IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 6, 2020, accepted March 1, 2020, date of publication March 26, 2020, date of current version April 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2983146

Extracting Actionable Knowledge From Social Networks Using Structural Features

NASRIN KALANAT, EYNOLLAH KHANJARI, AND ALIREZA KHANSHAN
School of Computer Engineering, Iran University of Science and Technology, Tehran 16846-13114, Iran
Corresponding author: Eynollah Khanjari (khanjari@iust.ac.ir)

ABSTRACT In conventional data mining methods, the output is either a description of input data or a prediction of unseen data. But the real-world problems usually require interventions in order to alter the current data specifications towards a desirable goal. Actionable knowledge discovery is a field of study specifically developed for this matter. Existing methods rarely tackled the problem of extracting actionable knowledge from social networks. Moreover, due to the dependencies among the underlying network data, extracted actions should be evaluated since the changes suggested by the actions may not be described by the model constructed so far. This enforces the refinement of the model to preserve the quality of extracted actions. In this paper we propose a new method for action mining which incorporates an action evaluation process overcoming the mentioned problem while focusing specifically on social network data. Such data contains valuable information based on the links inside the network where a change in some feature values may result in a chain of changes in others due to the dependencies conveyed by the links in the network. We use a state-of-the-art structural feature extraction method to capture the information of the dependencies inside the network. Our proposed method iteratively updates structural features which are incorporated in the action extraction process. In this process, we thoroughly examine the effects of the application of actions by discovering the impact of possible changes in the network. We call this phenomenon ''change propagation''. According to our experimentations, our method outperforms the state-of-the-art methods in terms of action effectiveness and reliability with comparable efficiency.

INDEX TERMS Social networks mining, action mining, actionable knowledge discovery, structural features, change propagation, change-awareness.

I. INTRODUCTION

Data mining provides an array of methods to extract knowledge from raw data. This knowledge is a great help to solve today's world problems as the volume of digital data grows tirelessly. A variety of predictive or descriptive models can be learned from such data using mining methods, but in many cases, the need exceeds these outputs and a real-world problem cannot solely rely on the models derived from data. Actionable knowledge discovery (AKD) provides techniques and tools in order to facilitate automated decision making which is the area of expert systems. AKD exploits machine learning methods for discovering knowledge and patterns so that not only they are informative and/or predictive but also applicable in the sense that actions fulfil business expectations. It has been developed specifically to address this problem and suggests solutions to find appropriate interventions. In most existing AKD approaches, classification or association rules are extracted then they are further processed to result in a set of potential change suggestions in the feature space. These change suggestions are called actions. An action set is a set of such potential changes (actions) which results in a more desirable state if applied to input data which meets business needs. Such a process in which a set of suggestions are made by exploring the space of input data is called action extraction. There have been multiple applications for action mining such as in customer relationship management, healthcare, and e-commerce. The common property of all these methods is their input data which is in tabular format meaning the features are assumed independent. However many real-world applications rely on the dependencies between objects such as in social networks where the dependencies among the individuals (nodes) are essential. Applying the isolated schema of the conventional methods in datasets

The associate editor coordinating the review of this manuscript and approving it for publication was Yonghong Peng.

where objects are significantly connected is a setback as a lot of valuable information will be lost; especially in social networks and graphs in general. To the best of our knowledge, action extraction using such data has been investigated less than other areas of these studies. Like many data mining applications, in social networks, meaningful actions are needed to help in decision-making. Actions would suggest some potential changes in the space of the input network. For a given model (constructed based on original network data), if such changes are applied over the network, the model could fulfill some business requirements e.g. a change in the class label of a given user. For example, an action may suggest that a change in the impact factor of a journal could turn its class to top-ranked journals class. To this end, the journal's owner may decide to offer promotions (or any other domain-specific decision) to top-ranked authors if they submit their research works to the journal in order to increase the journal's IF.

Moreover, the application of actions may alter the structure of some areas of the network. More precisely, a change in an object's feature value may result in a chain of changes in others because of the dependencies inside the network. In other words, the application of actions changes some feature values where such changes could be propagated through the network. The change propagations may in turn affect the quality of the model constructed so far, i.e. the previously constructed model may be no longer valid regarding the new data space. Therefore, it is needed to rebuild the model and extract actions based on the new feature values.

In this paper, we utilize the underlying information based on the dependencies inside the network data. Our goal, on one hand, is to break the barriers of the current state-of-the-art methods and utilize the relationships among objects to extract actions. On the other hand, we provide a new method to better investigate the impact of actions inside the input data to ultimately extract more effective and reliable actions. The first mentioned goal can be handled by extending the techniques to extract structural features, and for the second one, we propose a novel change-aware approach which incorporates the effect of the changes in different areas of the network in the action extraction process by evaluating the quality of the actions and rebuilding the model if needed. It should be noted that existing action extraction approaches commonly use tabular (or transactional) data sets in which objects are assumed independent from each other. However, the structural nature of social networks could not be ignored in action extraction process.

In the context of social networks, the data involves nodes (actors) and links (relationships), where each node has a set of features independent of others; such as age, gender, education, etc. These are called node features. There is also another type of feature derived from links inside the network corresponding to the node features, which is usually the aggregation of labels of a node's neighborhood. These are called link features [1]. In our method, we utilize another type of features called structural features. These features are solely built according to the structure of the graph-as opposed to linking features where their construction relies on node features. Degree, ego-net degree and ego-net out-degree are examples of such features [2]. There are a number of methods to extract structural features and their ability to improve different mining tasks has already been proven [3]. The important aspect in our method is that the features used in the mining process are structural, where their values could be altered by applying the potential changes (suggested by the action) in the values of features of an intended object.

More precisely, in the context of social networks, a structural change in a part of a network such as addition or deletion of an edge may affect the rest of the network and may cause changes in structural features. We call this phenomenon ''change propagation''. Our method proposes a way to track and monitor these changes which ultimately complements our action evaluation process.

The contributions of the paper are as follows:

1) We devise an iterative process to extract effective actions from social network incorporating the structural nature of the network.
2) We extract structural features so that the potential changes in the value of such features are trackable thorough the network.
3) Our method extracts reliable actions by evaluating the underlying model against the changes suggested by the actions, and reconstructs the model if needed. That is, such actions are change-aware and more accurate.

We compare our method to Yang *et al.* [4] and Tolomei *et al.* [5], two of the widely used methods in the field. The experimental results show that our method outperforms these state-of-the-art methods in terms of action quality with comparable efficiency.

The rest of the paper is organized as follows. Section II provides a brief review of the related works in the domain, then we present the common literature and terminology of the field in Section III. In Section IV we comprehensively describe our proposed method followed by the experimental results and discussion in Section V. Finally the paper is concluded in Section VI.

II. RELATED WORK

Since the scientific contribution of our work is mainly focused on mining actions from social networks, in this section our review is restricted to the latest related researches in this scope, excluding a vast amount of bibliography which exists in the general field of social network analysis. Each research and its application will be described concisely. It is noteworthy that very few of the works in this field have studied social networks. Furthermore, to the best of our knowledge, change propagation has not been investigated in the researches in the action mining literature. Ras *et al.* in [6] made one of the first efforts to propose an inductive method for action mining. They defined the concept of action rules and proposed a method to extract them using pairs of classification rules. Afterward, in [7], [8] Kalanat *et al.* proposed a method to find profitable action rules.

In [9] Pothirattanachaikul *et al.* have proposed a method to determine the level of alternativeness between two actions inside a Q&A corpus. The goal is to suggest alternative actions to user queries to extract actions, the typical approach of using a model or classification rules is not used but instead, they choose a set of answers or candidate actions among the pair of questions and answers received in the initial phase. Then they suggest a ranked list of actions based on their alternativeness.

Subramani *et al.* proposed a novel framework in [10] to model and discover different types of domestic violence in the public domain and ultimately provide actionable knowledge. Their method uses topic detection to turn Twitter data into actionable knowledge. They use pattern mining techniques to identify the patterns then the MapReduce architecture is used on the mined patterns and finally only the terms more coherent to the topic are clustered.

In [11] Ranganathan *et al.* proposed a new efficient system, to generate meta-actions by implementing Specific Action Rule discovery based on Grabbing Strategy (SARGS) and applying it to Twitter data for semantic analysis.

Cui *et al.* used additive tree models (ATM) to extract actions in [12]. ATMs are typically not very well interpreted when extracting actions so in the paper a new framework is proposed to post-process such ATMs capable of proposing an actionable plan to modify each input with minimum cost to achieve the desired state. The main approach of this work is to formulate optimal action mining to an integer linear programming problem. In [13] Lu *et al.* proposed yet another approach to extract actions from ATMs. Here the optimal actionable planning (OAP) is targeted and defined for each ATM. Then a heuristic method is proposed to find the optimal solution from the state space graph. These methods used Crisp Decision Trees for mining actions. The sharp behavior of the methods results in unreal estimation about the profit of actions. In [14], [15] Kalanat *et al.* utilized Fuzzy Decision Tree (FDT) to overcome this problem.

In [16] Li *et al.* refer to the concepts of both actionable knowledge and causality and assert causality as a principle in data mining. In this paper, actionability is declared as an important property of knowledge. In order to carry out an action, the discovered knowledge has to imply causal relationships to be able to justify the occurrences inside the data under consideration. Having a set of simple data mining tools can be beneficial to discover causal relationships. Although these tools do not guarantee their finding to be absolutely causal but are capable of finding candidates excluding non-causal attributes.

Tzacheva *et al.* in [17] present a new method to extract actions. They redefine action as a set of value changes in some specific attributes to obtain a certain goal and use rules extracted by Apriori to extract actions.

In [18] Kuang *et al.* have worked on a hierarchical meta-action ordering strategy (used to represent action rules and their corresponding triggers in a tree-like structure) along with the effects of meta-nodes.

The method in [19] developed by Almardini *et al.* aims to find meta-actions by clustering patients using a graph-based model. This method is specifically designed to solve the prediction of patients' treatment path taken from readmission till the end of their treatment problem.

In [20] Su *et al.* proposed a method to extract actionable behavioral rules based on decision trees. Behavioral action mining is a rather new field in data mining in which the rules are used to give individuals explicit suggestions to change their behavior to achieve a certain goal. It needs conventional data mining tools to find frequent action sets in order to guarantee the applicability of the rules.

Kuang *et al.* in [21] proposed one of the main modules of HAMIS (recommender system designed for 34 industrial services companies). Their method introduces a module that is responsible for discovering meta-actions from a massive set of textual comments obtained from their customers during surveys about their satisfaction of the company's provided services.

The work in [22] has been developed by Touati *et al.* in which the evaluation of action rules is done by measuring benefits of meta-actions by two measures of probability and reliability of performance.

In [23] Shamsinejad *et al.* try to solve the problem that the correlations found do not necessarily imply causation. To address this, they proposed ICE-CREAM in which actions are extracted using inductive causation introduced by Pearl and Verma in [24]. They benefit from a causal structure obtained from data and use causal inference to extract actions based on causality.

The common characteristics of these researches are that they are mainly focused on either business or health domains. The works focused on social network data do not incorporate dependencies; hence, they fail to examine the change propagation defined earlier. Whereas our method focuses on social network data, incorporates dependencies in the network and propagates the potential changes to achieve more qualified actions.

Finally, the concept of action extraction from social network has been initially introduced in [25], [26] which presents a relatively generic framework for exploring the space of the input network data in order to extract cost-effective optimal actions incorporating structural nature of such data.

III. TERMINOLOGY

The concepts and definition of actionable knowledge discovery vary in different sources. We intend to redefine terms used in this paper for clarification and better understanding. For simplicity, through the paper we usually omit a notation D denoting an input data set in the space of possible data sets, which is obvious in the context.

Feature Vector: Assuming an object o with m attributes, its characteristics are defined as the vector $f_0 = [A_1(o), A_2(o), \ldots, A_m(o)]$ in which A_i is the i^{th} attribute of o and $A_i(o)$ is a value from its domain i.e. $A_i(o) \in Dom(A_i)$.

Feature Matrix: Assuming there are n objects, a feature matrix is a $n \times m$ matrix in which each row corresponds to the feature vector of an object. A derivative F_{ij} corresponds to j^{th} feature value of the object corresponding to the i^{th} row, where $1 \leq i \leq n$ and $1 \leq j \leq m$.

Social Network: We model a social network as the tuple $< G, F >$. $G = (V, E)$ is the underlying graph where V is the set of n nodes, each representative of an object (or actor) inside the network and E is the set of edges showing the links between actors. As an example, the edge (u, v) shows some interaction between objects u and v. Finally, F is the feature matrix we described it earlier.

We use terms 'object' and 'node' as well as ''actor'' interchangeably through the paper according to the context.

Class Label: Assuming A is the set of all attributes, one attribute L and a specific value $l \in Dom(L)$ is chosen to determine the state of the object according to business specifications/requirements. Given an object o, its state is defined desirable if $L(o) = l$, with its associated probability, and undesirable otherwise. The set of objects in undesirable state is denoted by O^-.

Action: For an intended object $o \in O^-$, an action is formally denoted by $\alpha = (A_i, a \rightarrow a')$ where $A_i \neq L$, a is the current value of attribute A_i for o and a' is the (new) value suggested by α, where a,$a' \in Dom(A_i)$ and $a \neq a'$.

Action Set: The result of an action mining process is a set of actions denoted by $\Gamma = \{\alpha_1, \ldots, \alpha_k\}$ where $k \leq m$ and each attribute appears at most once in an action set. That is, change suggestion on attribute A_i is non-repetitive w.r.t the actions in the action set. The intended object $o \in O^-$ after the application of an action set Γ is denoted by $o\Gamma$.

For a given $o \in O^-$ the action extraction process should result in an action set Γ which will turn $L(o)$ to the desired label if all the actions inside Γ are applied to the input data.

Cost: A cost is associated with each feature that is the cost of changing the feature value. For a given action α a cost C is associated since it suggests a feature value change. Usually, the cost of an action set Γ is calculated through the summation of the cost of each individual action i.e. $C(\Gamma) = \sum_{\alpha \in \Gamma} C(\alpha)$.

Effectiveness: to describe the ability of an action set Γ to boost the likelihood of an intended object $o \in O^-$ to be in a desirable state, we introduce this measure. It measures the gain in probability achieved by applying the action set, which is defined as follows:

$$Effectiveness(\Gamma, o) = Pr_{o\Gamma} - Pr_o \tag{1}$$

where Pr_o and $Pr_{o\Gamma}$ are the desired class label probabilities of o before and after the application of Γ, respectively. Obviously, the value of Effectiveness ranges between 0 and 1.

Profit: Profit is a business gain achieved by the application of action set which results in a desirable state. A profit value $p_l > 0$, associated with (a desired label) l, is defined by domain experts.

Net Profit: the net profit of an action set is defined as:

$$NetProfit(\Gamma, o) = \begin{cases} Effectiveness(\Gamma, o) \times p_l - C(\Gamma), \\ \qquad\qquad L(o) \neq l, L(o\Gamma) = l \\ -C(\Gamma), \quad \text{otherwise} \end{cases} \tag{2}$$

Reliability: for an action extraction method M it measures how objects are correctly labeled as the class suggested by action sets. To evaluate this, the underlying classifier could be rebuilt, incorporating the structural changes in the network suggested by an action set. It is defined as follows:

$Reliability(M)$
$=$ the fraction of objects correctly classified by action sets

Structural Feature: a structural feature is the type of feature generated by solely investigating the structure of the network data i.e. the nodes and links in the graph. As an example, given a node v in the social network modeled as graph G, its degree, ego-net degree (the number of edges connecting the neighbors of v) and ego-net out-degree (the number of edges connecting the neighbors of v to the other nodes in the graph) [2] are structural features.

IV. PROPOSED METHOD

Our method has three main elements: feature extraction, action mining core, and action evaluation process. First, the adopted input feature extraction method is described. The method we utilized is explained in more details and we reason why we use this method compared to other methods of structural feature extraction. Second, the action mining core and evaluation process are described under the topic of ''change-aware action extraction and evaluation'' where we dive into the details of extending the feature extraction method to track the changes and how the iterative evaluation process is carried out.

A. FEATURE EXTRACTION AND MODEL CONSTRUCTION

To capture the information which lies within the dependencies inside the network data, we extract structural features and later feed them to the action mining process. To extract such features we utilize ReFeX [2]. The output of this method is a set of features built based on the structure of the network. ReFeX calculates three simple measures: degree, ego-net degree and ego-net out-degree for each node. These features are called local, meaning they can be easily induced by probing only the neighborhood of each node [2]. Moreover, to generate more features that further describe the nodes of the graph, ReFeX performs an iterative process including the following steps: initially it calculates the three aforementioned measures for each node inside the network's graph. It specifically creates a feature vector of size three for each node in its first phase. Then for each node v and each measure m it defines a new measure m' which is the aggregate of m over the neighbors of v. These aggregated measure values

are then added to the corresponding feature vectors. The feature generation process continues until no new information is given by the newly added features. In other words, ReFeX keeps only the most informative features.

There are other novel methods proposed in this category such as [3], [27]–[29] which also extract structural features. These methods produce structural features by either matrix factorization or random walks but the resulting features are only informative i.e. it is unclear what exactly such features and their values represent. In other words, there is no a set of predefined features and since an action suggests a change in the value of some feature, we clearly need to extract a predefined set of structural features. In this regard, we chose ReFeX for the sake of its ability to generate a predefined set of features.

The second step is to construct a predictive model using an improved decision-tree learning algorithm as in [4] from input network data. According to the case under experimentation, the model is built using just the features provided in the input data, or by also incorporating the structural features extracted in the earlier step. In both cases, a decision tree is built to predict if an object is in the desired class or not along with the associated probability value.

B. ACTION EXTRACTION AND EVALUATION

Our proposed method aims to extract actions from network data. One of our main contributions is that we track the changes occurring to the nodes and the edges. The input of our method is a set of structural features extracted by ReFeX which we previously discussed. However, we extended ReFeX to make tracking the propagations possible because even though it extracts suitable structural features but additional information needs to be kept to make tracking possible. The details of this process are discussed in section IV-C.

We will exploit a decision tree model to extract actions as in [4] as our action mining core. Using this model is common in many action mining tasks [12], [13], [20]. So we learn a decision tree from input data incorporating the aforementioned structural features. A typical step is to traverse the decision tree for an intended object (which is currently in undesirable state) from root to leaf in a way that in each step a tree node is chosen on the way to a leaf with a higher probability of being in the desirable state. To make a node take a certain path, some of the object's feature values need to be changed, i.e. the feature should take a different value than its current value. An action suggests such a potential change in the feature value of the intended object as we introduced earlier. After the traversal of a tree path is finished, we will have an action set containing a set of changes on the values of different features of the object along the path. A collection of action sets will be produced after all of the candidate paths are taken and the action set with the highest net profit will be chosen as an optimal action set for the intended object (an object corresponds to a node in the graph). This is a typical procedure in many action mining tasks but it is noteworthy that the input features in those methods are assumed independent whereas in our work we target network datasets in which dependencies are inclusive. This may degrade the effectiveness of action since a change in the targeted object may trigger a set of changes in another object that may negatively affect the targeted object itself i.e. the side effect of change propagation.

The important characteristic of ReFeX structural features is that a node's feature is generated by investigating different areas and neighborhoods of various parts of the graph. In other words, a feature can be an aggregation of the features of other nodes or other neighborhoods inside the network.

In our case where a feature change suggested by an action results in some change in the structure of a neighborhood in the graph, any feature generated using that neighborhood will consequently change. These chain of changes may be harmless in some cases and make target object's status desirable as intended, but in some other cases, the chain of changes may negatively affect the target object meaning the suggested action may fail to make the intended object's status desirable. This is why we need to evaluate actions and extract new ones - if needed- to obtain an effective action set (one that makes a situation desirable considering the chain of changes).

A change suggested by an action for a given object may result in changes inside the input network. In this case, we may need to retrain the model because the feature values that the model was built upon could be changed. This situation is referred to as ''concept drift'' [30], even if this is the consequent of the change propagations over the network. Retraining the model will be repeated until the feature values used to build the model do not change i.e. the concept drift does not occur (or a predefined threshold for iterations is reached). Our method presents an iterative action evaluation process denoted as DAESF (Drift-aware Action Extraction and Evaluation using Structural Features) which extracts actions while including these chain effects and model reconstructions.

When an action set Γ is found, we apply it to the network. Then we check whether the feature values used to build the model are changed or not. If changed, the model will be rebuilt using new feature values i.e. respecting the change-awareness. In this case, the rebuilt model may predict $L(o\Gamma)$ as positive (desirable). If so, the algorithm halts. If $L(o\Gamma)$ is still predicted as negative (undesirable), a new action set will be extracted (that is, the previously extracted action set is not reliable as intended) using the rebuilt model and its evaluation process will be similarly resumed. In the case that the initial feature values used to build the model do not change in the first place, the action set is effective and the algorithm terminates.

The DAESF can be simply described by the following steps:

1) Given a graph G, it extracts structural features (line 2)
2) Construct a tree based model using the features
3) For a given object o, it extracts an optimal action set using the tree based model (line 5)

Algorithm 1 Pseudo code for DAESF algorithm

```
1 Procedure DAESF (G, o)
      Input  : Social Network Graph G, Intended Object o
      Output: Action Set Γ, Updated Graph G
2 F, info ← Ext-ReFeX(G)
3 repeat
4     M ← Construct-Model(G, F)
5     Γ ← ActionExtaction(F, o, M);
6     F^new_Train ← PropagateChanges(Γ, G, F, info)
7     check ← Compare(F_Train, F^new_Train)
8 until !check;
9 return Γ
```

4) In line 6 it applies the suggested changes onto the o's structural features and subsequently on the whole network
5) If the data that the model was built upon changes it then rebuilds the model due to the change propagation and the process is repeated (go to line 4). To detect these changes, we compare the feature values of training data before (F_{Train}) and after ($F^{new}{}_{Train}$) the change propagation.
6) Otherwise, return the updated graph and the resulting action set (line 9)

The time complexity of this method mostly relies on the adopted feature extraction method where the change propagation is dependent on it. Let n be the number of nodes, m be the number of edges and f be the number of features. The time complexity of the feature extraction phase is $O(f \cdot (m + nf))$ where $f \ll n$ for real-word graphs [2]. The other aspect of the complexity in our method is the tree traversal through which the optimal action sets are found. Let v be the number of nodes of the decision tree, the action extraction process takes $O(v)$.

Now to calculate the complexity of DAESF, let k be the number of iterations that our method has to take to obtain a situation where concept drift does not negatively affect the class label prediction by the newly constructed model. Hence, the time complexity of our method is $O(k \cdot (v + f \cdot (m + nf)))$. Setting the value of k is flexible and depends on the business requirement, whether the requirement is the effectiveness of actions or their reliability, i.e. actions that are truly accurate.

C. TRACKING CHANGE PROPAGATIONS

In this section, we provide a comprehensive explanation of how the structural changes suggested by an action set are propagated inside the network and how we track them.

Initially, we consider three simple features for each node as described earlier that is a result of probing the neighborhood of them. New features are generated by aggregation in an iterative fashion, then normalized and pruned in each step. Assume that the changes suggested by an action set are applied over the network. More precisely, the feature value changes suggested by an action set corresponds to addition and/or removal of some edges in certain areas of the network. Since the features are all based on the degree, the changes correspond only to the deletion and insertion of edges.

Even though an action suggests changes over an intended object o, knowing the features extracted by ReFeX and its values inside an action alone will not help to determine the number and area of changes in the network since the changes are possibly propagated over the network. Therefore, it is necessary to provide the capability for tracking such changes on the network in order to get change-aware actions.

To address this, in our method, we save some additional information in the feature extraction phase for each feature which will subsequently help to exactly determine the changes.

We save normalization factor, iteration number and feature type to achieve the requirements of our method. By having the following elements we can get the initial value and the type of each feature, its neighborhood and the number of edges to be changed:

1) *Normalization factor*: it is a number for normalizing the value of features during the feature extraction phase. There are a variety of functions to do so. Having this number and the function used for normalization enables us to obtain the original value from its normalized value.
2) *Iteration number*: by having this number we can find the neighborhood to which the changes should be applied. Given an object o, the features generated in the 1st iteration are neighbors of o, features from the 2^{nd} iteration are distance-2 neighbors of o and so on. Thus, the iteration number (k) determines the distance of neighborhood that we need to apply changes to. That is, k corresponds to the neighborhood distance of a given object for which we suggest an action.
3) *Feature type*: finally we need to determine what edges are to be either added or removed, according to the changes suggested by the action. We need to determine whether the change is related to the degree, ego-net degree or ego-net out-degree to know where exactly the edges have to change. To do this we need to keep the name of each feature generated in the first phase and forward this name to the features extraction in aggregating phase. This way, when the feature extraction process finishes, each feature will have an indicator attached to it, showing its type i.e. degree, ego-net degree or ego-net out-degree.

Example 1. This example is to better illustrate the importance of knowing ''feature type'' as defined above. Consider the situation in which an action set for o is extracted. Now it is needed to discover the effects of such action set on the graph.

Imagine the features degree, ego-net degree, and ego-net out-degree are selected for feature extraction where these features and their aggregations build up the entire feature vector for an object o. Now let us assume that after the inspection of an action α, the features of node v has to be updated to ultimately alter the state of o. Let $N_v = \{a, b, c\}$ be the set of

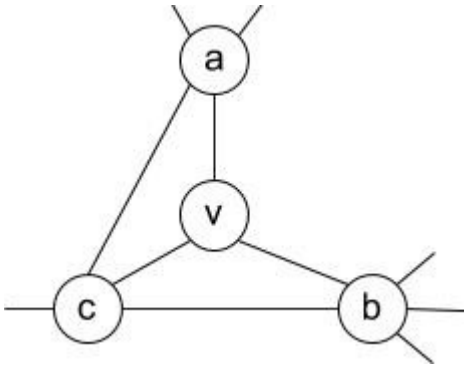


FIGURE 1. A sample graph.

TABLE 1. Possible changes on features suggested by an action for the intended object *v* with respect to feature type.

Feature Type	Change	Possible changes for sample graph in Fig. 1
Degree	Increase	Add an edge between v and another node.
	Decrease	Randomly remove one of the edges connecting to v
Ego-net degree	Increase	*Case 1*: if there are non-connected pairs of nodes in N_v, add an edge randomly. *Case 2*: otherwise if all the nodes in N_v are connected, then add a new node x, edge (v, x) and edge (x, u) where $u \in N_v$.
	Decrease	Randomly remove an edge among two nodes inside N_v.
Ego-net out-degree	Increase	Randomly add an edge (m, n) where $m \in N_v$ and $n \notin N_v$
	Decrease	Randomly remove an edge (m, n) where $m \in N_v$ and $n \notin N_v$

neighbors of v. As already described, additional information saved during feature generation helps us to determine the type of the feature and number of the edges to be changed i.e. increase/decrease its value. The following cases will occur when changing the feature value of v with respect to the ''feature type''. Table 1 illustrates possible changes when applying an action for intended object v:

This example shows the possible outcomes of applying changes when only one unit is either increased or decreased. This can easily be generalized to larger units. Consequently, these changes will affect the structure of the graph and could alter the values of the features of the other nodes.

Example 2. In this example, we will demonstrate how change propagation affects the graph structure and subsequently the structural feature values of nodes. Given the following graph (Figure. 2 (left)), in our method, a set of structural features will be extracted. In this example, we will limit the features to only three types (degree, ego-net degree, and ego-net out-degree), where no pruning and normalization is applied.

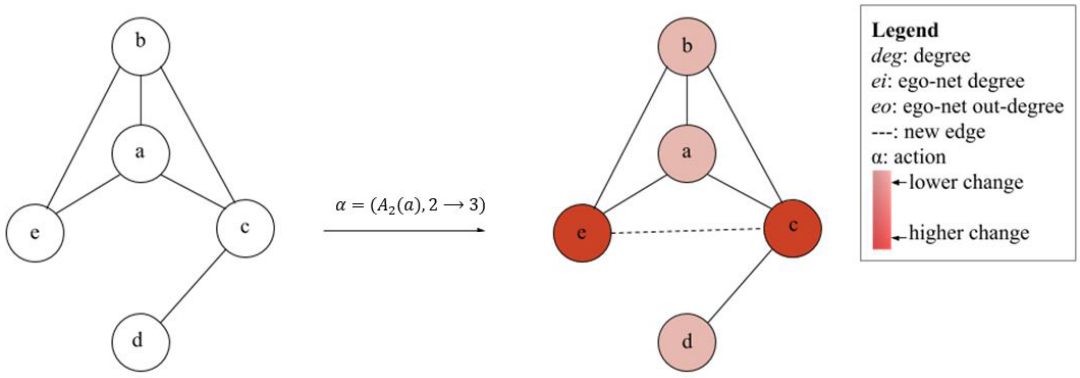
According to the graph, the features extracted are $f_a = [3, 2, 1]$, $f_b = [3, 2, 1]$, $f_c = [3, 1, 2]$, $f_d = [1, 0, 2]$, and $f_e = [2, 1, 2]$ where the 1^{st} feature corresponds to the degree, the 2^{nd} feature corresponds to the ego-net degree and 3rd one corresponds to the ego-net out-degree. Let us assume that an action $\alpha = (A_2(\text{a}),2 \rightarrow 3)$ is suggested for node a. As the action suggests, ego-net degree of node a has to be increased one unit. According to Table 1 in the previous example, we can add an edge (c, e) as shown in Fig. 2 (right). Adding this edge will affect any other feature value that has had a dependency to it. The feature vector after change propagation will become $f_a = [3, 3, 1]$, $f_b = [3, 3, 1]$, $f_c = [4, 3, 0]$, $f_d = [1, 0, 3]$ and $f_e = [3, 3, 1]$. We can see that although the action is suggested only for node a but all other node's feature values changed accordingly. Note that since the model is constructed based on such features, the propagation of changes in the network may necessitate reconstruction of the model.

V. EXPERIMENTAL RESULTS

In this section, we will compare DAESF with other state-of-the-art methods in this domain and we make comparisons against different datasets and measurements. Two methods, of Yang *et al.* [4] and Tolomei *et al.* [5], are chosen as competitors since they are both based on decision trees. The former is one of the fundamental methods in the actionable knowledge discovery domain and it is the base for many other novel methods in the field and the latter which benefits from a tree-based model (random forest), is concerned with making actionable recommendations to reach a pre-defined desirable goal. To assess the performance of the methods, we have selected the network data of friendship (Facebook and Google Plus) and co-authorship networks (DBLP). They all are publicly available and provided with edge lists which help us to create the corresponding graph and ultimately extract structural features. They are also provided with node features which is a necessity to compare competitor methods as they rely on these type of features. Facebook and Google Plus are provided with node features (shown in Table3). We use *Locale* as class feature for Facebook and *Place* as the feature for Google Plus. In case of DBLP and Hep-th, for every node u in the network the following features are generated:

- Number of papers u authored
- Number of papers u authored in goal conference
- Number of papers in which u is the first author
- The time since u authored the last paper
- The time since u last authored a paper in goal conference
- Number of time slices in which u authored a paper
- Number of Conferences/Journals in which u authored a paper
- Number of Conferences/Journals in which u was the first author
- Number of citations of u
- Number of citations of u in the goal conference
- Number of the papers cited by u
- Number of the papers in the goal conference cited by u

For DBLP and Hep-th, we use *Number of citations of u in the goal conferences* as class feature. In our experimentations,



ID	*deg*	*ei*	*eo*
a	3	2	1
b	3	2	1
c	3	1	2
d	1	0	2
e	2	1	2

ID	*deg*	*ei*	*eo*
a	3	**3**	1
b	3	**3**	1
c	**4**	**3**	**0**
d	1	0	**3**
e	**3**	**3**	**1**

FIGURE 2. Input sample network before (left), and the network after (right) change propagation.

TABLE 2. Data sets.

Dataset	# of Node Features	# of Edges	# of Nodes
Facebook	27	88,234	4,039
Google Plus Largest Node Set (1)	6	54,792	4,938
Google Plus Largest Edge Set (2)	6	1,143,015	4,926
DBLP	12	8,803	4,112
Hep-th	12	13,919	8,929

TABLE 3. Node features of Facebook and Google Plus.

Dataset	Node Features
Facebook	Education(classes, concentration, degree, school, type, with, year), work(employer, start_date, end_date, from, location, type, position, with, projects), languages, location, locale, first_name, middle_name, last_name, gender, birth_day, hometown, religon, political
Google Plus	Institute, job, last_name, Place, gender, university

we consider a specific label value as desired (positive) and others as undesired (negative).

We have used a computer of 8 GB of RAM and a Core i7 1.6 GHz processor and all the implementations are in the python programming language.

In our experiments, we compare DAESF, Yang and Tolomei in terms of several measures we described earlier. DAESF iteratively evaluates the effects of actions on the graph and rebuilds the model in the case of change propagations. This continues until the changes in training data do not affect the underlying model constructed upon or some user-defined conditions hold.

We measure the quality of actions by comparing the probability of intended object falling in a desirable state before and after the action is applied. The number showing the deficit of these two probabilities is called *effectiveness*. This number shows the improvement gained by the application of the action on the intended object. For example, an object having a 20% probability of being in a desirable state, turns into 75% after the application of action set. So the effectiveness of such an action set is 55%, meaning the object has 75% chance of falling into desirable state. This is not to be confused with the accuracy.

The performance of competing methods is compared in terms of cost, net profit, effectiveness, and total runtime. In addition to input node features of Yang and Tolomei methods, we separately fed structural features to Yang's methods (Yang-SF in Table 4) in order to compare these methods more thoroughly in the context of structural features.

In order to better compare the results, we relaxed the negative net profits to zero. This happens when the cost of an action is too high and/or the profit is very small. We have also selected the class label randomly to avoid any biased outcomes.

We sampled objects belonging to negative class randomly and extracted an action set for each sampled instance using the methods in Table 4. We then averaged the value of each measure. The average net profit in DAESF is constantly higher than Yang's and Tolomei's which shows the effectiveness of structural features. The cost is less and also the effectiveness is higher compared to the competing methods. For Yang and Tolomei only the node features are fed to them and no feature extraction was needed.

As it is shown in Table 4, the time consumed by DAESF is higher than the competing methods because we have included the time of feature extraction and feature reconstruction

TABLE 4. Experimental results.

	Methods			
DBLP	**Yang**	**Tolomei**	**Yang-SF**	**DAESF**
Avg Cost	38.81	25.6	17.5	18.6
Avg Net Profit	0.2	0.27	0.34	0.39
Avg Effectiveness	0.25	0.3	0.36	0.41
Total runtime (s)	8.7	160.59	12968.3	27895.01
Facebook				
Avg Cost	0	30.3	31.8	35.7
Avg Net Profit	0	0.09	0.3	0.38
Avg Effectiveness	0	0.12	0.33	0.42
Total runtime (s)	3.03	139	11843.41	122443.71
Google Plus (1)				
Avg Cost	210.7	95.9	58.3	58.3
Avg Net Profit	0.11	0.24	0.35	0.35
Avg Effectiveness	0.32	0.34	0.41	0.41
Total runtime (s)	7.73	491.5	15673.72	15784.1
Google Plus (2)				
Avg Cost	161.8	91	50.6	56.05
Avg Net Profit	0.1	0.07	0.28	0.31
Avg Effectiveness	0.27	0.16	0.33	0.37
Total runtime (s)	5.87	319.6	15993.12	26993.91
Hep-th				
Avg Cost	21.85	19.45	15.8	17.5
Avg Net Profit	0.12	0.19	0.24	0.31
Avg Effectiveness	0.14	0.21	0.26	0.33
Total runtime (s)	11.19	980.5	31875.5	45710.81

(change propagation) in our calculations. In other words, total runtime for Yang and Tolomei corresponds to the total time of extracting action sets for sampled objects whereas in Yang-SF it is the time of initial feature extraction in addition to action extraction, and finally in DAESF it is the time of feature extraction in addition to action extraction in each evaluation iteration.

For a more contextualized comparison, we fed structural features to the aforementioned methods namely Yang-SF. The time of the feature extraction process is also included. Yang-SF and DAESF have comparable results when no changes are propagated.

TABLE 5. Reliability of actions for competing methods.

	Methods	
Data sets	**Yang-SF**	**DAESF**
DBLP	0.86	0.89
Facebook	0.81	0.9
Google Plus (1)	0.78	0.78
Google Plus (2)	0.83	0.92
Hep-th	0.89	0.9

What we see in DBLP, Google Plus (1) and Hep-th are more complex as the DAESF needs more iterations after the changes. This shows the importance of our method as it detects the inability of an action to improve the situation of an object after the changes are propagated through the graph. Table 5 shows average reliability for Yang-SF and DAESF. We can see there are some cases for which suggested actions by Yang-SF will not result in desired label. The problem is solved by considering change propagations in the proposed method.

Even though DAESF has constantly performed better than competing methods in terms of cost, net profit and particularly effectiveness but it takes more time than the competitors. It must be noted that even though some of the competing methods' results may be comparable to ours but the actions they extract may not be as applicable as ours since they are not evaluated. Furthermore, their actions are not as reliable as ours since their application may cause concept drift. We can see the comparison results especially at Yang-SF where their improvements rely on using structural features that they do not use by default and we used such features for the sake of better and contextualized comparability. We clearly leveraged Yang and Tolomei methods by feeding structural features which increases the effectiveness of the competing methods.

VI. CONCLUSION

In this paper, we proposed a new method to better evaluate the actions in the field of actionable knowledge discovery. Our method specifically targets the network data and focuses on the dependencies in this type of data. The dependencies may result in possible unexpected changes due to the application of actions for an intended object. These changes may result in structural changes in some areas of the network other than the one which target object resides in. These changes may negatively affect the class label of the target object and degrade the effectiveness of the action. We have proposed a new method called DAESF which tracks changes in order to evaluate the actions and rebuild the model if needed. In our experiments, we show that change propagation can potentially occur when features and their values have dependencies to each other. We can also see that our method outperforms the competing methods in terms of action quality.

Another aspect of our work is the use of structural features which suits best when working with network data since the very foundation of such data is the relationships between instances. The structural features are one of the better ways to effectively encode the characteristics based on dependencies.

By inspecting the social data we can infer that constant changes to these structures are inevitable and these changes could be propagated through the network. So developing tools and methods to track the changes in real-time and rebuilding the model accordingly can be enormously beneficial.

Actionable knowledge discovery from social networks goes beyond the scope of mining social networks which commonly generates descriptive as well as predictive models / patterns while they could not directly be applied in the related domain. AKD could explicitly be emerged as a response to this need especially in the domain of social network applications. Our current work tackled the problem of change propagation in the social network in order to obtain more qualified actions, however, this can only be considered as initial steps towards extracting actionable knowledge from social networks. Structural nature of social networks has different impacts on general problem of AKD which could be separately discussed and investigated.

First, although we adopted a selected feature extraction method as well as a decision tree-based classifier from which actions are extracted for an intended object, many other feature learning as well as node classification methods in social networks could potentially be adopted regarding the type of features and how structural changes are propagated through the network. Second, since the underlying data is graph, the conducted approach could be extended for a variety of non-social complex networks including transportation, electrical transmission, wireless sensor, wired/wireless signal transmission networks as well as many others. Third, considering the problem space, in the current work we extract actions for an intended node (object) although the problem space could be very huge. The approach could be extended for more general situations where actions are extracted for a group of intended/selected objects. Fourth, we assume that the network structure is static. That is, the nodes and edges are not added/removed and the value of attributes as well as the weight of edges are not independently (and simultaneously) changed. Obviously, dynamism in the network imposes several challenging issues which requires considerable research investigations as future works. Finally, action extraction from social networks has its own limitations. For evaluating the quality of extracted actions in a real-word scenarios, the actions should be applied in the real network. However, this is only possible for the network's owner. Moreover, exploration of the space of possible changes in the network data is practically infeasible. That is, considerable research investigations are required to devise (domain-specific) heuristics to reduce the problem search space while keeping the quality of actions for real word network applications.

REFERENCES

[1] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," in *Social Network Data Analytics*. Boston, MA, USA: Springer, 2011, pp. 115–148.

[2] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos, "It's who you know: Graph mining using recursive structural features," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2011, pp. 663–671.

[3] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2016, pp. 855–864.

[4] Q. Yang, J. Yin, C. Ling, and R. Pan, "Extracting actionable knowledge from decision trees," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 1, pp. 43–56, Jan. 2007.

[5] G. Tolomei, F. Silvestri, A. Haines, and M. Lalmas, "Interpretable predictions of tree-based ensembles via actionable feature tweaking," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2017, pp. 465–474.

[6] Z. W. Ras and A. Wieczorkowska, "Action rules: How to increase profit of a company," *Proc. PKDD*, 2000, pp. 587–592.

[7] N. Kalanat, P. Shamsinejad, and M. Saraee, "Discovering cost-effective action rules," in *Proc. 4th IEEE Int. Conf. Comput. Sci. Inf. Technol. (ICCSIT)*, vol. 3, Jun. 2011, pp. 133–142.

[8] N. Kalanat, P. Shamsinejad, and M. Saraee, "Robust and cost-effective approach for discovering action rules," *Int. J. Mach. Learn. Comput.*, vol. 1, no. 4, pp. 325–331, 2011.

[9] S. Pothirattanachaikul, T. Yamamoto, S. Fujita, A. Tajima, K. Tanaka, and M. Yoshikawa, "Mining alternative actions from community Q&A corpus," *J. Inf. Process.*, vol. 26, pp. 427–438, May 2018.

[10] S. Subramani and M. O'Connor, "Extracting actionable knowledge from domestic violence discourses on social media," 2018, *arXiv:1807.02391*. [Online]. Available: http://arxiv.org/abs/1807.02391

[11] J. Ranganathan, A. S. Irudayaraj, A. Bagavathi, and A. A. Tzacheva, "Actionable pattern discovery for sentiment analysis on Twitter data in clustered environment," *J. Intell. Fuzzy Syst.*, vol. 34, no. 5, pp. 2849–2863, May 2018.

[12] Z. Cui, W. Chen, Y. He, and Y. Chen, "Optimal action extraction for random forests and boosted trees," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2015, pp. 179–188.

[13] Q. Lu, Z. Cui, Y. Chen, and X. Chen, "Extracting optimal actionable plans from additive tree models," *Frontiers Comput. Sci.*, vol. 11, no. 1, pp. 160–173, Feb. 2017.

[14] N. Kalanat, P. Shamsinejadbabaki, and M. Saraee, "A fuzzy method for discovering cost-effective actions from data," *J. Intell. Fuzzy Syst.*, vol. 28, no. 2, pp. 757–765, 2015.

[15] N. Kalanat and B. Minaei-Bidgoli, "An optimized fuzzy method for finding actions," *J. Intell. Fuzzy Syst.*, vol. 30, no. 1, pp. 257–265, Sep. 2015.

[16] J. Li, "Beyond understanding and prediction: Data mining for action," in *Proc. 26th Int. Conf. World Wide Web Companion*, Apr. 2017, p. 1361.

[17] A. A. Tzacheva, M. M. Sunny, and P. Mummoju, "MR-apriori count distribution algorithm for parallel action rules discovery," in *Proc. IEEE Int. Conf. Knowl. Eng. Appl. (ICKEA)*, Sep. 2016, pp. 127–132.

[18] J. Kuang, Z. W. Raś, and A. Daniel, "Personalized meta-action mining for NPS improvement," in *Proc. Int. Symp. Methodologies Intell. Syst.* Cham, Switzerland: Springer, Oct. 2015, pp. 79–87.

[19] M. Almardini, A. Hajja, Z. W. Raś, L. Clover, D. Olaleye, Y. Park, J. Paulson, and Y. Xiao, "Reduction of readmissions to hospitals based on actionable knowledge discovery and personalization," in *Beyond Databases, Architectures and Structures. Advanced Technologies for Data Mining and Knowledge Discovery*. Cham, Switzerland: Springer, 2015, pp. 39–55.

[20] P. Su, J. Yang, Z. Li, and Y. Liu, "Mining actionable behavioral rules based on decision tree classifier," in *Proc. 13th Int. Conf. Semantics, Knowl. Grids (SKG)*, Aug. 2017, pp. 139–143.

[21] J. Kuang and Z. W. Raś, "In search for best meta-actions to boost businesses revenue," in *Flexible Query Answering Systems*. Cham, Switzerland: Springer, 2016, pp. 431–443.

[22] H. Touati, Z. W. Raś, and J. Studnicki, "Meta-actions as a tool for action rules evaluation," in *Feature Selection for Data Pattern Recognition*. Berlin, Germany: Springer, 2015, pp. 177–197.

[23] P. Shamsinejadbabaki, M. Saraee, and H. Blockeel, "Causality-based cost-effective action mining," *Intell. Data Anal.*, vol. 17, no. 6, pp. 1075–1091, Nov. 2013.

[24] J. Pearl and T. S. Verma, "A theory of inferred causation," in *Studies in Logic and the Foundations of Mathematics*, vol. 134. Amsterdam, The Netherlands: Elsevier, 1995, pp. 789–811.

[25] N. Kalanat and E. Khanjari, "Action extraction from social networks," *J. Intell. Inf. Syst.*, vol. 54, no. 2, pp. 317–339, Apr. 2020.

[26] N. Kalanat and E. Khanjari, "Extracting actionable knowledge from social networks with node attributes," *Expert Syst. Appl.*, vol. 3, pp. 100013-1–100013-10, Sep. 2019.

[27] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*. Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[28] L. Tang and H. Liu, "Leveraging social media networks for classification," *Data Mining Knowl. Discovery*, vol. 23, no. 3, pp. 447–478, Nov. 2011.

[29] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2014, pp. 701–710.

[30] I. V. Z. E. liobait, M. Pechenizkiy, and J. Gama, "An overview of concept drift applications," in *Big Data Analysis: New Algorithms for a New Society*. Cham, Switzerland: Springer, 2016, pp. 91–114.

NASRIN KALANAT received the M.Sc. degree in computer engineering from the Isfahan University of Technology. She is currently a Researcher in data mining and machine learning with the Iran University of Science and Technology, Tehran, Iran. Her research interests include machine learning and actionable knowledge discovery from large databases and data mining.

EYNOLLAH KHANJARI received the bachelor's degree in computer science from the Sharif University of Technology, Tehran, Iran, and the Ph.D. degree in informatique from the Université de Tours, France, in 2009. He is currently an Assistant Professor with the School of Computer Engineering, Iran University of Science and Technology. He has published several research articles in the related journals and conferences. His main research fields are data analysis and extracting actionable knowledge from different data sources focusing on social networks as an emerging research domain for knowledge discovery machine learning. The current researches are collaboratively investigated in the Data Engineering Laboratory under his direction.

ALIREZA KHANSHAN received the M.Sc. degree in computer engineering from the Iran University of Science and Technology (IUST), Tehran, Iran. Since 2017, he has been a Researcher with the Data Engineering Laboratory, IUST, under the direction of Dr. Khanjari. His main research areas are in social media mining and actionable knowledge discovery mainly focused on social network data. His research interests include machine learning and big data analysis.