

TRUST ASSESSMENT IN ONLINE SOCIAL NETWORKS

Shubhangi Mahule¹, Yetaboina Pavani², Tulugu Karthik³, Sumayya Fathima⁴,
Potharaju Rajeev⁵

¹Associate. Professor, CSE Dept, ACE Engineering College, Hyderabad, India.

^{2,3,4,5}Student, CSE Dept, ACE Engineering College, Hyderabad, India.

ABSTRACT

Assessing trust in online social networks (OSNs) is critical for many applications such as online marketing and network security. It is a challenging problem, however, due to the difficulties of handling complex social network topologies and conducting accurate assessment in these topologies. To address these challenges, we model trust by proposing the three-valued subjective logic (3VSL) model. 3VSL properly models the uncertainties that exist in trust, thus is able to compute trust in arbitrary graphs. We theoretically prove the capability of 3VSL based on the Dirichlet-Categorical (DC) distribution and its correctness in arbitrary OSN topologies. Based on the 3VSL model, we further design the Assess Trust (AT) algorithm to accurately compute the trust between any two users connected in an OSN. We validate 3VSL against two real-world OSN datasets: Advogato and Pretty Good Privacy (PGP). Experimental results indicate that 3VSL can accurately model the trust between any pair of indirectly connected users in the Advogato and PGP.

1. INTRODUCTION

Online social networks (OSNs) are among the most frequently visited places on the Internet. OSNs help people not only to strengthen their social connections with known friends but also to expand their social circles to friends of friends who they may not know previously. Trust is the enabling factor behind user interactions in OSNs and is crucial to almost all OSN applications. For example, in recommendation and crowd sourcing systems, trust helps to identify trustworthy opinions and/or users. In online marketing applications, trust is used to identify trustworthy sellers. In a proactive friendship construction system, trust enables the discovery of potential friendships. In wireless network domain, trust can help a cellular device to discover trustworthy peers to relay its data. In security domain, trust is considered an important metric to detect malicious users or websites. Given the abovementioned applications, one confounding issue is to what degree a user can trust another user in an OSN. This paper concerns the fundamental issue of trust assessment in OSNs: given an OSN, how to model and compute trust among users?

Trust is traditionally considered as reputation or the probability of a user being benign. In online marketing, users rate each other based on their interactions, so the trust of a user can be derived from aggregated ratings. In the network security domain, however, the trust of a given user is defined as the probability that this user will behave normally in the future. Based on results from previous studies, we define trust as the probability that a trustee will behave expected, from the perspective of a trustor. Here, both trustor and trustee are regular users in an OSN where the trustor is interested in knowing how trustworthy the trustee is. This general definition of trust makes it applicable for a wide range of applications. We also assume that trust in OSNs is determined by objective evidence, i.e., cognition based trust, is not considered in this paper.

2. OBJECTIVES

The objective of the project is to address the fundamental challenge of trust assessment in Online Social Networks (OSNs). Given that OSNs are integral to strengthening social connections and expanding social circles, trust emerges as a critical factor influencing user interactions within these networks. Trust is essential for the successful operation of various OSN applications, such as recommendation systems, crowd sourcing, online marketing, proactive friendship construction, wireless networks, and security. The primary focus of the project is to develop a robust model and computational framework for trust among users in OSNs. The goal is to determine the degree to which a user can trust another user within the OSN environment. The project acknowledges that trust in this context is traditionally perceived as reputation or the probability of a user being benign. In online marketing, trust is often derived from aggregated ratings provided by users based on their interactions. User Interaction Analysis: Analyze user interactions within OSNs to identify patterns and behaviors that contribute to the establishment or erosion of trust. Incorporate insights from user interactions to enhance the accuracy of trust assessments. Integration of Reputation and Ratings: Acknowledge that trust in OSNs is often associated with reputation and user-provided ratings.

Integrate reputation mechanisms and aggregated user ratings into the computational framework to enhance the trust assessment process. Scalability and Adaptability: Ensure that the model and framework can scale to accommodate the

dynamic nature of OSNs and adapt to evolving user behaviors and network structures. Validation and Evaluation: Conduct thorough validation and evaluation processes to assess the effectiveness and accuracy of the developed model and computational framework.

3. METHODOLOGY

We model a social network as a directed graph $G = (V, E)$ where a vertex $u \in V$ represents a user, and an edge $e(u, v) \in E$ denotes a trust relation from u to v . The weight of $e(u, v)$ denotes how much u trusts v , which is commonly referred to as direct trust. A trustor may leverage the recommendations from other users to derive a trustee's trust, which is called indirect trust. We are interested in computing the indirect trust between two users who have not established a direct trust previously. To solve this problem, we first need to design a trust model that works with both direct and indirect trust. Based on the assumption that trust is determined by objective evidence, designing a trust model can be stated as follows.

Given the interactions between a trustor and a trustee, how to model the trust of the trustee, from the trustor's perspective? The second problem is to compute/infer indirect trust between users in an OSN. Solving this problem means the trust between two users, without previous interactions, can be computed. Because the indirect trust inference is available, a trustor can conduct a trust assessment of a trustee in an OSN. As such, the second problem is formulated as follows.

Given a social network $G = (V, E)$, $\forall u$ and v , s.t. $e(u, v) \notin E$ and \exists at least one path from u to v , how does one compute u 's trust in v , i.e., how should u trust a stranger v ?

4. LITERATURE SURVEY

Three-Valued Subjective Logic Model:

Liu et al. (2019) proposed a novel approach to trust assessment in online social networks (OSNs) using a three-valued subjective logic model. This model extends traditional binary logic to accommodate uncertainty and ambiguity inherent in trust assessments. By incorporating three truth values (true, false, and unknown), the model captures the subjective nature of trust evaluations, allowing for more nuanced and context-aware trust assessments.

Privacy and Trust in People's Opinions:

Basu et al. (2014) examined the intersection of privacy and trust within online environments, particularly focusing on the impact of privacy concerns on individuals' willingness to express opinions and engage in online interactions. The study highlighted the importance of privacy-preserving mechanisms in fostering trust and promoting participation in online communities.

Reputation Systems:

Resnick et al. (2000) introduced reputation systems as a mechanism for evaluating the trustworthiness of users in online communities. These systems aggregate feedback and interactions from users to generate reputation scores, which serve as proxies for trustworthiness.

5. PROPOSED SYSTEM

The proposed system introduces the AssessTrust (AT) algorithm, based on the 3VSL model, to address limitations in existing trust models. AT decomposes the network between trustor and trustee into a parsing tree, facilitating the computation of indirect trust. By leveraging trust operations such as discounting and combining, AT offers precise trust assessments, particularly in scenarios where uncertainty in trust is appropriately addressed.

6. HARDWARE AND SOFTWARE REQUIREMENTS

6.1 HARDWARE REQUIREMENTS:

- System: Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz
- Hard Disk : 1 TB.
- Input Devices : Keyboard, Mouse
- Ram- 4 GB.

6.2 SOFTWARE REQUIREMENTS:

- Operating system : Windows XP/7/10.
- Coding Language : Java
- Tool : Netbeans
- Database : Mysql

7. PACKAGES USED

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like K Java, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed dynamically on end user devices. MIDP contains the following packages, the first three of which are core CLDC packages, plus three MIDP-specific packages.

- * java.lang
- * java.io
- * java.util
- * javax.microedition.io
- * javax.microedition.lcdui
- * javax.microedition.midlet
- * javax.microedition.rms

8. TECHNOLOGY DESCRIPTION

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple ▪ Architecture neutral ▪ Object oriented
- Portable ▪ Distributed ▪ High performance ▪ Interpreted
- Multithreaded ▪ Robust ▪ Dynamic ▪ Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed.

9. SOURCE CODE

```
import os
import MySQLdb
from flask import Flask, session, url_for, redirect, render_template, request, abort, flash
from database import db_connect
from database import
db_connect,vit_act,view_vit,user_reg,user_loginact,user_viewimages,vit_info,get_recomendations,getdisease_recomendations
from database import db_connect
from werkzeug.utils import secure_filename
app = Flask(__name__)
app.secret_key = os.urandom(24)
PEOPLE_FOLDER = os.path.join('static', 'images')
app.config['UPLOAD_FOLDER'] = PEOPLE_FOLDER
@app.route("/")
def FUN_root():
return render_template("index.html")
@app.route("/index.html")
def logout():
return render_template("index.html")
# @app.route("/upload.html")
# def upload():
# return render_template("upload.html")
@app.route("/register.html")
def reg():
return render_template("register.html")
```

```
@app.route("/login.html")
def login():
return render_template("login.html")
# @app.route("/upload.html")
# def up():
#     username = session['username']
#     data = view_vit(username)
#     return render_template("upload.html" , data = data)
@app.route("/viewdata.html")
def up1():
return render_template("viewdata.html")
# -----register-----
@app.route("/regact", methods = ['GET','POST'])
def registeract():
if request.method == 'POST':
id="0"
status
user_reg(id,request.form['username'],request.form['password'],request.form['email'],request.form['mobile'],request.form['address'])
if status == 1:
return render_template("login.html",m1="sucess")
else:
return render_template("register.html",m1="failed")
@app.route("/Vitaminact", methods = ['GET','POST'])
def Vitaminact1():
if request.method == 'POST':
username=session['username']
status
vit_act(username,request.form['vita'],request.form['vitb'],request.form['vite'],request.form['vitd'],request.form['vite'],request.form['vitek'])
#data = view_vit(username)
print("dfdfdfdfdfdfdfdfdfdfdfdfdfdfdfdfdfdfdfdfdfdfdf")
print(status)
if status[0]==1:
vara="Deficient"
else:
vara="In Range"
if status[1]==1:
varb="Deficient"
else:
varb="In Range"
if status[2]==1:
varc="Deficient"
else:
varc="In Range"
if status[3]==1:
vard="Deficient"
else:
```

```
vard="In Range"
if status[4]==1:
vare="Deficient"
else:
vare="In Range"
if status[5]==1:
vark="Deficient"
else:
vark="In Range"
reco=get_recomendations(status[6])
recom=getdisease_recomendations(status[7])
print(status[6])
print(recom[0])
if status[6]==1:
full_filename = os.path.join(app.config['UPLOAD_FOLDER'], 'fd.jpg')
if status[6]==2:
full_filename = os.path.join(app.config['UPLOAD_FOLDER'], 'fcvmd.jpg')
if status[6]==3:
full_filename = os.path.join(app.config['UPLOAD_FOLDER'], 'cereals.jpg')
if status[6]==4:
full_filename = os.path.join(app.config['UPLOAD_FOLDER'], 'dryfruits.jpg')
if status[6]==5:
full_filename = os.path.join(app.config['UPLOAD_FOLDER'], 'fruits.jpg')
if status[6]==6:
full_filename = os.path.join(app.config['UPLOAD_FOLDER'], 'vegetables.jpg')
if status:
return
render_template("upload.html",m1="sucess",first=vara,second=varb,third=varc,fourth=vard,fifth=vare,sixth=vark,seventh=reco[0],eight=recom[0],user_image = full_filename)
else:
return render_template("userhome.html",m1="failed")
#-----Login-----
@app.route("/loginact", methods=['GET', 'POST'])
def useract():
if request.method == 'POST':
status = user_loginact(request.form['username'], request.form['password'])
print(status)
if status == 1:
session['username'] = request.form['username']
return render_template("userhome.html", m1="sucess")
else:
return render_template("login.html", m1="Login Failed")
#-----Upload Image-----
# @app.route("/upload", methods = ['GET', 'POST'])
# def upload():
# if request.method == 'POST':
# id="0"
```

```
# status = user_upload(id,request.form['name'],request.form['image'])
# if status == 1:
#     return render_template("upload.html",m1="sucess")
# else:
#     return render_template("upload.html",m2="failed")
#-----View Images-----
@app.route("/viewimage.html")
def viewimages():
data = user_viewimages(session['username'])
print(data)
return render_template("viewimage.html",user = data)
@app.route("/track")
def track():
username = request.args.get('username')
vita = request.args.get('vita')
vitb = request.args.get('vitb')
vitic = request.args.get('vitic')
vitd = request.args.get('vit')
vite = request.args.get('vite')
data = vit_info(username,vita,vitb,vitic,vitd,vite)
print("dddddddddddddddddddddddddddd")
print(data)
print("dddddddddddddddddddddddddddd")
print(data)
return render_template("viewimage.html",m1="sucess",data=data)
#-----Track-----
# @app.route("/track")
# def track():
#     name = request.args.get('name')
#     iname = request.args.get('iname')
#     data = image_info(iname)
#     print("dddddddddddddddddddddddd")
#     print(data)
#     data = v_image(data)
#     print("dddddddddddddddddddddddd")
#     print(data)
#     return render_template("viewdata.html",m1="sucess",users=data)
#-----Update Item-----
if __name__ == "__main__":
app.run(debug=True, host='127.0.0.1', port=5000,use_reloader=False)
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import cufflinks as cf
import plotly
from plotly.offline import init_notebook_mode,iplob,plot
init_notebook_mode(connected=True)
```

```
cf.go_offline()
df = pd.read_csv('C:\\Users\\dileep bandela\\Desktop\\Projects\\Major\\Cancer.csv')
df.head()
df.info()
df.drop(['Patient Id'],axis = 1,inplace=True)
df['Level']
df['Level'].replace('Medium','High',inplace=True)
df['Level'].replace('High','1',inplace=True)
df['Level'].replace('Low','0',inplace=True)
df.head()
df['Level'] = pd.to_numeric(df['Level'])
df.isnull()
df.isnull().any()
import seaborn as sns
sns.heatmap(df.isnull())
plt.figure(figsize=(10,5))
sns.countplot(x='Smoking',data=df)
plt.figure(figsize=(10,5))
sns.countplot(x='ChestPain',data = df)
plt.figure(figsize=(10,5))
sns.boxplot(x='ChestPain',y='Age',data = df)
plt.figure(figsize=(10,5))
sns.boxplot(x='Smoking',y='Age',data = df)
sorted_smokers = df.groupby('Age')['Smoking'].count().to_frame()
sorted_smokers.style.background_gradient(cmap = 'Reds')
df.style.background_gradient(cmap = 'Reds')
label = df.Age.sort_values().unique()
target = sorted_smokers.Smoking
print(label)
print(target)
import plotly.graph_objects as go
fig = go.Figure()
fig.add_trace(go.Bar(x=label,y=target))
fig.update_layout(title = 'Smokers per age',xaxis=dict(title='Age'),yaxis=dict(title='Smokers'))
fig.show()
fig = go.Figure()
fig.add_trace(go.Scatter(x=label,y=target,mode='markers+lines'))
fig.update_layout(title = 'Smokers per age',xaxis=dict(title='Age'),yaxis=dict(title='Smokers'))
fig.show()
# Machine Learning
## RandomForest Classifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import log_loss, f1_score
from sklearn.model_selection import cross_val_score
import numpy as np
acc_dict = {}
```

```
# create the data
X = df.drop('Level',axis = 1)
y = df['Level']
X_train, X_test, y_train, y_test = train_test_split(X,y)
from sklearn.ensemble import RandomForestClassifier
# create model
model = RandomForestClassifier()
# fit the data in the model
model.fit(X_train,y_train)
y_pred_randomF = model.predict(X_test)
print('Accuracy score : ',accuracy_score(y_test, y_pred_randomF)*100)
acc_dict['RFC_log_loss'] = log_loss(y_test, y_pred_randomF)
acc_dict['RFC_F!1_Score'] = f1_score(y_test, y_pred_randomF,average='weighted')
# prediction visualization
plt.imshow(np.log(confusion_matrix(y_test,y_pred_randomF)),cmap = 'Blues',interpolation = 'nearest')
plt.ylabel('True')
plt.xlabel('Predicted')
plt.show()
## KNeighbourClassifier
from sklearn.neighbors import KNeighborsClassifier
# to find the best k
score = 0
scores, highscore, bestk = 0, 0, 0
for k in range(3,12):
knn = KNeighborsClassifier(n_neighbors=k)
scores = cross_val_score(knn, X_train, y_train)
score = scores.mean()
if score>highscore:
highscore = score
bestk = k
print('Best k is {} with score {}'.format(bestk, highscore))
knn = KNeighborsClassifier(n_neighbors=bestk)
knn.fit(X_train,y_train)
# prediction
y_predict = knn.predict(X_test)
print('Accuracy score : ',accuracy_score(y_test,y_predict)*100)
acc_dict['KNN_log_loss'] = log_loss(y_test, y_predict)
acc_dict['KNN_F!1_Score'] = f1_score(y_test, y_predict,average='weighted')
# prediction visualization
plt.imshow(np.log(confusion_matrix(y_test,y_predict)),cmap = 'Blues',interpolation = 'nearest')
plt.ylabel('True')
plt.xlabel('Predicted')
plt.show()
## Tree Classifier
from sklearn.tree import DecisionTreeClassifier
tree_ = DecisionTreeClassifier()
tree_.fit(X_train,y_train)
```



```

y_pred = tree_.predict(X_test)
print('Accuracy score : ',accuracy_score(y_test, y_pred)*100)
acc_dict['Tree_log_loss'] = log_loss(y_test,y_pred)
acc_dict['Tree_f!1_score'] = f1_score(y_test,y_pred)
# prediction visualization
plt.imshow(np.log(confusion_matrix(y_test,y_pred)),cmap = 'Blues',interpolation = 'nearest')
plt.ylabel('True')
plt.xlabel('Predicted')
plt.show()
## SVM
from sklearn.svm import SVC
model = SVC()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print('Accuracy score : ',accuracy_score(y_test, y_pred)*100)
acc_dict['svc_log_loss'] = log_loss(y_test,y_pred)
acc_dict['svc_f!1_score'] = f1_score(y_test,y_pred)
# prediction visualization
plt.imshow(np.log(confusion_matrix(y_test,y_pred)),cmap = 'Blues',interpolation = 'nearest')
plt.ylabel('True')
plt.xlabel('Predicted')
plt.show()

```

10. OUTPUT

Provide the final trust assessment results, including trust values between users and user rankings.



Figure : 10.1

All End Users


User Image	User Name	Mobile	E-Mail	Address	Status
	Ramesh	9535866270	Ramesh.123@gmail.com	#7827,4th Cross,Rajajinagar	Authorized
	Kannan	9535866270	Kannan.123@gmail.com	#8928,4th Cross,Vijayanagar	Authorized
	Ashok	9535866270	Ashok.123@gmail.com	#8928,8th Cross,Malleshwaram	Authorized
	Manjunath	9535866270	tmksmanju13@gmail.com	#8928,4th Cross,Malleshwaram	Authorized

Figure : 10.2

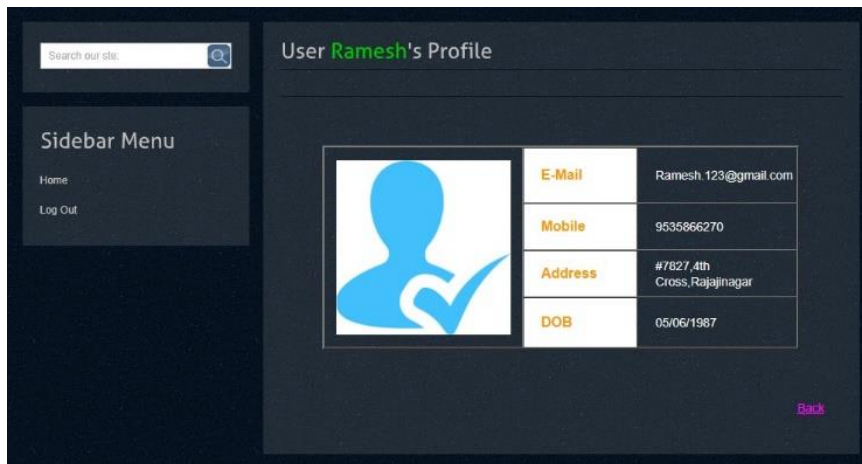


Figure : 10.3



Figure : 10.4

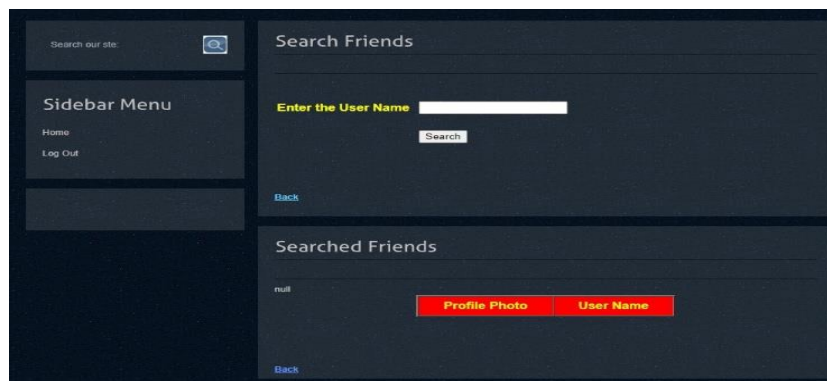


Figure : 10.5



Figure : 10.6

11. CONCLUSION

In this project, the three-valued subjective logic is proposed to model and compute trust between any two users connected within OSNs. 3VSL introduces the uncertainty space to store evidence distorted from certain spaces as trust propagates through a social network, and keeps track of evidence as multiple trusts combine. We discover that there are differences between distorting and original opinions, i.e., distorting opinions are so unique that they can be reused in trust computation while original opinions are not. This property enables 3VSL to handle complex topologies, which is not feasible in the subjective logic model.

Based on 3VSL, we design the AT algorithm to compute the trust between any pair of users in a given OSN. By recursively decomposing an arbitrary topology into a parsing tree, we prove AT is able to compute the tree and get the correct results. An open issue to 3VSL and Opinion Walk is how to estimate the value of evidences. This issue is further studied and addressed by probabilistic graphic models or neural network models [114].

We validate 3VSL both in experimental evaluations. The evaluation results indicate that 3VSL is accurate in modeling computing trust within complex OSNs. We further compare the AT algorithm to other benchmark trust assessment algorithms. Experiments in two real-world OSNs show that AT is a better algorithm in both absolute trust computation and relative trust ranking.

12. FUTURE SCOPE

The future entails refining the 3VSL model to capture evolving trust dynamics accurately. Improving the scalability and efficiency of the Assess Trust algorithm for large-scale OSNs. Integrating machine learning techniques to enhance adaptive trust prediction. Addressing privacy and security concerns to ensure user protection. Exploring diverse applications beyond traditional domains such as marketing and security. Deploying pilot implementations for real-world evaluation in collaboration with industry partners. Engaging with the research community to establish standards and best practices for trust assessment in OSNs.

13. REFERENCES

- [1] G. Liu, Q. Yang, H. Wang, and A. X. Liu. Three-valued subjective logic: A model for trust assessment in online social networks. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2019.
- [2] Anirban Basu, Jaideep Vaidya, Juan Camilo Corena, Shinsaku Kiyomoto, Stephen Marsh, Guibing Guo, Jie Zhang, and Yutaka Miyake. Opinions of people: Factoring in privacy and trust. *SIGAPP Appl. Comput. Rev.*, 14(3):7–21, September 2014.
- [3] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.
- [4] De-Nian Yang, Hui-Ju Hung, Wang-Chien Lee, and Wei Chen. Maximizing acceptance probability for active friending in online social networks. In *19th ACM SIGKDD*, pages 713–721, 2013.
- [5] Dapeng Wu, Junjie Yan, Honggang Wang, Dalei Wu, and Ruyan Wang. Social attribute aware incentive mechanism for device-to-device video distribution. *IEEE Transactions on Multimedia*, 19(8):1908–1920, 2017.
- [6] T. Cheng, G. Liu, Q. Yang, and J. Sun. Trust assessment in vehicular social network based on three-valued subjective logic. *IEEE Transactions on Multimedia*, 21(3):652–663, March 2019.
- [7] G. Liu, Q. Chen, Q. Yang, B. Zhu, H. Wang, and W. Wang. Opinionwalk: An efficient solution to massive trust assessment in online social networks. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, May 2017.