Tracing Relevant Twitter Accounts Active in Cyber Threat Intelligence Domain by Exploiting Content and Structure of Twitter Network

Avishek Bose[1]
Department of Computer Science
Kansas State University
Manhattan, KS,USA

Shreya Gopal Sundari[2]
Department of Electrical and Computer Engineering and Computer Science
University of New Haven
West Haven, CT, USA

Vahid Behzadan[3]
Department of Electrical and Computer Engineering and Computer Science
University of New Haven
West Haven, CT, USA

William H. Hsu[4]
Department of Computer Science
Kansas State University
Manhattan, KS, USA

***Abstract*—Due to the enormous volume of data and rate of data generation on Twitter, a challenging task is to trace user accounts to monitor these as instances of Cyber Threat Intelligence (CTI). In this paper, we propose a novel approach for cyber threat-associated user accounts tracing in the Twitter data stream based on the ranking of users according to their contextual relevance and topological information extracted from finding user communities in the Twitter network. In our approach, we use both structural information of the graph network and user accounts' tweet contents to find relevant user accounts concerning previously identified seed user accounts. Our proposed method outperforms over two relevant user recommendation methods on an annotated data of CTI related Twitter user accounts in tracing relevant user accounts as instances of cyber-threat intelligence.**

***Index Terms*—Content Aware, Community Detection, Directed Graph, Sample Network, User Account Weight**

I. Introduction

Twitter data streams as an open-source [1] for cyber threat monitoring is becoming a key point of interest in the research domain of Open Source Intelligence (OSINT). Although the feasibility and significance of Twitter as a streaming data source for tracing out Cyber Threat Intelligence (CTI) have already been established in earlier works [2], [3], the tasks of tracing user accounts as a potential information source about threats [4] and extracting pertinent information from the accounts are still less-explored research topics.

In Twitter, a user's domain-specific posts and shared contents over the social network tend to have more importance/influence to her followers if she structurally belongs to the community of relevant domain of interest such as Cyber Threat Intelligence. This assumption indicates the necessity of structural community detection prior to tracing accounts of relevant interest. On the other hand, user importance in a domain of interest can be evaluated by an automated ranking mechanism considering her relevant domain experience and activities. Therefore, many accounts belong to a community may not be domain relevant if a respective ranking measure is applied to filter user accounts. As our interest focus on some provided target topics, topic similarity calculation with user accounts' tweets is an obvious downstream process for tracing user accounts as CTI instances. From CTI perspective, it is more effective to trace and monitor groups of users rather than engaging to find a single source because this can help i) identifying emerging cyber threats, ii) measuring the credibility of the information, and iii) finding Twitter users who share the similar or paraphrased text of a cyber threat-related topic. This highly potential less explored research scope inspired us to propose an approach to extract Twitter user accounts that are involved actively in pursuing and propagating Cyber-Threat Intelligence in the Twitter network.

[abose, bhsu]@ksu.edu[1,4], $[ssund2, vbehzadan]@unh.newhaven.edu^{2,3}$

Many of the previous studies aim to trace relevant instances of CTI information using text analysis only, which is not an efficient step, as such approaches neither account for the underlying topology structure of Twitter nor are sufficient as Twitter produces a huge volume of data content with a high velocity. To address this issue, we leverage both the "followers" and "following" relations of users to formulate a homogeneous directed user graph network. After that, we detect user communities from the Twitter graph network to use the topological information of the graph for tracing CTI relevant user accounts in the form of extracting graph nodes. Additionally, one regression model employed is trained on tweets and account descriptions of user accounts that have numeric ratings (considering their pre-tag labels) to compute a weight for each user account by adding the predicted score from the regression model of each inputting text. The prediction scores for users in a community are then added to generate a overall weight for a community.

We proceed by finding similarities between the TF-IDF feature vectors of the community text comprised of all user accounts' texts and the feature vectors obtained from previously selected relevant nodes (i.e., seed nodes). We keep a record of the similarity score calculated for each community with respect to the input seed accounts and then we add each score to the corresponding community weight mentioned above to generate a sorted community ranking. To reduce computational complexity, we keep only top-*p* (where p$\in \mathbb{N}$) communities and for each of the obtained communities, we calculate the similarity between the text feature vectors of all users in the current community and the text feature vector of the seed users. Based on the aggregated value of similarity scores and respective user account weights stated above paragraph, we rank the user accounts where the resulting top-*k* (where k$\in \mathbb{N}$) user nodes are considered as traced nodes in the CTI domain. In this work, we run and evaluate our approach on a representative dataset that is sampled from an original larger dataset of user nodes in Twitter. The result demonstrates that our approach performs well in tracing user accounts who frequently tweet about CTI relevant information.

The main contributions of this paper are as follows:

- To the best of our knowledge, this is the first approach [1] for tracing relevant user accounts as instances of CTI-related information while utilizing both graph structure and user account contents.
- To generalize our approach, we apply a regression model for predicting scores of all posted tweets and the account description of each user account to rank and select user accounts according to their relevance to CTI.
- We consider central user influence to other users in a community during the process of ranking and selecting user accounts/nodes.
- We also answer two relevant questions that are helpful to understand our notion of methodological process.

II. Related Work

User account tracing comprised of several methodological steps is a hybrid task that is theoretically different from a user recommendation system. Although the purpose of this study is different from the extensively researched topic of user recommendation or, friend recommendation in the social network, the low-level working procedures of user account tracing and user account recommendation are very similar and share many common terms, backgrounds, and ideas. Considering the compatibility of our proposed work with respect to the current trend of research works in the recommendation system, we provide references to some earlier compatible works in the following paragraphs.

A recommender system can be implemented in two mutually exclusive processes using Graph structure or without using a graph structure. However, in the case of recommending users in a social network platform, the graph-based approach outperforms Non-graph-based [5] in recommending users using missing link prediction and identification. Based on the working principle of recommendation models, there are three main types of approaches for user recommendation, and these are i) Network Structure-based [6], ii) Content-based [7], ii) Hybrid [8]. These approaches encompass link prediction to efficiently find a potential future link or missing link between user nodes in social network graphs. It was evinced by several research works that neither network structure nor content-based alone can produce efficient user recommendation, on the other hand, a hybrid approach by fusing network structure and content performs better than the former ones.

Again, friend recommendation systems using Latent Dirichlet allocation (LDA) [9] also have limitations because most of the time the Twitter's short text content can only exhibit a single topic. Thus more than one topic is unlikely to obtain from such a small text. The work proposed by [10] includes structural and community information but this work lack content information to include in their analysis for generating an effective link prediction approach.

Since there is no such work as ours that is focused on tracing CTI specific user accounts in the Twitter data stream, our goal is to build a cost-effective approach by incorporating both graph-structural and content information.

III. Methodology

In this section, we describe the techniques and methods used in our proposed approach where the process steps are illustrated

[1]https://github.com/UNHSAILLab/CTI-UserNode-Identification/blob/abose17-patch-1/src_idn_updated_v10.ipynb

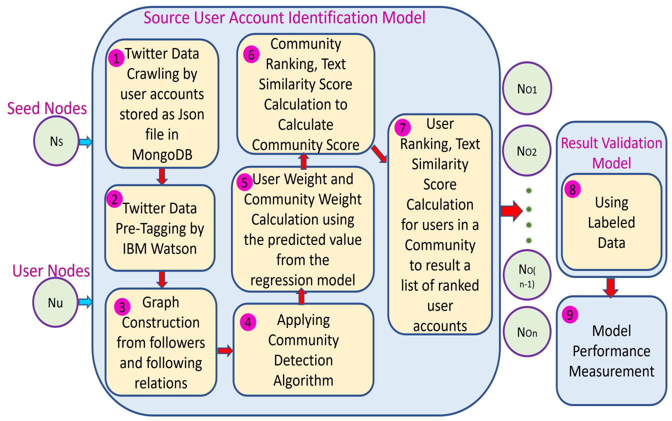
Source User Account Identification Model
Seed Nodes
Ns
User Nodes
Nu
1 Twitter Data Crawling by user accounts stored as Json file in MongoDB
2 Twitter Data Pre-Tagging by IBM Watson
3 Graph Construction from followers and following relations
4 Applying Community Detection Algorithm
5 User Weight and Community Weight Calculation using the predicted value from the regression model
6 Community Ranking, Text Similarity Score Calculation to Calculate Community Score
7 User Ranking, Text Similarity Score Calculation for users in a Community to result a list of ranked user accounts
No1
No2
No(n-1)
Non
Result Validation Model
8 Using Labeled Data
9 Model Performance Measurement

Fig. 1: Flow diagram of our proposed approach

in Figure 1. Our work is divided into two parts: (1) User account tracing process, and (2) the Result Validation process. This approach is formally depicted in algorithm 1 where 'U_acc' and 'U_accs' represent a single user account and multiple user accounts respectively.

A. Graph Construction

As we want to construct a directed user graph, therefore for outgoing edges in "follower" relation, we employ the current Twitter user account as a source node (n_{s_i}) (where i$\in \mathbb{N}$), and all the Twitter user accounts in the "follower" list of the current account as destination nodes (n_{d_j}) (where j$\in$ $\mathbb{N}$). Similarly, for incoming edges in the "following" relation, we consider all the user accounts in the "following" list of the current Twitter account as source nodes (n_{s_i}) and the current Twitter account as a destination node (n_{d_j}). We then use NetworkX to build a directed graph $\overrightarrow{G}$. This step is shown in sub-block 3 of the process flow diagram Figure 1.

B. Text Pre-processing

By nature, Twitter texts are often not grammatically correct, and the words in a tweet can be misspelled or abbreviated because of Twitter's promptness and short text length structure. To extract usable information from the text, we modularize the steps of the text pre-processing task in the following points i) contraction mapping: a contraction dictionary used for generating expanded words, ii) misspelled word correction: using a tool named SymSpell [11], and iii) token removal: remove punctuation, non-alphanumeric tokens, stopwords, and token length shorter than one, etc. For each Twitter account in the dataset, we pre-process all the most recent 50 tweets, the account's description, and a combined text produced by concatenating the recent 50 tweets. We then apply TF-IDF to generate feature vectors for the textual content of each account.

C. Community Detection

After constructing the directed homogeneous user graph $\overrightarrow{G}$, we apply Leiden [12] community detection algorithm to find communities $\cup_{i=1}^{i=\mathbb{N}} C_i = \mathbf{N}C$ in the user graph where $\mathbf{N}$ is the number of communities found by the algorithms. This detection algorithm extracts communities by optimizing modularity that compares the relative density of edges inside the community to edges outside the community. The underlying principle of the Leiden algorithm [12] follows the basic principle of well known Louvain [13] algorithm that generates communities that are connected, converge to a locally optimally assigned partition. However, the Leiden algorithm runs way faster than the Louvain algorithm.

Figure 2 illustrates an abstract structure of a community detected by the community detection algorithm where the three bubble notes and the table of node contents are displayed to simply represent how the proposed approach works on a community. The Look-Up table in the figure keeps the required information needed for each node in a community for future analysis and computation. Figure 1 process flow diagram has depicted this step in the sub-block 4.

D. Community Weight and User Weight Calculation

In the process of tracing CTI relevant user accounts U_r as outputs, we calculate the weight of each community after calculating the weights of their corresponding user accounts. We have grouped two different text entities i) all the fifty tweets and ii) the description of each Twitter account to store in a new dataset except the tweets and descriptions from the sample dataset user account set (discussed in the following Section of 4.2 and 4.3). Each text from the text dataset is analyzed and pre-tagged by the IBM Watson text categorization tool shown in sub-block 2 of the process flow diagram Figure 1. We applied an efficient semi-automated strategy to rate all the tweets and the descriptions from the text dataset such as assigning a numeric score to a text depending on the pre-tag category generated by IBM Watson NLU. The pre-tag category score for the dataset is specified distinctively by two cyber-security expert raters and clearly explained in Section 4.1. To show the performance of our proposed approach we only use the sample dataset so that we can evaluate the resulting outputs.

In order to generalize the process, we employ a ridge regression model being trained on the newly created dataset of tweets and descriptions of accounts with their corresponding scores discussed above paragraph. We fed all tweets and the description of each sample dataset user account to the regression model to predict each text's CTI relevance score that can be used to calculate weight for each user account of the sample dataset. To obtain a user account's weight value, the score for each tweet is added to the score of the account's description of a user account. The notion of this generalization by employing a regression model is crucial because we want the proposed approach to working on even different datasets or in different domains that may not be specific to CTI.

For a community C_i detected, we now have each user weight W_{n_i}, and we combine all user weights $\sum_{i=1}^{i=\mathbf{M}} W_{n_i} = W_{C_i}$ (**M** indicates number of users where $\mathbf{M} \in \mathbb{N}$) together in a community to calculate corresponding community weight W_{C_i}. It is very likely that bigger communities will obtain higher scores than the smaller communities because of having a large number of users, however, a smaller community can also have vital importance in terms of their users' shared content and users' expertise. So, to normalize the community weight, we divide the community weight value by the number of user accounts in the communities $(W_{C_i}/\sum_{i=1}^{i=\mathbf{M}} n_i)$. At this point of our analysis, we select only top-p (where p$\in \mathbb{N}$) filtered communities $\bigcup_{i=1}^{|F|} C_i^F$ (where $F \subset \mathbb{N}$) for the next step of the process to minimize the computational overhead. This process step is shown in sub-block 5 of Figure 1 and the detail of user weight calculation by predicting values from the regression model is shown in figure3.

E. Text Similarity Calculation and Community Ranking

After calculating all the community weights $\bigcup_{i=1}^{i=\mathbf{N}} W_{C_i}$ using predicted values from the regression model mentioned in the above subsection, we formulate text vector of each community

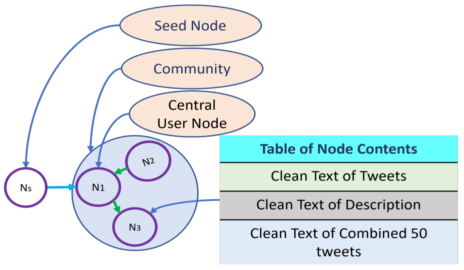

Fig. 2: High level structure of a user community in our analysis

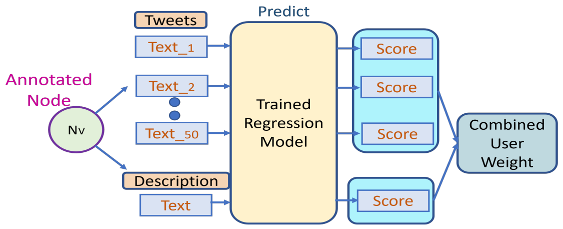

Fig. 3: User weight calculation using the predicted values of account's tweets and description

$\overrightarrow{T}_{C_i}$ using TF-IDF vectorization. However, before doing so, we combined all the fifty tweets of each user account in a community to make a single long text T_{n_i} for each community. We also add **Q** input seed nodes' $\mathbf{Q}S_i$ (where i$\in \mathbb{N}$) texts in the text vectorization process. Then we calculate cosine similarity *community_cosim_score* = *cosim*$(\overrightarrow{T}_{C_i}, \overrightarrow{T}_{S_i})$ between text vector of each seed node T_{S_i} and combined text vector for each community T_{C_i}. At this point, we have top-p communities that are already ordered based on their corresponding weights. Now, for each seed input S_i, we add each community weight W_{C_i} to the corresponding score of cosine similarity *cosim*$(\overrightarrow{T}_{S_i}, \overrightarrow{T}_{C_i})$ between the community combined text and seed node's text vectors. Therefore, the community score V_{C_i} can be presented by following equation. This process is presented in sub-block 6 of the process flow diagram Figure 1.

$V_{C_i} = W_{C_i}$ + *cosim*$(\overrightarrow{T}_{S_i}, \overrightarrow{T}_{C_i})$ **(1)**

F. User Node Scoring and Ranking

After community ranking (*rank*$(\bigcup_{i=1}^{i=\mathbb{N}} C_i)$), we start taking each user n_i from the top p-communities C_i^F consecutively based on the community ranking as we want to minimize computational complexity by reducing the number of communities **N**C. Then we extract the combined text of the top fifty tweets T_{n_i} for each user n_i and TF-IDF vectorize each user's texts in a community with the seed nodes' tweet texts $T_{\bigcup_{i=1}^{i=\mathbb{N}} S_i}$ as described earlier. After that, we calculate cosine similarity and add the similarity score *user_cosim_score* = *cosim*$(\overrightarrow{T}_{n_i}, \overrightarrow{T}_{S_i})$ with the user weight that we obtained from the process mentioned in subsection III D. At this point, as we have already got user accounts with their scores, we mention the user account as a resultant output U_r. We rank the user accounts based on the added score V_{U_r} of user weight W_{n_i} and similarity score *cosim*$(\overrightarrow{T}_{n_i}, \overrightarrow{T}_{S_i})$ and rank them by only taking into account the filtered communities $|F|C$ and for all the seed nodes $\forall S$. We mentioned earlier that the central user in a community might influence other members in the community. Therefore, we added a partial weight to other user nodes of the corresponding community by applying the following formula.

U_r - *max*(U_r) = U_r - *maxmal*(U_r) + *maxmal*(U_r)*β **(2)**

[β is the influence factor of the central user account in a community and *maxmal* is the user account with maximum value]

After that, we re-rank the user nodes list again based on their updated combined scores to produce the resulting user node output list in descending order of priority. This step can be seen in sub-block 7 of Figure 1.

G. Model Output Validation

Social network data annotation is an expensive task that has inspired us to develop our approach in a semi-supervised manner. Instead of running our implementation on the original 50K large dataset, we extracted a representative sample dataset from the original datasets. Then, we run and validate our result on the sample dataset. To serve this purpose, we annotate the sample dataset and propose a novel validation process. User nodes in the sample dataset are labeled as "relevant" and "not relevant" based on their relevance to CTI. We calculate the **Precision** value by finding the ratio between the number of resulting "relevant" user accounts and the total number of obtained user accounts. **Recall** value is calculated by finding the ratio between obtained "relevant" users count and the total number of "relevant" users in the sampled annotated dataset. We calculate **F1-**β where user can give input the value of β. We feed the seed accounts as a set of seed nodes to the model, and the resulting nodes we get from the model have been fed again to the model to find cyber threat-related nodes. This process iterates for a certain user-defined number of epochs, and after completing the process, we get the actual result to validate using the aforementioned performance metrics. We implement different validation cases according to different percentages of the number of resulting user nodes such that a certain percentage of resulting ranked users nodes is selected to feed into the model again for a certain number of epochs. For a particular set of seed nodes, for all different percentages of node selection, this process is run for the proposed method and all other compared methods. As we execute the method with different sets of seed nodes, we finally average all the calculated performance metrics values of the resulting nodes that are obtained from each set of given seed nodes. The final step of the proposed model includes **Precision**, **Recall**, and **F1-**β values for all the different selection percentages with respect to resulting user nodes. This process is exhibited in sub-block 9 of the process flow diagram Figure 1.

IV. Data Collection and Preparation

As we want to obtain the instances of CTI in the form of tracing Twitter user accounts, we crawled data through Twitter user accounts. For this purpose, we used a Twitter API called Tweepy [14] to start scrapping specific user details (account descriptions, the contents of their last 50 tweets, and lists of the "followers" and the "followings" U_accs) from an account of our colleague. Our colleague is a passionate cyber threat enthusiast and an active Twitter user who is following expert cyber threat professionals and also being followed by other cyber threat enthusiasts. We collected tweet objects up to three levels of the follower and following list from our colleague's account and this step made us able to collect 50K Twitter user accounts. For each user account, we collected their follower and following lists, account description, and recent fifty tweets. Then we create a new entity namely "AllText" by concatenating all the recent fifty tweets of each user along with the corresponding account's description. This step is illustrated in sub-block 1 of Figure 1.

A. Text Rating

The text dataset derived by combining tweets and the description (except sample dataset users) is rated to train the regression

Algorithm 1: Tracing User Accounts

```
input : seedNodeList
output: User NodeIDs
Edge generation $e(n_{s_i}, n_{d_j})$
Directed graph $\overrightarrow{G}$ construction
Community Detection
for each community $C_i$ generated in $\cup_{i=1}^{i=\mathbb{N}} C_i$ do
    for each userNode $n_i$ in Community $C_i$ do
        $\{n_i : W_{n_i}\} \leftarrow$ regression(U_acc $n_i$)
    end
    $\{n_i : W_{C_i}\} \leftarrow W_{C_i}$ (where $W_{C_i} = \sum W_{n_i} / \sum n_{c_i}$)
end
$\bigcup_{i=1}^{|F|} C_i^F$ := Filter_community(top-$p$)
$\overrightarrow{T}_{S_i}$ := Text_Vector(seedNodes' Texts $T_{S_i}$)
$\overrightarrow{T}_{C_i^F}$, := Text_Vector(filteredCommunity.Texts $T_{C_i^F}$)
$\bigcup_{i=1}^{i=|F|} \{C_i^F : V_{C_i^F}\}$ := $W_{C_i^F}$ + cosim($\overrightarrow{T}_{S_i}$, $\overrightarrow{T}_{C_i^F}$)
for each community in ranked($\bigcup_{i=1}^{|F|} C_i^F$) do
    $\overrightarrow{T}_{n_i}^F$ := Text_Vector($T_{n_i}^F$)
    $\bigcup_{r=1}^{r=|u_r^C|} \{u_r : V_{u_r}\}$ := $W_{n_i}$ + cosim($\overrightarrow{T}_{S_i}$, $\overrightarrow{T}_{n_i^F}$)
end
comMap := rank($\bigcup_{i=1}^{i=|F|} \{C_i^F : V_{C_i^F}\}$)
usrMap := rank($\bigcup_{r=1}^{r=|u_r^C|} \{u_r : V_{u_r}\}$)
traced.U_accs := score_distribution(comMap, usrMap)
if traced.U_accs.scores <= Threashold K then
    validate_result(traced.U_accs $u_r$)
else
    skip U_acc $u_r$
end
```

model. We used the scikit-learn ridge regression library where we assign alpha=1.0 and random_state=241 with default settings for the rest of the parameters. Raters assign a score to each text of the derived text dataset based on the text's relevance to any of the categories such as "CTI", "Technology" and "Computer". To facilitate and speeding up the rating process, raters took help from one of the text analytic features of IBM's Watson Natural Language Understanding (NLU) service [15] called 'Categories' to tag each text by category. This feature of Watson NLU returns a five-level taxonomy of each text and amongst them, we consider using three categories "antivirus and malware", "Technology and Computing", and "computer science". These three categories "antivirus and malware", "Technology and Computing", and "computer science" are mapped to the rater's defined three categories "CTI", "Technology", and "Computer" respectively. The categories assigned were limited up to the top three as of the highest score order given by the Watson NLU. Raters consider the category of "antivirus and malware" has precedence over the category of "Technology and Computing" and "Technology and Computing" has precedence over "computer science" for a particular text. If Raters agree with the Watson tagging of a text, they put 1 for "CTI", 0.5 for "Technology", and 0.1 for "Computer". If raters do not agree a tweet category falls in any of the raters' defined three categories, they put 0 as a score for that tweet. We tag tweets and the description of 50K user accounts (except sample dataset users) that are 1802852 tweets and 16596 account descriptions. To generalize our proposed approach, only the text rating process should be modified by assigning a different set of scores to different texts. The scores of text and their mapping with corresponding tagged categories from IBM Watson will be determined by the domain experts.

B. Extracting Sample Network

To validate the results produced by the model, we extracted a sample dataset from the 50K labeled Twitter user accounts and created a sample network from the sample dataset. We tried to retain the properties of the original network for creating this sample dataset where the root of the sample network remained the same. Starting from the root node, we randomly gathered 75 user accounts from the "Followers" and "Following" lists each (total of 150 user accounts). We observed that there may be a chance of duplication of IDs because of the network structure. So before moving on to the next step, duplicate user IDs are removed from the collection but not from the network which gave us 148 unique user IDs at this level. In the next stage of creating a sample network, we collected three user accounts from the "followers" and the "following" lists of previously obtained 148 user accounts that resulted in 789 unique user accounts. Finally, the sample network has 938 unique user accounts.

C. Annotation of Sample Network

The Twitter accounts of the sample network are manually annotated by two human-powered cyber threat experts. The accounts are classified into either **"relevant"** or **"irrelevant"** with respect to CTI. Twitter accounts are considered relevant to CTI if the description of the account is related to cyber threats and at least two of the collected tweets are about cyber threat incidents. In case, if there no description is present in the account details, but the account has three tweets related to CTI, then the account is labeled as **"relevant"**. Keywords such as "vulnerabilities", "zero-day", "malware attacks", "Phishing", "APT groups", "cyber espionage", etc are good indicators to label text category. The rest of the data of the sample network are labeled as **"irrelevant"**. Taking into account all these specifications to annotate the sampled dataset, we got 199 CTI **"relevant"** Twitter accounts out of 938 accounts in the sample network. After that, the text processing steps described previously are applied to the text data before running the validation process on the sample network. The sample network extraction and annotation steps are shown in sub-block 8 of Figure 1 where it is used to validate the proposed method's output. It is worth mentioning that the Inter Annotator Agreement (IAA) of the annotated sample dataset by the two annotators is 0.9617 using the Cohen Kappa method.

V. Experimental Setup

In this study, we applied the Leiden community detection algorithm for the 50k tagged dataset, but we evaluate our result on the sample annotated dataset. The standard Python library of this algorithm does not provide any hyperparameter tuning option, and this works as a black-box method. We evaluate the performance of our results against two relevant user recommendation methods [16], and we apply the Leiden algorithm on the 938 annotated user nodes where the Leiden algorithm finds 31 communities out of 938 user nodes. We experimented with three different sets of seed nodes where each set includes three "relevant" user nodes that are randomly picked from the annotated sample dataset. Each time, we consider selecting a percentage (from 10% to 100%) of the resulting nodes that are outputted using the given seed nodes. The selected nodes are then fed into all the methods (proposed, and two comparing methods) for 5 epochs. So, for each method, for one set of seed nodes, we get 10 different values of a performance metric according to different percentage selection (from 10% to 100%) of resulting nodes. Finally, we average each performance metrics output of all the seed node sets for each method. We used F1-β performance metric where we keep the value of β is 0.5, and the influence factor is 20%. We retrieve two maximally similar communities from the similarity matrix for each seed user node fed for tracing user nodes. Later on, we select the top 20 users (k=20) from the filtered communities where the communities are ranked based on the added score of a community weight with its corresponding similarity score. Similarly, user nodes in a community are also ranked based on the similarity scores added with their corresponding user weights.

VI. Result Evaluation and Discussion

To the best of our knowledge, this is the first work to trace CTI related user accounts in the Twitter data stream considering both structure and contents in the network. As our approach can be considered a specific case of user recommendation, we adopt two relevant user recommendation methods to compare the performance of our approach against them.

Friend of Friend (FoF) is a well known user recommendation method [16] in the social network domain that we first consider to compare its performance against our proposed approach. This approach results in a list of recommended users where a user connected to another user is connected to the target user.

Content-plus-link method (CplusL) [16] considers incorporating content matching process with social link information that is obtained from the underlying structure of a social network. This method emphasizes exposing a network path to a weak tie or an unknown user, the recipient user as a result would be able to accept the recommendation.

From the leftmost column of Table 1, we can see that our proposed method outperforms the two compared approaches in terms of precision for all the selection threshold percentages. However, the recall value is quite low in all the cases of selection percentage against both compared approaches. The reason behind this paradigm is that our method results in a fewer number of user nodes whereas the two approaches result in a large number of user nodes. As the number of resulting traced user nodes of our proposed method is not analogous to the compared approaches, the selection percentage cannot make here any difference. On the other hand, we value precision more than recall because the purpose of our work is to find highly CTI relevant user nodes rather than getting more relevant nodes. This intuition has inspired us to use $F_1 - \beta$ score where the $beta$ is 0.5 also mentioned above. Therefore, we can also see a noticeable performance improvement with respect to the $F_1 - \beta$ score. We have also calculated the Pearson Correlation Coefficient (R-value) between the predicted values of tweets and descriptions of sample user network from the regression model and the generated ratings from the semi-automated process. We conduct this analysis to evaluate how effective is our regression model for predicting a text score regarding CTI expects where texts' scores are added to be used to calculate corresponding user weights and we found the R-value is 0.7167. In this analysis, we observe that user accounts in the same community tend to share similar and relevant contents which helps us develop intuition to find specific information by identifying underlying communities in a social network.

Question 1: Why the predictive analysis performed on sample annotated user accounts is necessary?

TABLE I: Performance comparisons with different methods on the sampled annotated dataset

Selection Threshold Percentage	Proposed			FoF			CplusL		
	Prec	Rec	F$_1\beta$	Prec	Rec	F$_1\beta$	Prec	Rec	F$_1\beta$
10%	0.55	0.08	0.26	0.27	0.68	0.32	0.29	0.33	0.30
20%	0.58	0.11	0.32	0.27	0.70	0.32	0.29	0.34	0.29
30%	0.60	0.12	0.33	0.27	0.71	0.32	0.28	0.35	0.29
40%	0.56	0.11	0.31	0.27	0.73	0.31	0.28	0.37	0.29
50%	0.61	0.12	0.34	0.26	0.74	0.31	0.27	0.38	0.28
60%	0.64	0.12	0.35	0.26	0.74	0.31	0.28	0.42	0.30
70%	0.65	0.13	0.36	0.26	0.74	0.31	0.28	0.46	0.30
80%	0.66	0.13	0.37	0.26	0.74	0.31	0.28	0.49	0.31
90%	0.66	0.13	0.37	0.26	0.75	0.30	0.28	0.53	0.30
100%	0.67	0.13	0.37	0.26	0.75	0.30	0.27	0.57	0.30

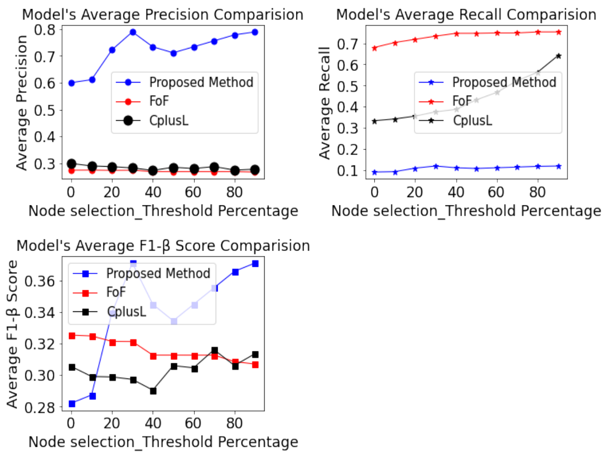

Fig. 4: Precision, Recall, and F$_1\beta$ comparison of our approach against two relevant user recommendation approaches

Answer: We mentioned in Subsection named "Community Weight and User Weight Calculation" that the proposed approach computes the users' weights using predicted scores from the regression model. The predictive analysis works for the sample dataset users whose contained texts are unknown to the regression model where the model is trained on the original 50K user datasets. We only annotate the sample user network to evaluate the performance of our approach because user account annotation is an expensive task. To practically present our proposed approach and to show its performance, we utilized the new validation technique after executing the implementation on the sampled dataset.

Question 2: Why do we develop the validation process in an iterative manner?

Answer: One of the reasons for iteratively conducting the validation process is that we intend to generate an evolving list of traced CTI user accounts. Feeding a fixed seed accounts can cause a biasness towards user preference and that should not be encouraged. Contrary to this, selecting a new set of seed accounts each time from the result can certainly address the biasness issue of seed node list selection. Another reason is that iterative filtering of user accounts also increases the probability of tracing more relevant CTI user accounts.

VII. Conclusion and Future Work

In this semi-supervised approach, we have tried to trace and recommend Twitter user accounts that can serve as instances of CTI related information according to a set of given seed nodes. We have introduced a couple of new methodological processes to accomplishing the task such as predicting a text score by the regression model, community formation, user weight calculation, and central user influence measurement. Moreover, our proposed approach can be easily adopted in a different domain of interest by training the regression model on a rated dataset respective to that domain using the semi-automated rating process. We found that applying different regression algorithms or distinct regression models for tweets and descriptions individually do not make any noticeable difference in the performance. We have also provided a method to validate and evaluate the generated results of our proposed approach where the method feeds resulting user nodes as seed nodes iteratively into the process with different selection percentages. However, our implementation should be validated with a larger representative sample annotated user dataset. An interesting future direction would be to apply a newly introduced tensor-based Graph Convolutional Network on a sufficiently labeled Twitter user dataset.

References

[1] A. Bose, V. Behzadan, C. Aguirre, and W. H. Hsu, "A novel approach for detection and ranking of trendy and emerging cyber threat events in twitter streams," in *2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2019, pp. 871–878.

[2] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, and R. C. Miller, "Twitinfo: Aggregating and visualizing microblogs for event exploration," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 227–236. [Online]. Available: https://doi.org/10.1145/1978942.1978975

[3] S. Mittal, P. K. Das, V. Mulwad, A. Joshi, and T. Finin, "Cybertwitter: Using twitter to generate alerts for cybersecurity threats and vulnerabilities," in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2016, pp. 860–867.

[4] V. Behzadan, C. Aguirre, A. Bose, and W. Hsu, "Corpus and deep learning classifier for collection of cyber threat indicators in twitter stream," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 5002–5007.

[5] L. Pan, T. Zhou, L. Lü, and C.-K. Hu, "Predicting missing links and identifying spurious links via likelihood analysis," *Scientific reports*, vol. 6, no. 1, pp. 1–10, 2016.

[6] I. Ahmad, M. U. Akhtar, S. Noor, and A. Shahnaz, "Missing link prediction using common neighbor and centrality based parameterized algorithm," *Scientific reports*, vol. 10, no. 1, pp. 1–9, 2020.

[7] R. Nidhi and B. Annappa, "Twitter-user recommender system using tweets: A content-based approach," in *2017 International Conference on Computational Intelligence in Data Science (ICCIDS)*. IEEE, 2017, pp. 1–6.

[8] S. Cheng, B. Zhang, G. Zou, M. Huang, and Z. Zhang, "Friend recommendation in social networks based on multi-source information fusion," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 5, pp. 1003–1024, 2019.

[9] G. Zhao, M. L. Lee, W. Hsu, W. Chen, and H. Hu, "Community-based user recommendation in uni-directional social networks," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013, pp. 189–198.

[10] J. Valverde-Rebaza and A. de Andrade Lopes, "Exploiting behaviors of communities of twitter users for link prediction," *Social Network Analysis and Mining*, vol. 3, no. 4, pp. 1063–1074, 2013.

[11] W. Garbe, "Symspell 6.4," 2020. [Online]. Available: https://github.com/wolfgarbe/symspell

[12] V. A. Traag, L. Waltman, and N. J. Van Eck, "From louvain to leiden: guaranteeing well-connected communities," *Scientific reports*, vol. 9, no. 1, pp. 1–12, 2019.

[13] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[14] J. Roesslein, "Tweepy documentation," 2020. [Online]. Available: https://docs.tweepy.org/en/v3.10.0/

[15] IBM, "Ibm watson natural language understanding documentation," 2021. [Online]. Available: https://cloud.ibm.com/apidocs/natural-language-understanding?code=python

[16] J. Chen, W. Geyer, C. Dugan, M. Muller, and I. Guy, "Make new friends, but keep the old: Recommending people on social networking sites," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 201–210. [Online]. Available: https://doi.org/10.1145/1518701.1518735