available users, environment and resource attributes etc. The corresponding sample labels come from verified access control log files. Some works have successfully classified access control historical records using machine learning methods with high accuracy. But there is no related work that discusses machine learning based access control from the perspective of a data stream. In reality, access control requests form a data stream to feed into the decision making models. Therefore, the work in our previous paper [42] proposed a consecutive batch learning framework to tackle the possible concept drifts by periodically updating the machine learning classifier with new samples.

Furthermore, dynamic class imbalance problems exist in real-world access control applications [39, 41]. In other words, most requests are legitimate and valid, but there will be a very small number of samples that are denied access due to mishandling or malicious attacks. For access control, a rejected access request usually means a malicious access request, which is the minority class. Misclassification of the minority class will cause severe data leakage. Therefore, improving the classification performance of the minority class (access deny) is vital for an access control problem.

To boost the performance of the minority class for access control, our previous work [42] proposed a Boosting Window (BW) algorithm within an adaptive incremental batch learning framework. Although experimental results demonstrated this work can enhance the performance of the minority class, the overall performance is still unsatisfactory because of the limited available attributes and the poor encoding and feature representing methods for high cardinality categorical data. For example, the manager ID is an essential user attribute related to the possible access permission to a specific system resource. However, in a large organization, such as Amazon, there will be millions of different manager

IDs. In this case, the values of the manager ID are high cardinality nominal categorical data.

In general practice, one-hot encoding, binary encoding and label encoding are the most popular methods for categorical data encoding [38]. When encoding high cardinality nominal categorical data, all of them have fatal disadvantages. Label encoding can mislead the classifier because of the big differences between numerical values. For example, the classifier can falsely give more weight to a manager with an ID of 100,000 than 1. One-hot encoding can address this problem but it will result in another serious problem, the curse of dimensionality. Binary encoding adopts binary code to represent ordinal values, working as a compromise between label encoding and one-hot encoding. However, binary encoding fails to represent relationships between different samples with the same attribute value.

Instead of encoding the values of different attributes as non-topological features for access control decision making, we extract relationships between entities from attributes and further extract topological features from relationships to train access control classifiers. To better represent user and resource and illustrate relationships between them, this paper constructs an access control domain-specific knowledge graph to assist decision making. As an extension of our previous work, we leverage knowledge graph to handle user and resource attributes with high cardinality values to further boost the performance of the minority class. Compared with the work in [42], our main contributions are as follows.

(1) We proposed a knowledge graph empowered online learning framework for access control decision making. To the best of our knowledge, this paper is the first try to leverage knowledge graph to extract graph topological features to improve the performance of the access control model.
(2) We proposed an algorithm to construct a knowledge graph from the existing user and resource attributes. We further demonstrate how to extract features from the established knowledge graph to represent users and resources. The extracted features are fed to a machine learning classifier to make access control decisions based on records in log files.
(3) We evaluate and verify the proposed knowledge graph empowered online learning framework on a much larger open-sourced real-word dataset and discussed the performance on different imbalance degrees in both online and offline scenarios.

The rest of the paper is organised as follows. Section 2 briefly introduces knowledge graph basics, typical graph topological features and link prediction solutions. Section 3 presents the workflow of the proposed framework, followed by an access control domain-specific knowledge graph construction algorithm and feature extraction details in Section 4. Section 5 displays the experimental results and concludes the paper with a discussion on future work.

2 Related work

A knowledge graph (KG), denoted as $\mathcal{G}$, is a multi-relational graph composed of entities as nodes and relations as different types of edges [37]. An instance of an edge is a triplet of fact (head entity, relation, tail entity), denoted as (h, r, t). Apart from the graph-structured data model, both entities and relationships can have multidimensional properties

to further describe complex data. KG is often used to represent interlinked facts, allowing both humans and computers to extract useful knowledge and further to do reasoning and prediction based on its contents. Typical ways to analyse a knowledge graph include but are not limited to (1) node classification to predict the type of a given node; (2) link prediction to predict whether two entities are linked or not; (3) community detection to identify densely linked entity clusters and (4) network similarity measurement to evaluate the similarity between two nodes or two networks.

The access control problem can be formulated as a link prediction problem between user entities and resource entities, which is essentially a binary classification problem. Specifically, if an access approve link exists between a user entity u and a resource entity r, the access request $(u \rightarrow r)$ will be approve. Otherwise, it will be refused. Once a knowledge graph has been constructed, a variety of graph topological features can be extracted to describe the local or global connections between entities based on homogeneous or heterogeneous subgraphs within the knowledge graph.

A basic solution for link prediction is structural similarity-based unsupervised learning methods, which determine the likelihood of linkage between two nodes based on some similarity or closeness indices deduced from the graph structure. When an index between two nodes exceeds a predefined threshold, they are considered to have a link between them. Common Neighbours (CN) [10] measuring the number of shared nodes between two nodes is the most intuitionistic index to indicate the linkage possibility of them. Similar indices, to name a few, include Adamic Adar (AA) [2], Preferential Attachment (PA) [3] and Resource Allocation (RA) [44]. Their definitions are listed as (1)-(3) for reference.

$$CN(u, v) = |\mathcal{N}(u) \cap \mathcal{N}(v)|, \tag{1}$$

$$AA(u, v) = \sum_{w \in \mathcal{N}(u) \cap \mathcal{N}(v)} \frac{1}{\log |\mathcal{N}(w)|}, \tag{2}$$

$$PA(u, v) = |\mathcal{N}(u)| * |\mathcal{N}(v)|, \tag{3}$$

$$RA(u, v) = \sum_{w \in \mathcal{N}(u) \cap \mathcal{N}(v)} \frac{1}{|\mathcal{N}(w)|}, \tag{4}$$

where u, v, w are nodes in the target graph, $\mathcal{N}(\cdot)$ denotes the set of nodes adjacent to the specified node in the brackets, $|\cdot|$ denotes the number of distinct nodes in the specified set. These indices are widely used in various domains because of their simplicity and reasonable performance. However, they only considered the node pair's local connectivity and ignored the global structure of a graph.

By contrast, global connectivity indices can provide more overall graph topology information. A well-known index for taking global connectivity into account is the Katz Index (KI), which leverage the length of paths between a pair of nodes to measure their similarity. KI can be calculated as (5) [11].

$$KI(u,v) = \sum_{l=1}^{l_{\max}=\infty} \beta^l \cdot \left|\text{path}_{u,v}^l\right|, \tag{5}$$

where l is the length of a path between nodes u and v, $\left|\text{path}_{u,v}^l\right|$ is the total number of distinct paths between node u and v with length l, β is a coefficient between 0 and 1 used to adjust the contribution of paths to KI.

Another popular global connectivity index is Average Commute Time (ACT), which calculates the average number of steps required by a random walker starting from node u to reach v and vice versa [1]. The ACT between nodes u and v can be calculated as (6) [26].

$$S_{ACT}(u,v) = \frac{1}{l_{uu}^+ + l_{vv}^+ - 2l_{uv}^+} \tag{6}$$

where l_{uu}^+, l_{vv}^+ and l_{uv}^+ are the corresponding entries in Laplacian Matrix, L^+.

Obviously, a common drawback for global connectivity indices is relatively higher computation cost compared with local connectivity indices. Decentralized approaches or parallel computing are also incapable of dealing with global graph computation, because the structural connectivity would be damaged by splitting the graph for decentralized or parallel computing. Therefore, these measures are not suitable for large-scale connected graphs.

Generally speaking, the common advantages of structural similarity-based unsupervised learning methods algorithms include that (1) they do not need labelled data to train a classifier; (2) the link prediction result is explainable based on the definition of the corresponding indices; (3) they often take less computation effort for costly feature engineering and classifier training procedures.

However, there is still no universal feasible method to determine the appropriate threshold for different indices and application domains. Besides, these methods are also criticised for poor performance due to only taking topological features into account and neglecting the attributes of nodes and relationships, which contain rich domain knowledge and play critical roles for most domain-specific link prediction tasks. Therefore, in most cases, when labelled data is available, supervised learning methods are more preferable due to superior performance and the flexibility of feature extraction.

When applying supervised learning methods, both non-topological features and topological features can be used to feed into a machine learning classifier to support link prediction. Non-topological features refer to the attributes of entities and relationships, which contain rich multi-modality domain knowledge. For example, in an access control knowledge graph, the non-topological features of a user entity include sector, department name, job title, job description, etc. By contrast, topological features refer to graph structural features for node representing. In addition to aforementioned local and global connectivity indices, common traditional node topological feature extraction methods in graph theory include Page Rank [6], Article Rank [19], Betweenness Centrality [5], Harmonic Centrality [21], etc.The performance of supervised machine learning methods for link prediction is determined by the capability of the extracted non-topological and graph topological features as well as the capability of the applied classifier.

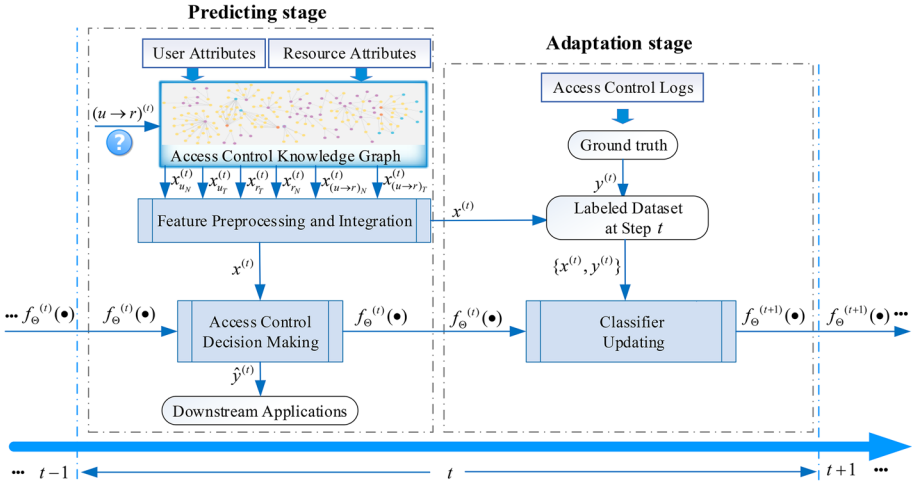

Fig. 1 Workflow of the proposed consecutive incremental batch learning framework

3 Methodology

We propose a general knowledge graph empowered online learning framework for access control in this section. Firstly, we introduce the workflow of the framework. Then we detail the construction algorithm of an access control domain-specific knowledge graph and the KG-based topological feature extraction method.

3.1 Workflow of the proposed framework

The supporting information for the access control decision-making problem studied in this paper includes user attributes, resource attributes and a verified access control log file in chronological order. According to the cardinality of category user attributes and resource attributes, an access control knowledge graph is constructed. The specific knowledge graph construction and refactoring algorithm is given in Section 3.2 and a real-world use case is demonstrated in Section 4.

Similar to our previous work [42], the proposed framework is essentially a classifier-agnostic consecutive incremental batch learning process for access control decision-making. Within this framework, a randomly initialized binary machine learning classifier works as the access control decision-maker, denoted as $f_{\Theta}^{(0)}(\cdot)$, where Θ is the trainable parameter set of $f(\cdot)$ and (0) means the initialization status of the time step. The classifier $f_{\Theta}^{(0)}(\cdot)$ is constantly updated at each time step as new samples are available for classifier training. We demonstrate the main process of a typical time step t ($t > 0$) in Figure 1, which consists of two stages, namely, the predicting stage and the adaptation stage.

At the predicting stage of the t-th time step, when user u request a resource r, denoted as $(u \rightarrow v)^{(t)}$, the classifier $f_{\Theta}^{(t)}(\cdot)$, which is updated at time step $t-1$, will make decision on the access control request $(u \rightarrow v)^{(t)}$. Firstly, six feature sets related to this access control

request will be extracted from the constructed access control knowledge graph, i.e., $x^{(t)}_{u_N}$, $x^{(t)}_{u_T}$, $x^{(t)}_{r_N}$, $x^{(t)}_{r_T}$, $x^{(t)}_{(u\to r)_N}$ and $x^{(t)}_{(u\to r)_T}$. Among them, $x^{(t)}_{u_N}$, $x^{(t)}_{r_N}$ and $x^{(t)}_{(u\to r)_N}$ are the non-topological feature sets extracted from the user entity, the resource entity and the existing relationships between them. Similarly, $x^{(t)}_{u_T}$, $x^{(t)}_{r_T}$ and $x^{(t)}_{(u\to r)_T}$ are the corresponding topological feature sets. The details of the feature extraction process are described in Section 3.3. These six feature sets are then preprocessed (outlier replacing and normalization) and integrated into one feature set $x^{(t)}$. Finally, the classifier $f^{(t)}_{\Theta}(\cdot)$ will make decision on the request $(u \to v)^{(t)}$ according to the result of equation (7).

$$\hat{y}^{(t)} = f^{(t)}_{\Theta}(x^{(t)}), (t > 0). \tag{7}$$

At the adaptation stage of the t-th time step, the verified ground truth $y^{(t)}$ corresponding to the request $(u \to v)^{(t)}$ is available and can be extracted from the verified access control log file. Then, the labelled samples $x^{(t)}$, $y^{(t)}$ can be used to finetune the classifier $f^{(t)}_{\Theta}(\cdot)$. The fine-tuned classifier, denoted as $f^{(t+1)}_{\Theta}(x^{\cdot})$, will be used at the predicting stage of the $t+1$-th time step. Since the classifier keeps updating with the latest verified samples $\{x^{(t)}, y^{(t)}\}$ at each time step t, it can learn possible new concepts emerging at time step t.

3.2 Access control knowledge graph construction

A knowledge graph consists of a set of entities (with multiple entity labels) and relationships between entities (with multiple relationship types). Each entity or relationship has its identification number and some of them have one or more properties. To construct an access control knowledge graph is to identify all entities including their labels and properties, and all relationships including their relationship types.

3.2.1 Attribute type

We construct an access control knowledge graph $\mathcal{G}$ from existing user attributes and resource attributes information. Apart from the ID attribute, from the perspective of constructing KGs, there are three kinds of attributes in Att_u and Att_r, i.e., Type 1, attributes showing the relationships between users and resources; Type 2, high cardinality categorical attributes; Type 3, the rest attributes. Let θ be a preset cardinality threshold. If the cardinality of a categorical attribute is larger than θ, it is a Type 2 attribute; otherwise, Type 3. The attribute types work as a guideline for step by step knowledge graph construction, seeing details in Section 3.2.2.

3.2.2 Algorithm pseudocode

Let Att_u be the list of users' attribute names, in which an attribute name'userID' is included. X_u denotes the attributes' values according to Att_u. $x_{uid} \in X_u$ is a vector containing all users' ID. Similarly, Att_r is the list of resources' attribute names containing a 'resourceID'. X_r is the attributes' values according to Att_r and $x_{rid} \in X_r$ is a vector containing all resources' ID. We elaborate on the construction process of an access control knowledge graph in Algorithm 1.

Algorithm 1 Access Control Knowledge Graph Construction

Input: Att_u, X_u, Att_r, X_r, θ.
Output: $\mathcal{G}$.

\\ Step1: create User and Resource entities.
for *uid* in unique(x_{uid}) **do**
  CREATE (u:User {u.userID=*uid*})
end for
for *rid* in unique(x_{rid}) **do**
  CREATE (r:Resource {r.resourceID=*rid*})
end for
\\ Step2: create properties or relationships for User entities from user attributes.
for *attn*, x_u in zip(Att_u, X_u-x_{uid}) **do**
  \\ Step2.1: create relationships from User entities to Resource entities from Type 1 attributes.
  if x_u shows a relationship between users and resources **then**
    for *uid*, *attv* in zip(x_{uid}, x_u) **do**
      Let the set of resourceIDs indicated by *attv* be denoted as Rid_t.
      for rid_t in Rid_t **do**
        MATCH (u:User userID:*uid*)
        MATCH (r:Resource resoureID:rid_t)
        CREATE (u)-[ref:HAS*attn* (ref.*attn*Property=*attv*)]->(r)
      end for
    end for
    \\ Step2.2: create new types of entities from the Type 2 attributes.
  else if x_u is a category feature AND cardinality(x_u) > θ **then**
    for *uid*, *attv* in zip(x_{uid}, x_u) **do**
      CREATE (a:*attn* {a.*attn*Property=*attv*})
      CREATE (u)-[ref:HAS*attn*]->(a)
    end for
    \\ Step2.3: create new properties for User entities from the Type 3 attributes.
  else
    for *uid*, *attv* in zip(x_{uid}, x_u) **do**
      MATCH (u:User userID:*uid*)
      SET u.*attn*Property=*attv*
    end for
  end if
end for
\\ Step3: create properties or relationships for Resource entities from Resource attributes.
for *attn*, x_r in zip(Att_r, X_r-x_{rid}) **do**
  \\ Step3.1: create relationships from Resource entities to User entities from Type 1 attributes.
  if x_r shows a relationship between resources and users **then**
    for *rid*, *attv* in zip(x_{rid}, x_r) **do**
      Let the set of userIDs indicated by *attv* be denoted as Uid_t.
      for uid_t in Uid_t **do**
        MATCH (r:Resource resoureID:*rid*)
        MATCH (u:User userID:uid_t)
        CREATE (r)-[ref:HAS*attn* (ref.*attn*Property=*attv*)]->(u)
      end for
    end for
    \\ Step3.2: create new types of entities from the Type 2 attributes.
  else if x_r is a category feature AND cardinality(x_r) > θ **then**
    for *rid*, *attv* in zip(x_{rid}, x_r) **do**
      CREATE (a:*attn* {a.*attn*Property=*attv*})
      CREATE (r)-[ref:HAS*attn*]->(a)
    end for
    \\ Step3.3: create new properties for Resource entities from the Type 3 attributes.
  else
    for *rid*, *attv* in zip(x_{rid}, x_u) **do**
      MATCH (r:Resource resourceID:*rid*)
      SET r.*attn*Property=*attv*
    end for
  end if
end for
\\ Step4: refactor the above-established access control knowledge graph.
for *rel* in *Rel* **do**
  MATCH (u1:User)-[:*rel*]->()<-[:*rel*]-(u2:User)
  CREATE (u1)-[:SHARE*rel*]-(u2)
  MATCH (r1:Resource)-[:*rel*]->()<-[:*rel*]-(r2:Resource)
  CREATE (r1)-[:SHARE*rel*]-(r2)
end for
return an access control knowledge graph $\mathcal{G}$

Some executive statements of the pseudocode in Algorithm 1 are written in Cypher query language, which is the graph query language for the Neo4j graph database. The naming convention thus follows the Cypher coding standards, where entity labels are in CamelCase; property keys are in camelCase and relationship types are in uppercase, such as FOLLOWS in a social media knowledge graph. As listed below, the process of access control knowledge graph construction can be divided into four main steps:

Step 1: create User and Resource entities as shown in lines 2-7. According to the userID and resourceID attributes, we create two entity types with a User label and Resource label respectively. For each unique userID *uid* in x_{uid}, we create a User entity with a userID property as shown in lines 2-4. Similarly, we create Resource entities with a resouceID property based on the *rid* in x_{rid} as shown in lines 5-7.

Step 2: create properties or relationships for User entities from user attributes as shown in lines 9-32. In line 9, *attn* refers to the attribute name traversing Att_u; x_u is the corresponding attribute values and X_u-x_{uid} means the relative complement of x_{uid} in X_u. In other words, X_u-x_{uid} means all attributes' values except x_{uid}. Steps 2.1-2.3 give details on how to create properties or relationships for User entities based on three attribute types defined in Section 3.2.1.

Step 2.1: create relationships from User entities to Resource entities based on Type 1 attributes as shown in lines 11-19. When a user attribute *attn* indicates a relationship between users and resources, we search the particular User entity and Resource entity based on the attribute value *attv* and create a HAS*attn* relationship between, where HAS*attn* is the relationship type in upper-case format. We further record the import attribute information in *attv* as a property of the created relationship, named *attn* Property. In line 12, *attv* means the attribute value of *attn* corresponding to the user with userID=*uid*. In line 13, Rid_t means a temporary resourceID set to distinguish it from x_{rid} line 5.

Step 2.2: create new types of entities for high cardinality categorical user attributes as shown in lines 21-25. To better represent high cardinality categorical features, we create new entities with a label named *attn* and a property named *attn* Property to record the value of the high cardinality categorical user attribute. Then, we create a relationship with a type of HAS*attn* to indicate that the user with userID=*uid* has a relation with the newly created entity.

Step 2.3: create new properties for User entities from the Type 3 attributes. The rest of user attributes are all added as the properties of the User entities as shown in lines 28-31.

Step 3: create properties or relationships for Resource entities from resource attributes as shown in lines 34-57. The process of Step 3 is similar to Step2. To avoid redundancy, we no longer describe the detailed process in words.

Step 4: refactor the above-established access control knowledge graph as shown in lines 61-61. We add a SHARE*ref* relationship between User Entities who share the same *attn* entities created in line 23. The subgraph consisting of SHARE*ref* relationships and User entities can be used to extract topological features to represent the original user attribute $attn \in Att_u$. Similarly, we also add a SHARE*ref* relationship between Resource Entities who share the same *attn* entities created in line 49 to facilitate the topological feature extraction from information provided by the original user attribute $attn \in Att_r$.

After the aforementioned four steps, an access control knowledge graph $\mathcal{G}$ is established for topological feature extraction.

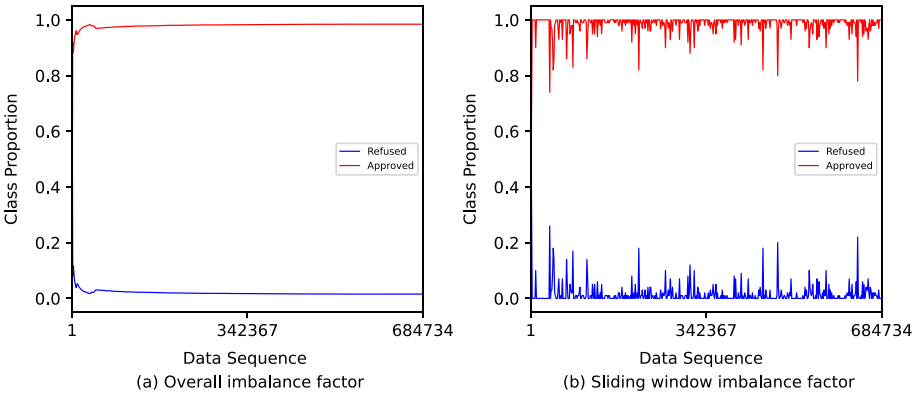

Fig. 2 Dynamic data imbalance statuses over time

Table 1 Dataset information

Attribute file	Attribute name	Type	Cardinality	Description
User	PERSON_ID	userID	36,063	ID of the user
	RESOURCE_LIST	Type 1	NP	list of resource ID that a users can possibly have access to
	MGR_ID	Type 2	3,207	manager ID
	DEPTNAME	Type 2	405	department description ID
	BUSINESS_TITLE	Type 2	4,979	title ID
	TITLE_DETAIL	Type 3	56	title description ID
	COMPANY	Type 3	49	company ID
	JOB_CODE	Type 3	13	job code ID
	JOB_FAMILY	Type 3	70	job family ID
	ROLLUP_1	Type 3	12	user grouping ID
	ROLLUP_2	Type 3	111	user grouping ID
	ROLLUP_3	Type 3	12	user grouping ID
Resource	RESOURCE_ID	resourceID	33,252	ID of the resource
	RESOURCE_TYPE	Type 3	3	group, system or host

3.3 Feature extraction for access control

To train the classifier $f_\Theta(\cdot)$ for access control, we use the log file containing access control requests and their corresponding verified decision (approval or refuse) to form labelled samples. Specifically, for each request from user u to resource r at time step t, denoted as $(u \rightarrow v)^{(t)}$, six sets of features can be exacted from the access control knowledge graph $\mathcal{G}$ constructed with Algorithm 1, namely, $x^{(t)}_{u_N}$, $x^{(t)}_{u_T}$, $x^{(t)}_{r_N}$, $x^{(t)}_{r_T}$, $x^{(t)}_{(u\rightarrow r)_N}$ and $x^{(t)}_{(u\rightarrow r)_T}$, as shown in Figure 1.

Among them, $x^{(t)}_{u_N}$, $x^{(t)}_{r_N}$ and $x^{(t)}_{(u\rightarrow r)_N}$ are the non-topological feature sets extracted from the User entity u, the Resource entity r and the existing relationships between them $u \rightarrow v$.

Table 2 Usecase of Algorithm 1

Step	Used information	Created entity	Created relationship	Created property
Step 1	PERSON_ID,	User entities,	none	u.userID,
	RESOURCE_ID	Resourc entities		r.resourceID
Step 2.1	RESOURCE_LIST	none	HAS_P_ACCESS	none
Step 2.2	MGR_ID,	Manager,	HAS_MANAGER,	m.managerID,
	DEPTNAME,	Department,	HAS_DEPT,	d.deptID,
	BUSINESS_TITLE	Title	HAS_TITLE	t.titleID
Step 2.3	TITLE_DETAIL,			u.titleDetail,
	COMPANY,			u.company,
	JOB_CODE,			u.jobCode,
	JOB_FAMILY,			u.jobFamily,
	ROLLUP_1,			u.rollup1,
	ROLLUP_2,			u.rollup2,
	ROLLUP_3	none	none	u.rollup3
Step 3.1	none	none	none	none
Step 3.2	none	none	none	none
Step 3.3	RESOURCE_TYPE	none	none	r.resourceType
Step 4	HAS_P_ACCESS,		SHARE_P_USER,	
	HAS_MANAGER,		SHARE_MANAGER,	
	HAS_DEPT,		SHARE_DEPT,	
	HAS_TITLE	none	SHARE_TITLE	none

These three non-topological feature sets can be exported from the properties of entities u, r and relationships $u \rightarrow r$.

By contrast, $x^{(t)}_{u_T}$, $x^{(t)}_{r_T}$ and $x^{(t)}_{(u\rightarrow r)_T}$ are the corresponding topological feature sets. $x^{(t)}_{u_T}$ is extracted from a subgraph of the constructed access control knowledge graph $\mathcal{G}$ which consists of User entities and relationships between them. Similarly, $x^{(t)}_{r_T}$ is extracted from a subgraph containing Resource entities and relationships between them. Both $x^{(t)}_{u_T}$ and $x^{(t)}_{r_T}$ are the topological features extracted to present the entities. The extracted topological features include but are not limited to (1) centrality scores which determine the importance of distinct nodes in a graph, such as page rank scores and betweenness scores; (2) community detection scores which indicate how groups of nodes are clustered or partitioned, as well as their tendency to strengthen or break apart, such as the weakly connected component id and triangle count of an entity. $x^{(t)}_{(u\rightarrow r)_T}$ is extracted from a subgraph containing User entities, Resource entities and relationships between User and Resource entities. $x^{(t)}_{(u\rightarrow r)_T}$ is used to present the closeness of entities u and r based on the graph with relationships between u and t. The possible features of $x^{(t)}_{(u\rightarrow r)_T}$ include but are not limited to Adamic Adar scores and common neighbours.

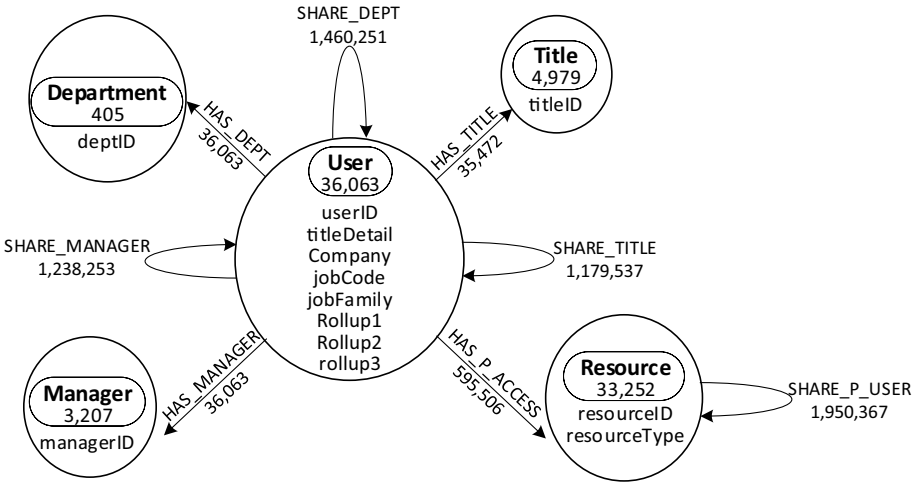

Fig. 3 The data model of the constructed access control knowledge graph $\mathcal{G}$

4 Experiment results

This section introduces a real-world access control dataset. Then provides a use case of the access control knowledge graph construction algorithm described in Algorithm 1 on this dataset. Finally, we compare the access control performance on topological features extracted from the established knowledge graph and non-topological features. Results show that the proposed knowledge graph empowered method outperforms non-topological methods in both offline and online scenarios.

4.1 Dataset

The experiments of this work are conducted on an open-source real-world Amazon employee access dataset[1]. The dataset contains a file listing all user and resource attributes and a time-series log file containing 684,374 user to resource access control requests and the corresponding permission records. The dataset is extremely imbalanced with 10,911 (1.59%) access rejection and 673,463 (98.41%) access approval. The dynamic data imbalance status is shown in Figure 2. Subplot (a) shows the overall imbalance factor of the refused requests and approved requests. The overall imbalance factor of the refused requests gradually converges to 1.59% after a fluctuation at the early stages and the approved requests converge to 98.41%. Subplot (b) shows the sliding window imbalance factor [40] of the two classes when the sliding window size is set to 100.

Table 1 lists the basic information of the attribute file. means Not Applicable As shown in Table 1, there are 36,063 unique users and 33,252 unique resources. We set the cardinality threshold θ=300. Based on the three attribute types defined in Section 3.2.1, the corresponding attribute type is listed in the Type column. The type information can be used to guide the knowledge graph construction, which is described in Section 4.2. For the Type 1 attribute, the cardinality is not applicable (NP).

[1] http://archive.ics.uci.edu/ml/datasets/Amazon+Access+Samples

Table 3 Feature extraction details

Feature	Source	Features
x_{u_N}	User entity	u.userID, u.titleDetail, u.Company, u.jobCode, u.jobFamily, u.Rollup1, u.Rollup2, u.rollup3
x_{u_T}	subgraphs: (u1:User)-[rel:SHARE_MANAGER]-(u2:User), (u1:User)-[rel:SHARE_DEPT]-(u2:User), (u1:User)-[rel:SHARE_TITLE]-(u2:User)	PageRank, ArticleRank, Betweenness, Degree, Closeness, Louvain, HarmonicCloseness, LabelPropagation, WCC, triangleCount, Modularity
x_{r_N}	Resource entity	r.resourceID, r.resourceType
x_{r_T}	subgraph: (r1:Resource)-[rel:SHARE_P_USER]-(u2:User)	PageRank, ArticleRank, Betweenness, Degree, Closeness, Louvain, HarmonicCloseness, LabelPropagation, WCC, triangleCount, Modularity
$x_{(u \to r)_N}$	relationship: SHARE_P_USER	none
$x_{(u \to r)_T}$	subgraph: (u:User)-[rel:HAS_P_ACCESS]-(r:Resource)	preferentialAttachment, totalNeighbor

4.2 Access control knowledge graph construction

According to the dataset introduced in Section 4.1, we construct an access control knowledge graph following the steps in Algorithm 1. The main process and intermediate knowledge graph construction results are summarised in Table 2.

In Step 1, based on the user attribute PERSIN_ID, 36,063 User entities with a userID property are created and 33,252 Resource entities with a resourceID property are created using the RESOURCE_ID attribute.

In Step 2, more entities, relationships and properties are created based on the three types of user attributes. Specifically, in Step 2.1, a HAS_P_ACCESS relationship is created between User and Resource entities to show the possibility of access requests based on Type 1 RESOURCE_LIST attributes. In step 2.2, three Type 2 attributes, namely, MGR_ID, DEPTNAME and BUSINESS_TITLE, are used to create three types of entities. The newly created entity labels are Manager, Department and Title respectively. The attribute values are added as the entity properties as shown in Table 2. The m.managerID means we created a manageID property for Manager entities. Similarly, d.deptID and t.titleID are properties added to Department and Title entities respectively. Furthermore, a HAS_MANAGER relationship is created between User and Manager entities. Similarly, a HAS_DEPT and a HAS_TITLE relationship is also created between User and Department entities as well as User and Title entities. Finally, in Step 2.3, 7 Type 3 attributes are added as the properties of User Entities, as shown in Table 2.

Table 4 Performance comparison of different classifiers on offline scenario

Classifier	Feature	Acc(%)	Macro average				Class 0			
			Pre(%)	Rec(%)	F1(%)	Δ F1	Pre(%)	Rec(%)	F1(%)	Δ F1
GNB	Nontopo	53.56	57.10	53.02	45.43		52.43	90.87	66.49	
	Topo	60.51	60.71	60.59	60.43	↑33.01%	62.57	55.09	58.59	↓11.89%
LR	Nontopo	61.04	61.14	61.09	61.02		62.61	57.55	59.97	
	Topo	62.89	63.08	62.96	62.83	↑2.97%	65.05	57.97	61.31	↑2.23%
NN	Nontopo	60.63	61.07	60.75	60.38		63.68	52.04	57.27	
	Topo	61.46	61.65	61.53	61.39	↑1.66%	63.51	56.39	59.74	↑4.31%
RF	Nontopo	70.65	70.74	70.69	70.64		72.52	67.81	70.08	
	Topo	73.61	73.64	73.63	73.61	↑4.21%	74.86	72.22	73.51	↑4.89%
SVM	Nontopo	61.07	61.18	61.13	61.04		62.69	57.38	59.92	
	Topo	50.51	47.95	49.83	35.21	↓42.31%	50.62	97.71	66.69	↑11.31%

In Step 3, only a Type 3 attribute, RESOURCE_TYPE, is available and we add a resourceType property to Resource entities.

In Step 4, a SHARE_P_USER relationship is created between two Resource entities who have HAS_P_ACCESS relationship with the same User Entity. Similarly, three relationships, namely, SHARE_MANAGER, SHARE_DEPT and SHARE_TITLE, are created respectively between two User entities who have HAS_MANAGER/ HAS_DEPT/ HAS_TITLE relationships with the same Manager/ Department/ Title entity.

Finally, an access control knowledge graph $\mathcal{G}$ is constructed based on the Amazon access control dataset. The data model (schema) of $\mathcal{G}$ is illustrated as Figure 3. A circle presents a type of entity with a bold label inside. Below the label is the total number of entities with that label. The properties of the corresponding entities are also listed inside the circle. An arrow represents a directed relationship. We also specify the relationship type and the total number of relationships along the arrow.

4.3 Feature extraction

We implement the access control knowledge graph $\mathcal{G}$ on the Neo4j [2] graph data platform, which provides a convenient way for both topological and non-topological feature extraction from existing entities, relationships and subgraphs of a knowledge graph. The topological features adopted in this work are implemented with the Neo4j Graph Data Science Library [3].

For an access control request from a user u to a resource r, six feature sets, i.e., x_{u_N}, x_{u_T}, x_{r_N}, x_{r_T}, $x_{(u\to r)_N}$, $x_{(u\to r)_T}$ are extracted from the User entities, Resource entities and their relationships. Table 3 presents our feature extraction strategies in detail. As shown in the first row, x_{u_N} is extracted from the properties of the User entity u. x_{u_T} presents the topological features extracted from subgraphs containing User entities and the relationships between them, as shown in the second row. The listed features are extracted to represent the importance or connectivity characteristics in the subgraphs. Similarly, x_{r_N} and x_{r_T} are

[2] https://neo4j.com/

[3] https://neo4j.com/docs/graph-data-science/current/algorithms/

Table 5 Offline learning performance comparison on different data imbalance statuses

Class 0	Feature	Acc(%)	Macro average				Class 0			
			Pre(%)	Rec(%)	F1(%)	Δ F1	Pre(%)	Rec(%)	F1(%)	Δ F1
30%	Nontopo	76.87	73.72	66.87	68.41		68.75	41.90	52.07	
	Topo	78.57	75.76	69.89	71.56	↑4.60%	71.02	48.21	57.43	↑10.30%
10%	Nontopo	89.41	67.70	59.16	61.49		43.63	21.38	28.69	
	Topo	89.56	68.58	59.82	62.28	↑1.30%	45.26	22.67	30.21	↑5.28%
1.59%	Nontopo	98.01	53.29	51.08	51.48		8.10	2.63	3.97	
	Topo	97.98	54.24	51.55	52.15	↑1.29%	9.99	3.62	5.32	↑33.83%

the non-topological and topological features of Resource entity r. In this usecase, no properties are added to the relationship SHARE_P_USER, therefore, $x_{(u \to r)_N}$ is none. We select two link prediction topological features, i.e., preferentialAttachment [4] and totalNeighbor [5], to present $x_{(u \to r)_T}$ in this work.

4.4 Evaluation metrics

For classification problems, based on the true labels and the predicted results, samples in the test set can be classified as:

- True Positives (TP): the correctly predicted positive samples, which means both the true label and the predicted result of these samples are a positive class.
- True Negative (TN): the correctly predicted negative samples, which means both the true label and the predicted result of these samples are a negative class.
- False Positives (FP): the incorrectly predicted positive samples, which means the true label is negative, but the predictive result is positive.
- False Negatives (FN): the incorrectly predicted negative samples, which means the true label is positive, but the predictive result is negative.

We evaluate the classification performance of the proposed approach on four well-known metrics, namely, accuracy, precision, recall and F1 score. Their definition are given as (8) - (11).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}. \tag{8}$$

$$Precision = \frac{TP}{TP + FP}. \tag{9}$$

$$Recall = \frac{TP}{TP + FN}. \tag{10}$$

[4] https://neo4j.com/docs/graph-data-science/current/alpha-algorithms/preferential-attachment/
[5] https://neo4j.com/docs/graph-data-science/current/alpha-algorithms/total-neighbors/

Table 6 Overall performance comparison of online learning

Class 0	Feature	Acc(%)	Macro average				Class 0			
			Pre(%)	Rec(%)	F1(%)	Δ F1	Pre(%)	Rec(%)	F1(%)	Δ F1
30%	Nontopo	73.41	67.72	61.06	61.78		59.51	30.94	40.71	
	Topo	73.68	67.97	62.31	63.25	↑2.37%	59.26	34.58	43.68	↑7.28%
10%	Nontopo	90.50	74.46	55.09	56.76		57.80	11.06	18.57	
	Topo	90.33	71.83	56.18	58.29	↑2.70%	52.32	13.72	21.74	↑17.10%
1.59%	Nontopo	98.39	64.77	51.70	52.76		31.03	3.53	6.33	
	Topo	98.38	65.14	52.17	53.53	↑1.45%	31.75	4.49	7.87	↑24.31%

$$F1Score = 2 \times \frac{Recall \times Precision}{Recall + Precision}. \tag{11}$$

4.5 Offline learning performance comparison

To verify the effectiveness of the proposed knowledge graph empowered framework, we compared the access control decision-making performance of using topological features extracted from established knowledge graph and non-topological features from original user and resource attributes on both online and offline scenarios.

Firstly, we verify the performance improvement of topological features on five different classifiers, i.e., naive Bayes (GNB), logistic regression (LR), neural network (NN), random forest(RF), and support vector machine (SVM). We use the scikit-learn [6] library to implement these classifiers. Considering the importance of class 0 (request rejection) in access control problems, results on both class 0 and the macro average on class 1 and class 0 are reported in Table 4. The results in Table 4 are conducted on a balanced dataset consisting of all negative samples of the original Amazon dataset introduced in Section 4.1 and the same number of positive samples randomly selected from the original dataset. Both the negative and positive samples keep the same order as the original dataset.

Although accuracy (Acc) is the most-used metric for evaluating classification models, it only works on balanced datasets. For severe imbalanced datasets, the results of accuracy (Acc) can be misleading and unreliable. Since the F1 score is an evaluation metrics combining two competing metrics, i.e. precision (Pre) and recall (Rec), we mainly discuss the F1 score when comparing the performance of topological and non-topological features. Δ F1 is the growth rate between the F1 score achieved on topological and non-topological features, calculated by Equation (12).

$$\Delta \text{F1} = \frac{F1_{\text{Topo}} - F1_{\text{Nontopo}}}{F1_{\text{Nontopo}}} \times 100\%, (t > 0). \tag{12}$$

As shown in Table 4, RF classifier achieves the best performance on all metrics with topological features. Using topological features extracted from the access control

[6] https://scikit-learn.org/stable/

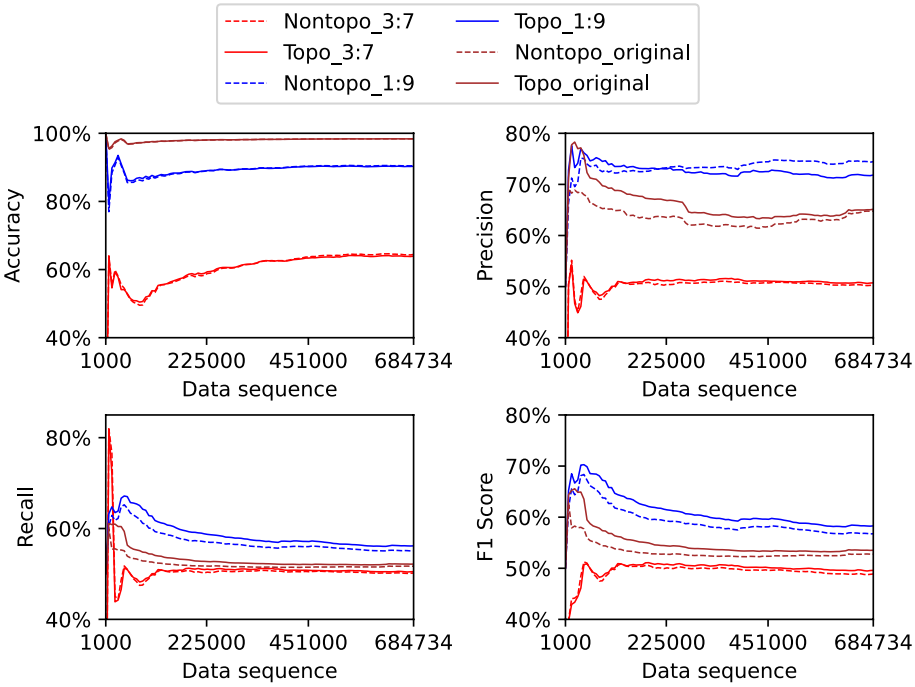

Fig. 4 The real-time macro average performance comparison of online learning

knowledge graph increases the F1 score on class 0 from 70.08% to 73.51%, which achieves an increase of 4.89%. Actually, an improvement of 4.21% also achieved on macro average F1 score by using topological features. The performance on NN and LR classifiers are also boosted on both macro average and class 0 with topological features. However, for the GNB classifier, the macro average F1 score is improved from 45.43% to 60.43% with a cost of the decrease of F1 score of class 0 from 66.49% to 58.59%. It means that topological features increase the performance on class 1 but decrease on class 0 when using the GNB classifier. By contrast, the SVM classifier increases the F1 score of class 0 from 59.92% to 66.69% but the macro average f1 score decreases from 61.04% to 35.21%. Generally speaking, it is fair to say that the topological feature can improve access control performance in the offline learning scenario.

To further verify the improvement effectiveness of topology features in different data imbalance statuses, we use an RF classifier, which performs the best in Table 4, to conduct experiments on different class proportions in an offline scenario, as shown in Table 5. Consistent with Table 4, topological features can improve the access control performance of both macro average and the minority class (class 0) on different degrees of imbalanced datasets. However, with the increase of data imbalance, the performance of the algorithm gradually deteriorates, but the results are still much better than a random decision. Specifically, topological features improve the macro average f1 score by 4.60%, 1.30% and 1.29% respectively when the class 0 accounts for 30%, 10%, 1.59% (the original dataset) in the dataset. Furthermore, topological features

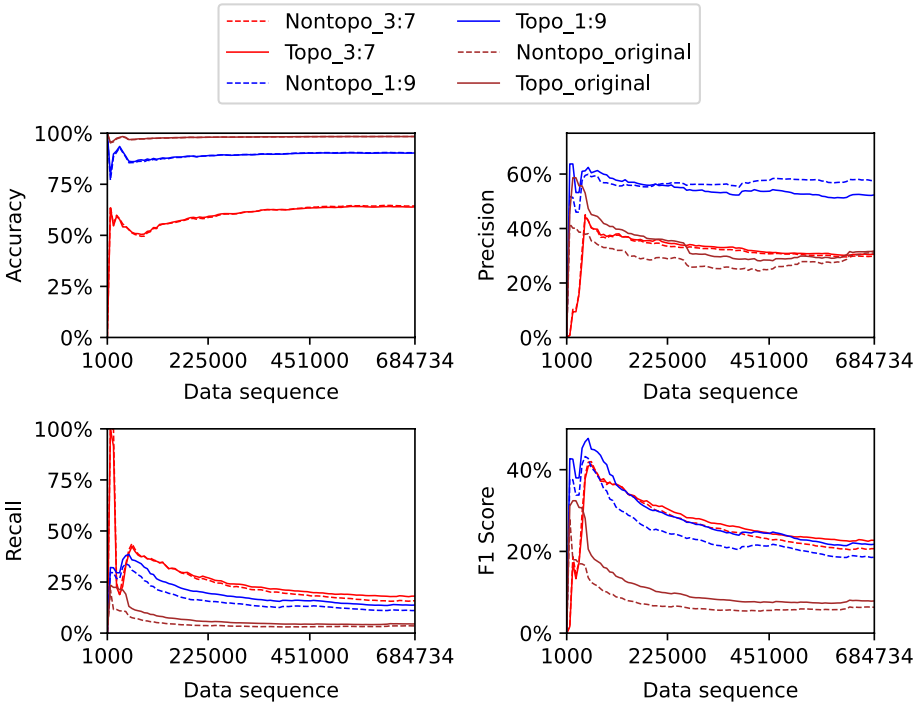

Fig. 5 The real-time performance comparison of online learning on class 0

are superior in improving the performance on class 0, which records an increase of 10.30%, 5.28% and 33.83% accordingly.

4.6 Online learning performance comparison

We also conduct online learning experiments on different degrees of imbalance statuses to verify the effectiveness of topology features in improving access control performance. Table 6 shows the overall performance comparison results. The time step size is set as 1/1000 of the dataset size. Topological features improve the macro average f1 score by 2.37%, 2.7% and 1.45% respectively when the class 0 accounts for 30%, 10%, 1.59% (the original dataset) in the dataset. In particular, topological features are superior in improving the performance on class 0, which records an increase of 7.28%, 17.10% and 24.31% accordingly.

Figure 4 shows the real-time macro average performance comparison of online learning. The red lines show access control performance comparison when using topological and non-topological features on a dataset with class 0: class 1 = 3:7. Similarly, the brown lines and blue lines present the results of the original dataset and a dataset with class 0: class 1 = 1:9. Figure 4 demonstrates that topological features can improve the overall f1 score without decreasing the accuracy.

Similarly, Figure 5 shows the real-time performance comparison of online learning on class 0 (the minority class). Though the trends are the same with Figure 4, the degree of improvements are larger in Figure 5 .

4.7 Discussion

Results shown in Tables 5 and 6 demonstrated the effectiveness of topological features in improving the access control performance in both offline and online scenarios. However, for privacy and security reasons, the Amazon access control dataset only provides 12 categorical user attributes and 2 resource attributes. These attributes use ID numbers to distinguish different values to prevent sensitive data leakage. It is very challenging to achieve high predictive performance without more text attributes to provide rich semantic information for mining. Therefore, the overall performance and the minority class performance is still unsatisfactory.

In fact, the problem of data insufficiency, especially the lack of attributes information, is common for access control. ABAC rule mining algorithms also suffer from severe overall performance deficiency caused by the poor quality of available real-world access control datasets. For example, the work in [9] proposed an iterative rule mining algorithm, named Rhapsody, to automatically mine ABAC rules from sparse logs and prevent over-permissiveness. They reported the F1 scores of five ABAC rule-based algorithms including Rhapsody on the same Amazon dataset with us. The range of the reported F1 scores is from 0.01 to 0.35, which is equivalent to our method. However, they only choose the top eight most requested resources and their corresponding requests to form eight instances for algorithm evaluation instead of evaluating the algorithm on the whole log file as we do. Therefore, the generalization performance of their algorithm is not guaranteed.

5 Conclusion

To better encode high cardinality categorical user and resource attributes and improve the machine learning based access control performance, we proposed a knowledge graph empowered online learning framework for access control decision-making. Through transferring tabular user and resource attributes into a comprehensive knowledge graph, we extracted topological features from the established knowledge graph to represent uses and resources. Experimental results show that topological features outperform non-topological features encoded by binary encoding method in both online and offline settings.

In future, we will find datasets with rich user and resource attributes to further verify the proposed knowledge graph based online leaning framework. Furthermore, more deep learning based embedding algorithms will be explored to extract high level entity and relationship features to further improve the access control performance.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions

Declarations

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

References

1. Abbas, K., Abbasi, A., Dong, S., Niu, L., Yu, L., Chen, B., Cai, S.M., Hasan, Q.: Application of network link prediction in drug discovery. BMC Bioinformatics **22**(1), 1–21 (2021). https://doi.org/10.1186/s12859-021-04082-y
2. Adamic, L.A., Adar, E.: Friends and neighbors on the web. Social Networks **25**(3), 211–230 (2003). https://doi.org/10.1016/s0378-8733(03)00009-1
3. Barabási, A.L., Albert, R., Jeong, H.: Scale-free characteristics of random networks: the topology of the world-wide web. Physica A: Statistical Mechanics and Its Applications **281**(1-4), 69–77 (2000). https://doi.org/10.1016/s0378-4371(00)00018-2
4. Bertino, E., Bonatti, P.A., Ferrari, E.: Trbac: a temporal role-based access control model. In: Proceedings of the fifth ACM Workshop on Role-Based Access Control. pp. 21–30. https://doi.org/10.1145/344287.344298 (2000)
5. Brandes, U., Pich, C.: Centrality estimation in large networks. International Journal of Bifurcation and Chaos **17**(07), 2303–2318 (2007). https://doi.org/10.1142/s0218127407018403
6. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer networks and ISDN systems **30**(1-7), 107–117 (1998). https://doi.org/10.1016/s0169-7552(98)00110-x
7. Chen, Z.G., hui Zhan, Z., Wang, H., Zhang, J.: Distributed individuals for multiple peaks: a novel differential evolution for multimodal optimization problems. IEEE Trans. Evol. Comput. **24**, 708–719 (2020). https://doi.org/10.1109/tevc.2019.2944180
8. Cheng, K., Wang, L., Shen, Y., Wang, H., Wang, Y., Jiang, X., Zhong, H.: Secure *k* k-nn query on encrypted cloud data with multiple keys. IEEE Trans. Big Data **7**, 689–702 (2021). https://doi.org/10.1109/tbdata.2017.2707552
9. Cotrini, C., Weghorn, T., Basin, D.: Mining abac rules from sparse logs. In: 2018 IEEE European Symposium on Security and Privacy (Euros&P). pp. 31–46. IEEE. https://doi.org/10.1109/eurosp.2018.00011 (2018)
10. Daminelli, S., Thomas, J.M., Durán, C., Cannistraci, C.V.: Common neighbours and the local-community-paradigm for topological link prediction in bipartite networks. New Journal of Physics **17**(11), 113037 (2015). https://doi.org/10.1088/1367-2630/17/11/113037
11. Dong, L., Li, Y., Yin, H., Le, H., Rui, M.: The algorithm of link prediction on social network. Math. Probl. Eng. 2013. https://doi.org/10.1155/2013/125123 (2013)
12. Ge, Y.F., Cao, J., Wang, H., Zhang, Y., Chen, Z.: Distributed differential evolution for anonymity-driven vertical fragmentation in outsourced data storage. In: International Conference on Web Information Systems Engineering. pp. 213–226. Springer. https://doi.org/10.1007/978-3-030-62008-0_15 (2020)
13. Ge, Y.F., Orlowska, M., Cao, J., Wang, H., Zhang, Y.: Knowledge transfer-based distributed differential evolution for dynamic database fragmentation. Knowl.-Based Syst. **229**, 107325 (2021). https://doi.org/10.1016/j.knosys.2021.107325
14. Ge, Y.F., Orlowska, M., Cao, J., Wang, H., Zhang, Y.: Mdde: multitasking distributed differential evolution for privacy-preserving database fragmentation. The VLDB Journal, 1–19. https://doi.org/10.1007/s00778-021-00718-w (2022)
15. Ge, Y.F., Yu, W.J., Cao, J., Wang, H., Zhan, Z.H., Zhang, Y., Zhang, J.: Distributed memetic algorithm for outsourced database fragmentation. IEEE Trans. Cybern. **51**(10), 4808–4821 (2020). https://doi.org/10.1109/tcyb.2020.3027962
16. Hu, H., Li, J., Wang, H., Daggard, G.: Combined gene selection methods for microarray data analysis. In: Knowledge-Based Intelligent Information and Engineering Systems. pp. 976–983. Springer, Berlin. https://doi.org/10.1007/0-387-23077-7_16 (2006)
17. Jiang, H., Zhou, R., Zhang, L., Wang, H., Zhang, Y.: Sentence level topic models for associated topics extraction. https://doi.org/10.1007/s11280-018-0639-1, vol. 22, pp 2545–2560 (2019)
18. Kabir, E., Mahmood, A., Wang, H., Mustafa, A.: Microaggregation sorting framework for k-anonymity statistical disclosure control in cloud computing. IEEE Transactions on Cloud Computing PP 1–1. https://doi.org/10.1109/TCC.2015.2469649 (2015)

19. Li, J., Willett, P.: Articlerank: a pagerank-based alternative to numbers of citations for analysing citation networks. In: Aslib Proceedings. Emerald Group Publishing Limited. https://doi.org/10.1108/00012530911005544 (2009)
20. Liu, W., jiao Gong, Y., neng Chen, W., Liu, Z., Wang, H., Zhang, J.: Coordinated charging scheduling of electric vehicles: a mixed-variable differential evolution approach. IEEE Trans. Intell. Transp. Syst. **21**, 5094–5109 (2020). https://doi.org/10.1109/tits.2019.2948596
21. Marchiori, M., Latora, V.: Harmony in the small-world. Physica A: Statistical Mechanics and its Applications **285**(3-4), 539–546 (2000). https://doi.org/10.1016/s0378-4371(00)00311-3
22. Paci, F., Squicciarini, A., Zannone, N.: Survey on access control for community-centered collaborative systems. ACM Computing Surveys (CSUR) **51**(1), 1–38 (2018). https://doi.org/10.1145/3146025
23. Rasool, R.U., Ashraf, U., Ahmed, K., Wang, H., Rafique, W., Anwar, Z.: Cyberpulse: a machine learning based link flooding attack mitigation system for software defined networks. IEEE Access **7**, 34885–34899 (2019). https://doi.org/10.1109/ACCESS.2019.2904236
24. Sandhu, R.S.: Role-Based Access Control. In: Advances in Computers, vol. 46, pp. 237–286. Elsevier. https://doi.org/10.1016/s0065-2458(08)60206-5 (1998)
25. Servos, D., Osborn, S.L.: Current research and open problems in attribute-based access control. ACM Computing Surveys (CSUR) **49**(4), 1–45 (2017). https://doi.org/10.1145/3007204
26. Srilatha, P., Manjula, R.: Structural similarity based link prediction in social networks using firefly algorithm. In: 2017 International Conference on Smart Technologies for Smart Nation (SmartTechCon). IEEE. https://doi.org/10.1109/smarttechcon.2017.8358434 (2017)
27. Sun, X., Wang, H., Li, J., Pei, J.: Publishing anonymous survey rating data. Data Min. Knowl. Discov **23**, 379–406 (2011). https://doi.org/10.1007/s10618-010-0208-4
28. Sun, X., Wang, H., Plank, A.: An efficient hash-based algorithm for minimal k-anonymity. Proc. Thirty-First Aust. Conf. Comp. Sci. **74**, 101–107 (2008). https://doi.org/10.1145/1378279.1378297
29. Verizon: Data Breach Investigations Report. Tech. rep., Verizon. https://enterprise.verizon.com/resources/reports/2020-data-breach-investigations-report.pdf (2020)
30. Vimalachandran, P., Liu, H., Lin, Y., Ji, K., Wang, H., Zhang, Y.: Improving accessibility of the australian my health records while preserving privacy and security of the system. Health Information Science and Systems **8**(1), 1–9 (2020). https://doi.org/10.1007/s13755-020-00126-4
31. Wang, H., Cao, J., Zhang, Y.: Ticket-based service access scheme for mobile users. Australian Computer Science Communications pp 285–292. https://doi.org/10.1145/563857.563834 (2002)
32. Wang, H., Cao, J., Zhang, Y.: A flexible payment scheme and its role-based access control. Knowl. Data Eng. IEEE Trans. **17**, 425–436 (2005). https://doi.org/10.1109/TKDE.2005.35
33. Wang, H., Sun, L.: Trust-involved access control in collaborative open social networks. 2010 Fourth International Conference on Network and System Security pp 239–246. https://doi.org/10.1109/nss.2010.13 (2010)
34. Wang, H., Sun, L., Bertino, E.: Building access control policy model for privacy preserving and testing policy conflicting problems. J. Comput. Syst. Sci. **80**, 1493–1503 (2014). https://doi.org/10.1007/978-3-030-31729-4_11
35. Wang, H., Wang, Y., Taleb, T., Jiang, X.: Editorial: Special issue on security and privacy in network computing. World Wide Web 23. https://doi.org/10.1007/s11280-019-00704-x (2019)
36. Wang, H., Zhang, Y., Cao, J.: Effective collaboration with information sharing in virtual universities. IEEE Trans. Knowl. Data Eng. **21**, 840–853 (2009). https://doi.org/10.1109/TKDE.2008.132
37. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 28. https://ojs.aaai.org/index.php/AAAI/article/view/8870 (2014)
38. Yin, J., Tang, M., Cao, J., Wang, H.: Apply transfer learning to cybersecurity: Predicting exploitability of vulnerabilities by description. Knowl.-Based Syst. **210**, 106529 (2020). https://doi.org/10.1016/j.knosys.2020.106529
39. Yin, J., Tang, M., Cao, J., Wang, H., You, M.: A real-time dynamic concept adaptive learning algorithm for exploitability prediction. Neurocomputing **472**, 252–265 (2022). https://doi.org/10.1016/j.neucom.2021.01.144
40. Yin, J., Tang, M., Cao, J., Wang, H., You, M., Lin, Y.: Adaptive online learning for vulnerability exploitation time prediction. In: International Conference on Web Information Systems Engineering. pp. 252–266. Springer. https://doi.org/10.1007/978-3-030-62008-0_18 (2020)
41. Yin, J., Tang, M., Cao, J., Wang, H., You, M., Lin, Y.: Vulnerability exploitation time prediction: an integrated framework for dynamic imbalanced learning. Word Wide Web **1**(1), 1–23 (2021). https://doi.org/10.1007/s11280-021-00909-z
42. You, M., Yin, J., Wang, H., Cao, J., Miao, Y.: A minority class boosted framework for adaptive access control decision-making. In: International Conference on Web Information Systems Engineering. pp. 143–157. Springer. https://doi.org/10.1007/978-3-030-90888-1_12 (2021)
43. Zhang, J., Li, H., Liu, X., Luo, Y., Chen, F., Wang, H., Chang, L.: On efficient and robust anonymization for privacy protection on massive streaming categorical information. IEEE Transactions on Dependable and Secure Computing **14**, 507–520 (2017). https://doi.org/10.1109/tdsc.2015.2483503

44. Zhou, T., Lü, L., Zhang, Y.C.: Predicting missing links via local information. The European Physical Journal B **71**(4), 623–630 (2009). https://doi.org/10.1140/epjb/e2009-00335-8