PRIVACY AND UTILITY-BASED DECISION MAKING FOR SHARING POSTS IN ONLINE SOCIAL NETWORKS

by

Tarık Berkant Kepez

B.S., Computer Engineering, Bogazici University, 2015

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2017

PRIVACY AND UTILITY-BASED DECISION MAKING FOR SHARING POSTS IN ONLINE SOCIAL NETWORKS

APPROVED BY:

Prof. Pınar Yolum (Thesis Supervisor)	....
Assoc. Prof. Arzucan Özgür	....
Assoc. Prof. Murat Şensoy	....

DATE OF APPROVAL: 29.11.2017

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitudes to my thesis supervisor, Prof. Pınar Yolum Birbil, for her endless wisdom, support, patience and guidance. It has been a great honor for me to work with her during this process. I would like to thank to my jury members, Assoc. Prof. Arzucan Özgür and Assoc. Prof. Murat Şensoy for their insightful comments and suggestions.

I would like to thank my friends and colleagues in Artificial Intelligence Laboratory: Nadin Kökciyan, Nefise Gizem Yağlıkçı, Gerçek İlke Gültekin, Can Kurtan and Mert Tiftikçi for their support, academic discussions and coffee breaks. I deeply thank to my friends in the computer engineering department who motivated me to work harder by raising my spirit and providing me with their dearest friendship, namely, Taha Yusuf Ceritli, Çağatay Yıldız, Emre Erdoğan, Nezihe Pehlivan, Setenay Ronael, Necati Cihan Camgöz, Mehmet Burak Kurutmaz and Hazal Koptagel for their support and motivating nerdy talks.

I have been fortunate enough to meet wonderful people at Boğaziçi. It is not possible to mention everyone here, but I would like to thank Çağatay Yurdasal, Mustafa Kaba, Hande Sodacı, Betül Uysal, Seray Yüksel, Şebnem Gelmedi and Muhammed Tel for letting me share all the happiness and stress of my graduate years. I would like to thank Asena Ayça Özdemir for her endless and unconditional patience, support and understanding. I would also like to thank Hazel Tanrıkulu and Zuhal Duran for their support and motivational talks during this process.

I would like to express my gratitude to my beloved family, to whom this thesis is dedicated. They have been next to me whenever I needed. They made me -and everything I achieved- possible.

ABSTRACT

PRIVACY AND UTILITY-BASED DECISION MAKING FOR SHARING POSTS IN ONLINE SOCIAL NETWORKS

The use of online social networks is growing rapidly. With this rapid increase, preserving privacy of users is becoming harder and harder. Typically, social networks address the privacy problem by asking users to define their privacy constraints up front. However, many times deciding on whom to show a post is dependent on the post itself and its context. Hence, users are forced to configure each post specifically, which is both cumbersome and prone to error. Accordingly, this study first proposes an approach that suggests privacy configurations for each post. The suggestions are based on learning from users' previous posts and configurations. However, when the user does not have many previous posts, recommendations need to take other information into account. We propose a multiagent system architecture where agents of the users consult other users' agents about possible privacy rules they can take into account. In addition to privacy-based decision making, we also propose another approach that considers users' utility of sharing each post since users also regard its benefits when they decide to share. The system aims to estimate benefits such as the number of likes that a post gets, the number of comments that it receives and how many times it is shared again. These estimations are built on learning from users' previous posts and their benefits. These privacy-based and utility-based approaches are combined in order to assist users in their sharing decisions. We evaluate single-agent and multi-agent privacy-based approaches on a benchmark dataset that we created based on content from Flickr and Reuters while we evaluate the latter one on a dataset that is collected with our Facebook application.

ÖZET

ÇEVRİMİÇİ SOSYAL AĞLARDA GÖNDERİ PAYLAŞIMINA DAİR MAHREMİYET VE FAYDA TEMELLİ KARAR ALMA

Çevrimiçi sosyal ağların kullanımı hızla artmaktadır. Bu artışla birlikte, kullanıcıların mahremiyetinin korunması her geçen gün zorlaşmaktadır. Sosyal ağlar bu mahremiyet sorununu kullanıcılara mahremiyet sınırlamalarını başlangıçta sorarak çözmeye çalışmaktadır. Ancak bir gönderinin kime gösterileceğine karar vermek çoğu zaman gönderinin kendisine ve bağlama bağlıdır. Dolayısıyla kullanıcılar her bir gönderinin gösterileceği kişileri ayarlamak zorunda kalmaktadır. Bu süreç zahmetli ve hata yapmaya açıktır. Buna uygun şekilde, bu çalışma ilk olarak her bir gönderi için mahremiyet ayarı öneren bir yaklaşım ileri sürmektedir. Bu öneriler kullanıcıların eski gönderileri ve mahremiyet ayarlarından yapılacak yapay öğrenmeye dayanmaktadır. Fakat, kullanıcının yeterince eski gönderisi olmadığında öneriler yapılırken başka bilgiler de hesaba katılmalıdır. Kullanıcı etmenlerinin hesaba katabilecekleri olası mahremiyet kurallarına dair diğer kullanıcılarınkilere danışabilecekleri çok etmenli bir sistem mimarisi önermekteyiz. Kullanıcılar gönderileri paylaşmaya karar verirken onları paylaşmanın faydalarını da değerlendirdikleri için, mahremiyet temelli karar almaya ek olarak, kullanıcıların her bir gönderiyi paylaşmasının faydasını dikkate alan bir sistem önermekteyiz. Bu sistem bir gönderiyi paylaşmanın fayda bileşenlerinden onun alacağı beğeni sayısı, ona yapılacak yorum sayısı ve yeniden paylaşılma sayısını tahmin etmeyi hedeflemektedir. Bu tahminler kullanıcıların eski gönderileri ve onların faydaları üzerinden yapılacak öğrenmeye dayanmaktadır. Kullanıcılara paylaşım kararlarında yardım etmek üzere bu mahremiyet ve fayda temelli yaklaşımlar birleştirilmektedir. Mahremiyet yaklaşımlarını kendi yarattığımız değerlendirme veri kümeleri üzerinde, sonuncuyu ise Facebook'tan topladığımız veri kümesi üzerinden değerlendirmekteyiz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a	An agent
$trusts_{a,i}$	Trust value to agent a with i support posts
x_i	ith Input features of a training instance
$x^{(i)}$	Input features of ith training instance
$y^{(i)}$	Output feature of ith training instance
$\hat{y}_i$	Predicted output value of ith training instance

LIST OF ACRONYMS/ABBREVIATIONS

ASP	Answer Set Programming
CBOW	Continuous Bag of Words
CBR	Case-Based Reasoning
ELM	Extreme Learning Machines
FFNN	Feed Forwarded neural networks
ILP	Inductive Logic Programming
IPP	isProfilePicture
IRI	Internationalized Resource Identifier
IT	Image Tags
KNN	K-Nearest Neighbor
NB	Naive Bayes
NCC	Normalized Comment Count
NLC	Normalized Like Count
NO	Normalization
NSC	Normalized Share Count
NTP	Number of Tagged People
OSN	Online Social Network
OWL	Web Ontology Language
P3P	The Platform for Privacy Preferences
PCA	Principal Component Analysis
PD	Privacy Dynamics
PEA	Policy Enforcing Agent
RF	Random Forest
RMSE	Root Mean Squared Error
SI	Social Identity
ST	Standardization
SLFN	Single Hidden Layer Feed-Forward Neural Networks
SVC	Support Vector Classification

SVM	Support Vector Machines
SVR	Support Vector Regression
SWRL	Semantic Web Rule Language
SY	Sharing Year
TE	Text Embeddings
WEKA	Waikato Environment for Knowledge Analysis

1. INTRODUCTION

Online Social Networks (OSNs) are web-based platforms, which provide their users mediums to interact with each other. In OSNs, users can build social relations, communicate with other people or legal entities such as companies, share various type of content such as text and multimedia, and consume the content produced by others. Most of current OSNs encourage, some such as Facebook even require, their users to create their on-line identity based on their real identity and to build their on-line social environment based on their environment in real-life. Thus, users' behaviors in OSNs are more attached to their real identities than other web applications. In addition to their identity information, people also disseminate their personal multimedia content, interests, locations where they have been located, events that they attend, special life-events, and so on. For example, a user may share a photo of her baby after birth, her current location, a political meeting which she attends, or a photo of her vacation. This personal dynamics of OSNs raise privacy concerns. Privacy is recognized as the right to control the dissemination of information about oneself [1]. The information in OSNs may be exploited by malevolent parties in the form of privacy attacks such as identity theft, profile porting and secondary data collection [2]. Additionally, it can abused by online stalkers and cyberbullies [3]. Even though there are no malevolent parties, it can still lead harm in people's lives. The reveal of information to unintended people has lead people to lose their jobs or their home to be robbed [4] [5] [6]. Moreover, information in OSNs can employed by universities or companies in order to screen their candidates, or to monitor behaviors of their current students or employees [7] [8] [9]. Taking these scenarios into the consideration, adequate privacy measures should be taken in order to protect the privacy of users.

The current OSNs respect the privacy of users to a degree. One of the mechanisms provided by some OSNs such as Facebook is setting audience of a post, which is the focus of our study. A post is a content, which is shared in OSN by a specific user. It may consist of text, multimedia or location information. Its audience is people who are allowed to access it. In this study, we name the preference of a user on audiences of her posts as her *privacy preferences*. The expressiveness of audience mechanisms varies from OSN to OSN. For example, when user shares a post in Facebook, she can include or exclude a set of people into or from the audience of that post. The set of people can be predefined by OSN such as friends of friends or a set defined by the user such as Bogazici University students. In Facebook, you can set the audience of a post before you share it or you can set audiences of future posts at any time. You can also modify the audience after you share the post. However, neither Facebook nor other OSNs provide any fine-grained audience mechanism which allows user to express her privacy preferences based on content of the posts. For example, it is currently not possible for a user to define the privacy rule that ”my colleagues are not allowed to see any of my vacation photos”. The user should set the audience of each post separately and manually, which brings a burden on the user as she needs to decide for each post who should be in the audience [10]. It also creates additional load on the users as well as introduces a factor of human error. For a person that shares content frequently, it is easy to forget to add or remove a certain individual from a post's audience. Even if OSNs provided that kind of fine-grained privacy preference expression opportunity, it would not be helpful since it would require user to foresee various sharing cases and to decide on it beforehand. However, recent studies show that many users are not even aware of their own privacy rules and only discover them as they experience privacy violations [11]. That is, there is a gap between users' privacy expectation and their privacy-related behaviors [12]. Hence, while enabling users to set audience for each post is useful, there are still a set of users who would not be able to set it correctly as they are not aware of their privacy rules or not aware of the privacy implications.

Consider the exemplary social graph, which is depicted in Figure 1.1: Users are Alice, Bob, Carol, Dave, Eve, and Frank. Their user ids are, respectively, 1, 2, 3, 4, 5, and 6. Bob, Carol, and Dave are friends of Alice. Eve is a friend of friend of Alice through Dave. Frank is not connected to anyone. Consider the following example based on this social graph:

Example 1: Alice travels to New York. When she visits touristic places, she likes to share them on her online social network account. She plans to visit her friends, Bob

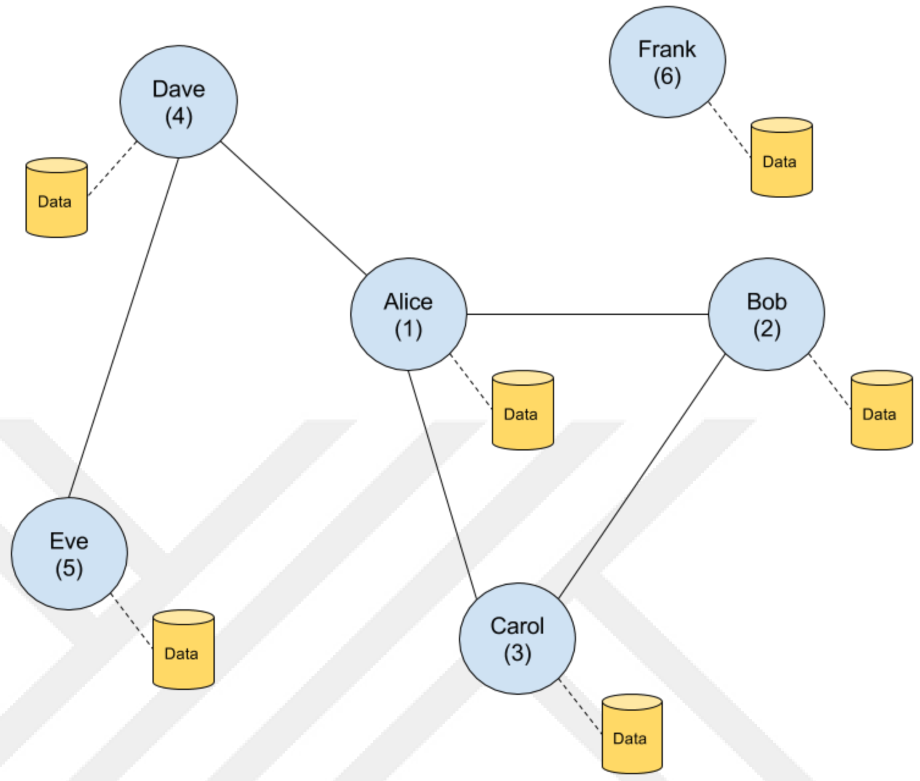

Figure 1.1. The social graph of OSN in Example 1

and Carol, who live there, too. However, she wants her visit to be a surprise. Thus, she does not want Bob and Carol to see her posts. She successfully hides various posts from Bob and Carol by removing them from the post's audience. However, she forgets to set it correctly when she visits Statue of Liberty. Both Bob and Carol see the post and the surprise is ruined.

Ideally, if Alice's software could learn over time that Alice is not sharing her recent posts with Bob and Carol and if it could warn Alice accordingly, it would be of tremendous help. We realize this by employing machine learning techniques. The idea is that Alice's posts are classified over time based on whether they are being shown to certain individuals (in this case Bob and Carol). When Alice is in a position to share a post with others, the classifier first checks whether this would cause any harm based on previous sharing behavior. If so, the system interrupts upload and asks for permission.

When a user has enough posts in the system, learning from previous posts is a good direction. However, many users face privacy problems more when they first start to use a system, since the implications of their action on the system are not clear. Consider the following example:

Example 2: Charlie is a new user of the online social network. He does not know what types of things he needs to consider to preserve his privacy. When he has a post, he would like to get help about how he should set the audience of the post.

In such cases, we need to have a software that can help users in Example 2. One way of approaching this problem is from a system's point of view. That is, if overall the system data are available, one can build clustering systems so that a user can be recommended privacy policies based on what other users are doing. However, when such data are not available, then the user is left to her own resources to figure out. In offline world, users ask each other for help. Here, we mimic a similar approach in a multiagent setting, where a user in need of privacy recommendations turns to others in the system for help. The user's agent queries other agents, collects their recommendations, and consolidates them for a final result. We study various ways to realize this.

In addition to the privacy, users also consider the financial, virtual or psychological gains of revealing their information. For sharing in OSNs, these gains are the number of likes that posts get, the number of comments that posts receive, and how many times posts are shared, which we call *utility*. These are the motivation of some of OSN users for sharing. The trade-off between privacy and utility varies from user to user. Thus, we also study a method to understand users' utility characteristics, that is, how much they expect these gains. This method requires users' data to extract some statistics about their sharing behavior. Then, it employs them to determine the attitude of users to this utility concepts. Furthermore, we study an approach which aims to estimate the utility of posts by applying machine learning techniques on users' own data.

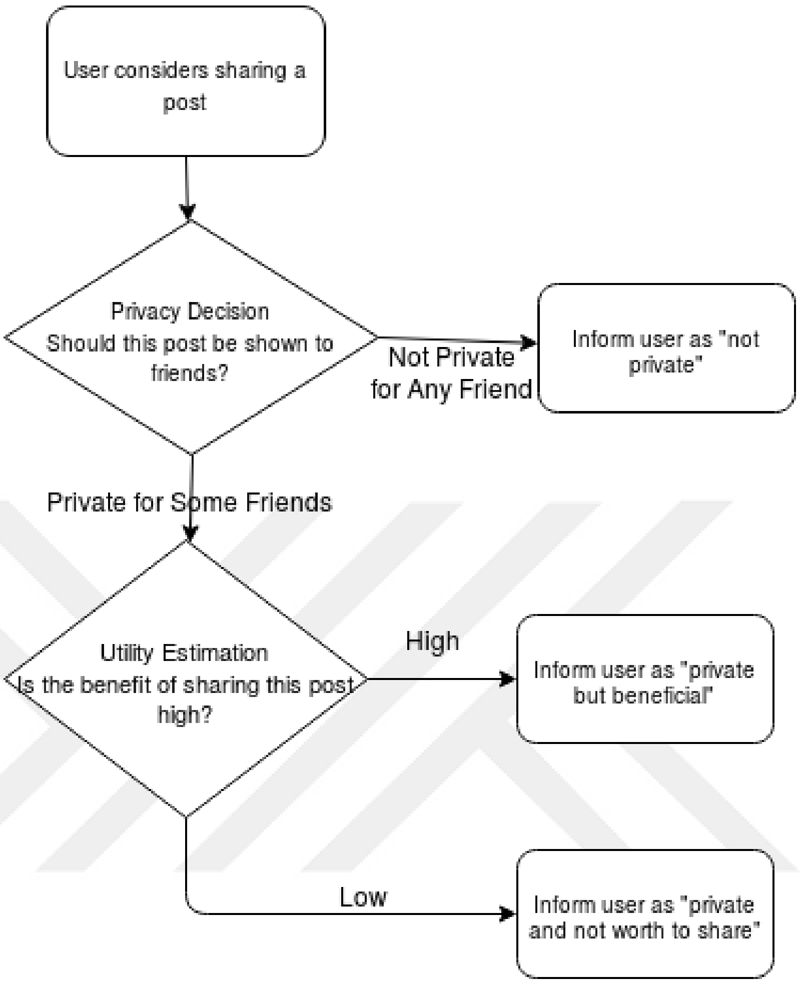
User considers sharing a post
Privacy Decision
Should this post be shown to friends?
Not Private for Any Friend
Inform user as "not private"
Private for Some Friends
Utility Estimation
Is the benefit of sharing this post high?
High
Inform user as "private but beneficial"
Low
Inform user as "private and not worth to share"

Figure 1.2. The general flowchart of our system when user attempts to share a post

Our system assists its user when she attempts to share a post. The flow of the system is depicted in Figure 1.2. First, a set of audience is suggested by our system with the help of its privacy classifier. In the case it is denied to be shown to some of user's friends, then its utility is estimated with the help of its utility estimators. At the end, the user is informed with the privacy result and, if computed, the utility result.

Chapter 2 describes our approach for suggesting privacy configurations in both single-agent and multi-agent settings. Chapter 3 explains our method to understand utility characteristics of users and the one to estimate utility of sharing. Chapter 4 describes our implementation details such as the machine learning techniques that are used in the our system, settings of experiments and their results. Finally, Chapter 5 discusses our work in relation to existing work in the literature.

2. PRIVACY DETERMINATION

We envision a system where each user is represented by a software agent. An agent is a computation that can perceive, reason, act and communicate with other agents [13]. Contrary to current social network applications, where humans are the only users of the system, we envision a system where the user actions are supported by the agent. The agent's broad aim is to help its user manage her privacy in OSNs. The user may consider her profile information, her posts, her activity logs and so on as private. We focus on the privacy of the user's posts. Thus, the specific goal of our agent is to assist its user in setting who will be able to see a post which she is publishing. For example, when a user is about to upload a photo, her agent can warn her about particular consequences (e.g., an unintended audience will see the post because the audience is set to public). The user is free to override the agent.

The agent has three main sources of information, on which it can act. The first source is the user's own privacy rules. If the user already has predefined privacy rules, then enforcing them over a post is easy. This is akin to current online social networks, such as Facebook, where the user's sharing behavior is applied to all future posts by default. The user is given an option to change it if necessary. Since this case is generally straightforward, we do not explicitly address this.

The second case is where the user might not have specified its privacy rules explicitly but has shared many posts in the system. A user that enters the system can share posts as she sees fit and she can set the audience of the posts. The audience is essentially a set of users in the system. This set can contain a predefined set of users (such as friends) or a list of users. When a predefined set is used, a user can put a restriction on the set by removing certain individuals from the set. Since we are concerned with the individuals to whom the post is not intended to be shown, the agent can track cases where a user has explicitly removed a certain individual from an audience and learn over time a privacy policy for the user.

Example 3: Following Example 1, this would correspond to Alice sharing a post of Brooklyn Bridge and restricting Bob and Carol. While other individuals are allowed to see the post, Alice's agent records the fact that Alice has disallowed Bob and Carol.

2.1. Background: Machine Learning Methods

Machine Learning techniques are well-suited for the task of learning privacy preferences of a user and her utilities of sharing posts. For privacy preferences, the idea is to learn whom a given type of post is shared with based on the audience of the post. In the case of utility, it is to estimate the number of its comments, the number of its likes and how many times it will be shared. Since the output of the privacy preference problem is a decision (allow or not to share), we approach to it as a classification problem. However, numbers are aimed to estimate in the utility cases; hence, they are regression problems. Thus, classification techniques are employed in the former one while the regression methods are used in the utility problems. The techniques which are used in our study are Decision Trees, Random Forests (RF), M5P trees, Support Vector Machines (SVM), Naive Bayes (NB), and Extreme Learning Machines (ELM). For all of the techniques except ELM and SVM, we have employed their implementations in Waikato Environment for Knowledge Analysis (WEKA), a machine learning software developed in Java by Machine Learning Group at the University of Waikato, New Zealand [14]. We have embedded Java library of Weka in the implementation of our system while it has also a desktop application with GUI, which we have only used while working on the experiments for trial and error. For SVM, Weka does not provide an implementation, but it provides a wrapper for the SVM implementation in LIBSVM, a library for SVMs in various languages including Java, which is developed by Chang *et al.* [15]. The Java implementation of ELM which we have employed has been developed by Dong Li [16].

2.1.1. Decision Trees

Decision trees are trees of which each interior node keeps a condition based on one of the features of instances, and each leaf node represents the class labels [17]. According to the conditions stored in the interior nodes, an instance follows a path starting from the root node until a leaf node, which will determine the class label of that instance. We have used ID3 algorithm [18] to generate the decision tree (in other words, to determine those conditions and class labels). It iterates through each possible condition related with each unused feature. Then, it calculates entropies of the splits resulting from those conditions and the one with the minimum entropy is added to the tree. Then, this process continues in a depth-first-search manner until one of three following conditions is satisfied: all of instances in a node have the same label, all features have been used, or there are no instances falling in that node. In the first case, a leaf node is created with the label of its instances. In the second and third case, a leaf node is created with the majority voting result of labels of, respectively, its instances and its parent's instances.

Random Forest (RF) consists of a collection of decision trees generated with random subsets of the original training data or random subsets of the input features [19]. Then, the final label is determined by majority voting of those decision trees.

ID3 algorithm and RFs are employed in the classification problems. M5P is one of the algorithms which are presented to use decision trees in the regression problems. It has been invented by Quinlan and has been improved by Wang *et al.* [20], [21]. Firstly, it creates a decision tree a similar way to ID3 algorithm; however, its goal is to minimize the variance of the instances within branches instead of the maximization of the information gain in each nodes. Then, from the bottom of the tree, leaf nodes are pruned and the non-leaves nodes are converted into the regression models, the regression planes.

2.1.2. Support Vector Machine

SVMs aim to construct a hyperplane or a set of hyperplanes, which will serve as a decision boundary during the task of classification [22]. One of the key concepts behind those hyperplanes is functional margins, which should be explained first. Given a training instance $(x^{(i)}, y^{(i)})$ the functional margin of (w, b) are defined as the following:

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x + b) \tag{2.1}$$

where w and b are the parameters of the underlying linear classifier, and $y^{(i)} \in \{-1, 1\}$. Since the parameters can be normalized, it can be intuitively interpreted as the distance between that instance and the boundary decision. Then, with respect to a training set, the functional margin of (w, b) is defined as the minimum of the functional margins of the instances in that set. That definition leads to the fact that the larger the functional margin is, the higher degree of confidence of the classifier.

SVMs aim to maximize its functional margin. This turns the problem into an optimization problem. Since that problem is not solvable by the standard optimization algorithms (due to the its non-convex structure), scaling constraints are added, which makes the problem solvable. However, in order to work efficiently with data in very high dimensional spaces, kernels are introduced, which is enabled by employing Lagrangian relaxation. Additionally, in order to handle the linearly non-separable data, the optimization problem is reformulated by using l_1 regularization. Then, the problem arising from Lagrangian relaxation is solved. Thus, the SVM model is constructed from the training data.

SVMs are also used for regression, in which case the algorithm is called Support Vector Regression (SVR). We employ ϵ-SVRs, a type of SVR. The general algorithm is similar to the SVMs mentioned above. Differently from the regular SVMs, an amount of error is tolerated in ϵ-SVRs. The amount of the error margin is the hyper-parameter

ϵ, which is set before the training. The default value of the library is 0.1, which is also used in the our application.

2.1.3. Naive Bayes

Naive Bayes classifier is one which assigns a class label to an instance based on its maximum likelihood using Bayes' rule [23]. The Bayes' rule is the following:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.2}$$

In a classification problem, the Bayes' rule is employed as:

$$P(y|x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n|y)}{P(x_1, ..., x_n)} \tag{2.3}$$

where y is the class variable and x_1 through x_n is the feature vector. In the context of our problem, y corresponds to the decision of sharing with the possible values Y (yes/allow) or N (no/deny) while x_1 through x_4 corresponds to the attributes of a post.

Naive Bayes classifier simplifies the problem by assuming that features are independent given the class variable, which results as the following:

$$P(y|x_1, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, ..., x_n)} \tag{2.4}$$

Since $P(x_1, ..., x_n)$ is constant, the classifier determines the class of a instance with this estimation:

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i|y) \tag{2.5}$$

where $P(y)$ and $P(x_i|y)$s can be estimated by employing Maximum A Posteriori estimation. Thus, the classifier calculates $P(y)$ as frequencies of classes in the training data. For $P(x_i|y)$s, it assumes various distributions.

2.1.4. Extreme Learning Machine

ELM is a learning algorithm for feed forwarded neural networks (FFNNs). It has been proposed for single hidden layer feed-forward neural networks (SLFNs) since the gradient-based learning algorithms are slow [24]. With ELM, the input weights of SLFNs are randomly assigned and the output weights are calculated by solving a linear regression problem. Because of these features, ELMs have very low training time compared to the neural networks which are trained using conventional learning methods such as backpropagation.

2.2. Privacy Determination by Single Agent

The idea then is to develop agents that can generalize whom the posts are not shown to. For generalization, our agent employs machine learning techniques, which are explained in Section 2.1. That is, it aims to build a model of the relation between the features of post, and its audience, and the allowance by exploiting the data. Since this model categorizes the data into classes, namely two classes of allowing and denying, it is called classifier.

2.2.1. Training Instance Generation

Each recorded post can be interpreted as a training data instance that is fed to one of the classifiers. That is, the training data of the classifier is the past posts which the user has posted or which has been posted to the user's page and the access rules for them determined by the user. When converting the posts to the instances for the classifier, our agent selects following attributes of posts: the type, the sharing time, the location, the location context of posts and the ids of tagged friends in them. For each post, an instance with those attributes are created. When creating instances, attributes are converted to features of instances. However, our agent employs a subset of these features (the sharing time and the location) in the privacy experiments in Section 4.1 since we cannot access the real OSN data for our experiments and we eliminate some of the features.

The type is kept as a categorical value since it may be one of predefined values such as text, photo or video. The sharing time can be employed in various ways. Its Unix timestamp, the number of seconds which have passed since Unix time 0 (January 1, 1970 UTC) can be directly used as a feature. Since a user does not presumably share any two posts at the exact same time, timestamps of all posts of the user are different numerical values from each other. Additionally, the similarity of sharing times of two posts is only based on the numerical difference of their timestamps, which does not represent most of the information kept in time. Thus, we focus on semantical representations. For example, the number of seconds, the number of minutes or the number of hours from midnight (00:00) have more meaning compared to the plain timestamp since the daily cycle has an effect on the interaction of users with OSNs. Furthermore, more canonical approaches have been employed in Ravichandran *et al.* [25] and Mugan *et al.* [26] even though they are used when the privacy of sharing location information rather than one of sharing posts is studied. In Ravichandran *et al.*'s study [25], the day is divided into a number of equal size intervals; then, the interval when location information has been shared is used as a categorical feature. The number of intervals has been decided experimentally. In Mugan *et al.*'s study [26], users' sharing time of their location has been examined and a sharing time window, which represents the longest time interval in which users share their location more than its 80%, has been defined for each user. Then, the sharing time's being or not in the sharing time window of the user is used as a binary feature of instances of that user. We do not employ those two approaches since the former approach requires beforehand decision of number of intervals and the placement of intervals; thus, it is more data-dependent and the latter one is more compatible with information frequently shared such as the location information than sharing posts.

The location information can be used in various ways, too. If the OSN provides the latitude and the longitude of the location, then they can be employed as numerical features. However, the closeness between two locations does not necessarily signal similarity. Another possibly is to use location information in various granularities. That is, the name of country, the name of city, the name of place and so on can be used as categorical values. We select the country name as our location-related feature. For the posts with no location information, a predefined special value representing no location is assigned to this feature, which is *NO_LOCATION*. The location information can be also canonicalized with the values such as *home* and *work*. In [26], each location information has been mapped to *home*, *work* and *other*, that are, respectively, chosen as the place where the most of the data has been gathered, the one where the second most of the data has been gathered, and the rest. We do not adopt this approach since this meaningful location information is provided by neither the current OSNs nor their users and this canonicalization itself can be a privacy problem for the users.

The information of tagged friends in a post can be directly employed by assigning ids of those friends as a categorical feature for instances. By taking advantage of the friend lists of users or the connection types in the social graph of the OSNs having different type of connections, this information can be canonicalized such as *friend*, *relative* and so on, which offers a higher level interpretation of similarities between friends. This again can be kept as a categorical feature in instances. However, most of the current OSNs support one type of connection such as *friend* in Facebook and *connection* in LinkedIn. That is, they do not support multiple type of connections. Additionally, they do not provide content of user-defined friend lists. Both absences of support prevent us from accessing these high level relations. Thus, we employ ids of tagged friends as a categorical feature. If there are multiple tagged users, then multiple instances are generated for each of them.

Additionally, using the access rules, for each friend of the user, each instance is reproduced and their ids are added as a feature. Whether a friend is allowed or not to access to a post is added into the instance as its class label. The allowing and denying are, respectively, represented as Y and N. Since there are two classes of instances, that is, two types of labels, this machine learning problem is a binary classification problem. At the end of this conversion process of the user's past posts to instances, the training data of the classifier is ready.

Example 4: In Example 1, Alice shares a post about her visit of Statue of Liberty. This post has the following attributes: text as its type, 05/15/2015 13:12:25 as its sharing time, and *Statue of Liberty National Monument, New York, USA* as its location information. When converting it to a machine learning instance, its type is assigned as *text*, which is one of the values of this categorical feature. Depending on the method chosen, the sharing time can be converted into the following values: 1431695545 as the plain timestamp, 47545 as the number of seconds from the midnight, and 792 as the number of minutes from the midnight. Our system chooses the feature of the number of seconds from the midnight, thus the value 47545. Depending on the method preferred, the location information of the post can be converted into the following values: (40.6892534, -74.0466944) as the tuple of latitude and longitude values, USA as country name, New York City as city name, Statue of Liberty National Monument as the place name, and *other* as the canonicalized location attribute. Except the geographical coordinates being numerical features, other ones are the categorical features. In our system, the location feature is assigned as USA since we choose the country name for this feature. Since no friend is tagged in this post, we may assign *NO_TAGGED_FRIEND* to this feature. Thus, our instance is "text, 47545, USA, NO_TAGGED_FRIEND" in the format of "type, time, location, tagged friends". Then, this instance is reproduced for each friend of Alice, which results in the following instances with the format of "type, time, location, tagged friends, id of audience, privacy decision":

(i) text, 47545, USA, NO_TAGGED_FRIEND, 2, N
(ii) text, 47545, USA, NO_TAGGED_FRIEND, 3, N
(iii) text, 47545, USA, NO_TAGGED_FRIEND, 4, Y

Note that instances are created only for the connections of Alice. They are Bob, Carol, and Dave, which can be seen in the Figure 1.1. Eve and Frank are not included in the instances since they are not connected to Alice. Additionally, the privacy decisions reflect that Bob and Carol are not allowed to access to the post while Dave can access.

2.2.2. Instance Generation of a New Post

When the user shares a post, our system intervenes the sharing process. The post to be shared is converted to corresponding machine learning instances in similar way to the conversion of the past posts of the user. The type feature is assigned directly as in the one of the past posts. The feature of sharing time is also handled in the same way as the sharing time of the past posts. If it is chosen as a numerical feature, all possible values of this attribute of the new post can be handled by the classifier. If it is decided as a categorical feature (e.g., the intervals of a day), any possible value of one of the new post cannot be out of domain since all time values in a day fall into one of those intervals. Regardless of which one of above methods is chosen, the classifier is informed about all possible values of the feature of sharing time. However, the feature of location information needs a little bit different conversion process unless the canonicalization method is preferred. With this method, the location feature is again a categorical feature with few possible values. Thus, it can be handled in the same way as the training posts are handled. However, in our method, where only location-related feature is the country name, we can define only the values in the training data or all country names to the classifier. The former one is not able to handle the values, which are not encountered in the training data. It is very unlikely that all possible values exist in the training data. The latter option may lead sparsity in the data with respect to this feature when the user is not sharing posts from various countries a lot. When the more fine-grained location information such as the city name is preferred in the system, this sparsity grows. The more granular the location information is, the more sparse the data is with respect to this feature. When the place name is selected for this feature, the values which are unknown at the training time may be encountered in the new post since it is hard to collect all existing place names and it is also temporally impossible due to openings of new places. Due to the severity of foreseeing and collecting all possible values, and this sparsity, we employ the former one by modifying it slightly in the following way: If it encounters a post with a location value to the classifier, it assigns to this feature a predefined special value, *UNKNOWN_PLACE* in our method. At the end of the conversion of those features, an instance with these features is generated. Then, for each friend of the user, this instance is reproduced and their ids are added as a feature. At that point, their labels, that is the sharing decision, are still missing. Thus, those instances are considered as test instances. Our agent labels those instances with the help of its classifier. Then, it suggests people who should be denied access to the post based on the instances labeled as *not share*. Then user decides the final access rule.

Example 5: In Example 3, Alice shares a post about her visit to Brooklyn Bridge. This post has the following attributes: text as its type, 05/15/2015 21:12:32 as its sharing time, and *Brooklyn Bridge, New York, USA* as its location information. When converting it to an instance, its type is again assigned as *text*. Since we choose the method of the number of seconds from midnight, the sharing time is converted into the following value: 76352. The country name in location information of the post is USA. Since it is encountered in the training data, it is stored as USA. Since no friend is again tagged in this post, we assign *NO_TAGGED_FRIEND* to the tagged friend feature. Thus, our instance is "text, 76352, USA, NO_TAGGED_FRIEND" in the format of "type, time, location, tagged friends". Then, this instance is also reproduced for each friend of Alice, which results in the following instances with the format of "type, time, location, tagged friends, id of audience, privacy decision":

(i) text, 76352, USA, NO_TAGGED_FRIEND, 2, ?
(ii) text, 76352, USA, NO_TAGGED_FRIEND, 3, ?
(iii) text, 76352, USA, NO_TAGGED_FRIEND, 4, ?

Since we consider the sharing of a new post this time, we do not know who should be able to access it before sharing it. Thus, the privacy decision labels are represented with a question mark. These instances are then sent to the privacy classifier of the agent for labeling.

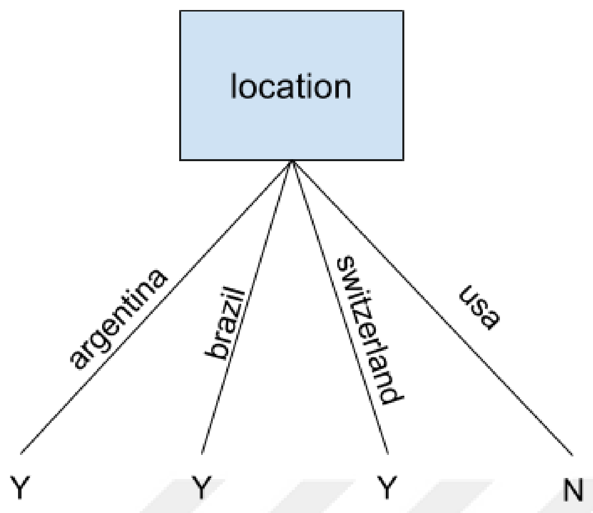

Figure 2.1. The decision tree which can be learned for Example 1

Figure 2.1 visualizes the decision tree, which has been learned with the ID3 algorithm during the first of the single-agent experiments, which are explained in 4.1.2 in detail. In the learning process, the tree has been trained with the training instances that have been generated from the user's past posts with the procedure explained above. This tree has a root node, which represents a decision for location attribute. The possible values for that attribute are ”argentina”, ”brazil”, ”switzerland”, and ”usa”, which are represented as branches. Note that other possible values are omitted in the figure to make it simpler. Then, following each branch, there may be a non-leaf or leaf node in general. In this case, there are leaf nodes, which corresponds to the access decisions (Y or N) for a post. Before a new post will be shared, it is converted into a machine learning instance with the procedure explained above. Then its instance is sent to the decision tree. Its location attribute is checked. According to its value, a branch is followed. The leaf at the end of that branch determines the sharing decision for that new post.

2.3. Privacy Determination with the Help of Multiple Agents

As mentioned before, user might not have shared many posts in the system, which is possible if the user is a new user in the OSN or the user rarely shares posts.

In that case, making a decision only based on the user's past posts will not yield accurate results. Accordingly, the agent can make use of the other agents' knowledge in the system to reach a decision. One approach to realize this could be if the other agents share their privacy rules with the agent. Then, the agent can aggregate that information and create a privacy policy for itself. However, sharing of privacy policies with others is another type of privacy violation for the other agents. Thus, many users would be reluctant to share that. However, if an agent is asked to classify a given post using her privacy rules, it would not lead to privacy violation since the reasons would still be hidden. Thus, here we opt for such an approach. That is, the agent who needs help to set the privacy setting of its post queries a selected set of agents in the system with the post to ask them to classify the post as Y or N. The agent then retrieves the results and makes a decision. In this process, the set of agents who are queried can be selected in various ways and their responses can be combined in various ways too.

We can select the set of agents who are queried in various ways. One straightforward way would be to ask all agents in the system. This approach is not computationally efficient due to the huge number of users in the current OSNs. For example, Facebook had 1.28 billion daily active users on average in March 2017 [27]. Another straightforward way is to ask agents of all connections of the user in OSN. This method is not computationally as costly as the previous one since users' direct connections are not too many to cause a great burden on computation. For example, a user had an average around 190 friends in Facebook [28]. However, it involves the agents of the unrelated connections, too. To focus only on a subset of agents, trust can be employed. Based on their past interactions with the agent needing help or their users' profile similarity with the user of the agent, a trust concept can be established. Then, the untrustworthy agents, in which the agent's has a trust lower than a predefined threshold, are excluded from the set of agents who are queried. The trust values may be dynamic or static. That is, the trustworthinesses of those agents can be calculated each time when the agent needs help in sharing a new post or they can be calculated at the first usage of the system and the addition of a new connection, thus one of a new agent. This method requires a well-structured trust computation method and it may be computationally efficient or not depending on the computation of trust.

The responses of those agents, which are selected with one of the approaches mentioned above, are combined in various ways. A straightforward approach would be to use majority voting, where every agent's vote (response in our case) is equally weighted. Then the result would be the decision of more than half of those agents. Other voting mechanisms such as two-thirds vote can be employed, too. The average of responses can be employed in order to combine them by considering the responses Y and N, respectively, as 1 and 0. Then, the average of responses is compared to a threshold, possibly *0.5*, the middle of the interval. In this way, this averaging is equivalent to the majority voting. Since these approaches treat all agents in the same manner and they are also simplistic, we do not prefer them. To differentiate the responses of agents, a weighted average can be employed. Trust can be employed in this step by considering trustworthiness values of agents as the weights of their responses.

The computation of trustworthiness of agents can be approached differently. The similarity between the agent's user and the users of the other agents can be examined. The similarity can be extracted from their profile information such as the age, the educational level since these demographic information may give a hint about their privacy expectations. Another way to examine similarity is to focus on their privacy behavior. That is, the similarity between users in terms of the privacy attitude can be explored by comparing their privacy decisions on the same posts. Thus, our agent initially sends the selected agents a fair amount of its (user's) posts whose privacy decisions are already known by the agent itself. We call those posts *support posts.* From the machine learning perspective, they can be considered instances with class labels. However, when sending other agents support posts, the privacy decisions are removed so that other agents do not know the correct answer and it is harder for them to understand that they are being tested.

```
Algorithm 1: CollectiveDecision($owner$,$n$,$ins$)
   Input: $owner$, the post owner
   Input: $n$, the max number of support posts of $owner$
   Input: $ins$, the instance to be classified
   Output: $results$, the labels of $ins$ for the cases with 1 to $n$ support posts
 1 $A \leftarrow owner$.getConnections();
 2 $trusts \leftarrow$ calculateTrust($owner$,A);
 3 $responses \leftarrow$ initList($A.length$) ;
 4 $results \leftarrow$ initList($n$) ;
 5 foreach $a$ in $A$ do
 6     $responses[a] \leftarrow a$.classify($ins$);
 7 end
 8 for $i \leftarrow 1$ to $n$ do
 9     $totalweight \leftarrow 0$;
10     $totalresponse \leftarrow 0$;
11     foreach $a$ in $A$ do
12         $totalresponse \mathrel{+}= trusts[a][i] * responses[a]$;
13         $totalweight \mathrel{+}= trusts[a][i]$;
14     end
15     if $totalresponse \,/\, totalweight > 0.5$ then
16         $results[i] \leftarrow 1$;
17     else
18         $results[i] \leftarrow 0$;
19     end
20 end
21 return $results$
```

Figure 2.2. Collective Decision Algorithm

$$trusts_{a,i} = \begin{cases} 0 & \text{if } i \text{ equals } 0 \\ trusts_{a,i-1} + 1 & \text{if } a \text{ labels } i\text{th instance} \\ & \text{as labeled by the owner} \\ trusts_{a,i-1} & \text{otherwise} \end{cases} \tag{2.6}$$

In our system, we prefer that the agent first asks all connections of its user; then it retrieves their responses and calculates their weighted average by converting them to 0 or 1s while employing its trust to them as weights. Thus, we do not involve the trust when selecting agents, but in the aggregation of their responses.

Algorithm 1 is used by each agent for decision making. First of all, its user's connections are assigned to A (line 1).

CALCULATETRUST: Then it employs trust levels to decide rather than the majority voting (line 2). To calculate those levels, in addition to sending the post that it wants to be classified, it also sends *support post*s, which are from its user's training data, and the correct labels are known by our agent. This enables the agent to cross-check the results and establish various levels of trust among other agents. We vary the number of support posts to evaluate their effect in finding the correct labels for posts. Our agent sends various numbers of support posts to all agents. After getting their labels for those posts, the trust levels are calculated with the Equation 2.6, where i is the number of support posts and a represents the agent to whom Alice asks. In that way, the original labels of the agent and the labels by other agents are compared, which are handled by second and third conditions in the equation. These trust values, $trusts_{a,i}$, are returned directly by the method CALCULATETRUST and assigned to *trusts* matrix (line 2).

Then, *responses*, the list which stores the responses of those agents for the test instance, and *results*, the list which stores the final collective results for the instance, are initialized via INITLIST, which returns a list filled with zeros in the given length (lines 3 and 4). Since it has trust values for each agent, it first gets the responses of other agents for *ins* and stores them in *responses* (lines 5 to 7). For each case with different number of support posts (lines 8 to 20), it calculates the total weight, *totalweight*, by summing the trust values for each agent, and the total response, *totalresponse*, by summing the weighted responses of each agent for that instance, which are weighted by agents' trust values (lines 9 to 14). Following it, the agent calculates a weighted average response by dividing them and it is compared with 0.5. If it is higher than 0.5, then result in that case is assigned as 1 (YES), otherwise 0 (NO) (lines 15 to 19). At the end, *results* is returned. In order to treat YES and NO in a balanced way, we have set this threshold as 0.5. However, it does not have to be that value. It can be set higher so that the agent is more cautious in sharing content. A lower value can be also set to it so that the agent shares more liberally. It can also be tuned based on the training data.

3. UTILITY DETERMINATION

When a user attempts to share a post, she decides to do so by considering not only her privacy but also the benefit of sharing the post. These benefits are rooted in her motivations underlying this decision such as receiving likes or comments. Our system assists users decide to share a post by considering these motivations together with their privacy expectations.

The motivations behind sharing a post in OSNs are explored in the study conducted by Brett [29]. His research claims five motivations of sharing. The first motivation of users is to improve lives of those about whom the users care by presenting useful, high-quality or enjoyable posts to them. Since users are interested in these people, they want to share useful or amusing information with them. The next motivation is to express their identity to themselves and others. When sharing a post, they, whether intentionally or not, shape their personalities or the projections of their personalities in the others' perspective. Thus, they can idealize their identity via their online identity. Another motivation of sharing is to strengthen their relationship with others. Users can share about their recent thoughts, experiences, events and updates on their personal or professional life, ranging from major events such as marriage and getting a job to less important events such as promotions and anniversary celebrations. Thus, people stay updated and connected by sharing in OSNs. The next motivation is self-satisfaction. This motivation is connected with the first one. Sharing valuable items with the cared ones satisfies them and getting feedbacks about these items from them increases the satisfaction. Thus, the expectation of this kind of satisfaction motivates users to share. The last motivation is to aid causes that they favor. By sharing, they can arouse others's interest, get their support for a cause, or inform others about it. Thus, a cause is a great motivation considering the effectiveness of OSNs.

3.1. Utility Personas

Based on these motivations, Brett [29] categorizes the OSN users into six sharing personality types (personas) with respect to their sharing behaviors, which are *altruists*, *boomerangs*, *careerists*, *connectors*, *hipsters*, and *selectives*.

- *Altruists* refers to users who are more focused on people about whom they care than self-interested people are as the name suggests. They have elementarily the first motivation in the study, to present valuable content to others. Besides that, they are motivated by the causes and mostly the positive feedback from others about the content that they shared with these other users.
- *Boomerangs* is also aptly named since the responses to the content that they shared are their main consideration. Any response are preferred than no response whether or not the response is positive. Generally they are driven by the desire of approval of them by others.
- *Careerists* are users one of whose main goals are to establish personal and/or professional network in OSNs. They also like to share content with people with the aim of triggering debates and discussions. However, their interaction is generally bidirectional. That is, they are not only the sharing part of their relationships, but their relations also share content with them. They also interact frequently with the content shared by their relations.
- *Connectors* are the users who care about mutual experiences. Thus, while sharing posts, they try to involve others in this mutual experience. Additionally, they like to share fun content.
- *Hipsters* is well defined by the motivation to develop an online identity, to shape it and to keep in touch with their surroundings. This persona is underpinned by the online content sharing. Thus, they watch for opportunities to share and if possibly to share first in their social circles.
- *Selectives* represents the users who selectively share the content with others. That is, they share in a case that they think that a connection would really enjoy a content and she cannot find it herself then they share. Thus, their motivations

of sharing a post are similar to *altruists* except that they target a specific subset of their connections depending on the content of the post.

In our system, the benefits of a user from the sharing are grounded on these personas. Since the main characteristics of these personas can be reflected by the likes/reactions of posts, their sharing count, their comments, and tags of other users in them, we consider the benefit of a user from the sharing as these elements of OSNs. We quantify the benefit as utility computations based on those OSN elements. We assign one utility computation, that is, one utility function, to each persona, which are presented in Equations (3.4) to (3.9).

We define utility functions based on other basic terms. They are Normalized Comment Count (NCC), Normalized Like Count(NLC), and Normalized Share Count(NSC), which are computed as, respectively, in Equation 3.1, Equation 3.2, and Equation 3.3. Our aim of introduction of these basic terms is to keep utility values in a fixed range, which is [0,1] in our case. In NCC calculation of a post, the number of comments that the post receives or it is predicted to receive is firstly normalized by dividing it by the maximum of the numbers of comments of all past posts of the user. If the post is also a past post, then this operation normalizes the value to the range [0,1] since the number of comments of this post is less than or equal to the maximum number of comments of past posts. Otherwise the range cannot be known beforehand. If the post is predicted to receive a number of comments greater than the maximum of the past posts, the result of operation is greater than 1. To normalize the division result, the minimum of its value and 1 is calculated. The posts with number of comments less than or equal to the maximum one is not affected by the minimum operation while only the ones greater than the maximum one is behaved same as the maximum one. Considering the infrequency of exceeding the maximum comment number, it is fair to deal them same as the maximum ones. Thus, NCC values represent numbers of comments always in the range of [0,1]. Additionally, NLC and NSC normalize, respectively, the number of likes and the number of shares exactly in the same way as NCC normalizes the number of comments.

$$NCC = min(\frac{NumberofCommentsWhichAPostReceives}{max(NumberofCommentsUser'sPostsHaveReceived)}, 1) \tag{3.1}$$

$$NLC = min(\frac{NumberofLikesWhichAPostReceives}{max(NumberofLikesUser'sPostsHaveReceived)}, 1) \tag{3.2}$$

$$NSC = min(\frac{NumberofSharesWhichAPostReceives}{max(NumberofSharesUser'sPostsHaveReceived)}, 1) \tag{3.3}$$

The utility functions, which are presented in Equations (3.4) to (3.9), are designed as a linear combination of NCC, NLC, and NSC. Their coefficients are chosen in such a way that they add up to 1, which ensures the utility values are in the range [0, 1]. The utility functions, thus their coefficients, have been decided intuitively by considering the characteristics of personas. For OSNs that do not support comments, likes or sharing, our utility functions would need to be updated. In order to involve these scenarios, the weights of the elements supported can be normalized among them. For example, if the likes are not supported in an OSN, then the like term, the one with NLC, can be removed from the utility functions and the weights of NCC and NSC can be normalized. That is, the weights of NCC and NSC can be divided by their sum so that they still add up to 1.

- *Altruists* care positive feedbacks, which are best revealed with the likes. Additionally, the comments may imply positive feedbacks. The number of shares may also represent them with a slight probability. Thus, in Equation (3.4), NLC's weight is set as the largest one in the equation and the one of NCC is set as the next largest one.

$$U_{Altruist} = 0.35 * NCC + 0.60 * NLC + 0.05 * NSC \tag{3.4}$$

- *Boomerangs* like responses regardless of their attitude. Since users can express their opinions more clearly with comments than with likes or by sharing, NCC's weight is set as the largest one in the equation. Even though likes and shares are not as important as comments, they still have a role in utility. Thus, NLC's and NSC's weights are not set as 0s but low values.

$$U_{Boomerang} = 0.70 * NCC + 0.15 * NLC + 0.15 * NSC \tag{3.5}$$

- *Careerists* generally aim to start a debate. Since the likes or shares cannot contribute to debates so much, comments are the most important element for utilities of careerists. Thus, NCC's weight is set again the largest value in the equation while others' weights are very low to affect the utility value.

$$U_{Careerist} = 0.80 * NCC + 0.10 * NLC + 0.10 * NSC \tag{3.6}$$

- *Connectors* attributes importance to mutual experiences. Thus, all types of interactions are important for their sharing utility. All the weights are set almost equal to each other; however, NSC's weight is set slightly lower since sharing creates another environment to interact and it may have a slight chance to decrease the utility of the original environment where it is shared first.

$$U_{Connector} = 0.35 * NCC + 0.35 * NLC + 0.30 * NSC \tag{3.7}$$

- *Hipsters* ground their online identities on sharing. They aim to share first and to spread contents in their online environments. Therefore, the numbers of sharing of their contents are important for their utility. Hence, NSC's weight is set the highest. Since *Hipsters* care to create an online identity, other interactions also matter even though they are not as important as sharing. Thus, weights of NLC and NCC are set as lower than the one of NSC but enough to effect the utility value.

$$U_{Hipster} = 0.20 * NCC + 0.20 * NLC + 0.60 * NSC \tag{3.8}$$

- *Selectives* aim to enjoy a specific subset of users. They expect their interactions whether they are likes or comments. They do not care much about content being shared by others since they care enjoying some of their connections. Therefore, weights of NCC and NLC are set the largest ones in the equation. The weight of NSC is set so low that it slightly affects the utility value.

$$U_{Selective} = 0.45 * NCC + 0.45 * NLC + 0.10 * NSC \tag{3.9}$$

Considering the artificial nature of utility, it is not possible to exactly measure a utility value since they are not real values. Instead they have values which indicates the intensity of a utility; thus, it enables to compare utility values. Due to this, it is hard to find a way to determine utility functions besides our intuitive method based on the personas, which are defined based on a user study. However, more advanced user studies can be employed in order to define these utility functions.

When a user starts to use our system, her past posts are examined by the system with two utility-based purposes. The first one is to assign the user to a persona based on the procedure proposed in Section 3.2. The latter purpose of examination of her past posts is to learn the relation between the attributes of her posts and the utility of sharing these posts, and to employ this learned relation to predict the utility values of the future posts, which the user will share, which is explained in Section 3.3.

3.2. Persona Assignment

In order to assign a user to a persona, we intuitively aim to focus on numbers of likes, numbers of comments and other elements of her past posts. The features of posts which are employed to assign a user to a persona are as follows: We can focus on the number of likes per posts for altruists since they mainly care about the feedback and approval from others. We can employ the number of posts that they shared in other users' profile pages and the number of those that others shared on their profile pages as the base of the persona of careerists since they like to share with people with the aim of networking and sharing on others' profile pages gives the opportunity of networking. To differentiate hipsters, we can examine their frequency of sharing posts due to their great desire to share. For boomerangs, the number of comments per posts can be evaluated by our system since boomerangs share with the expectation of responses and whether the responses are positive or not does not matter for them. Our system can identify connectors by checking the number of posts which contain multimedia content such as image or video and the frequency of posts in which they tagged other users as they are interested in sharing entertaining content and they prefer to include other users in their sharing experience. For selectives, we can test the percentage of their posts in which they tagged other users since they target specific connections while sharing.

With the intent of determining a well-defined procedure based on these intuitions, we have collected the data from 40 Facebook users. The mean of these users' ages is 26.98 ± 5.46, with the minimum of 21 and the maximum of 49. 22 of users are female while the other 18 users are male. 33 users are from Turkey while the rest are from Belgium (1), Germany (1), Italy (1), Netherlands (1), United Kingdom (2) and United States of America (1). The mean of number of posts that they have shared is 778.85 ± 950.32, with the minimum and the maximum of (14, 4802). Their membership times in Facebook have a mean value of 6.48 ± 2.62 years and minimum-maximum values of 0.5-10 years. Based on these two statistics, we have measured the average number of posts shared per year. With minimum and maximum values of 2 and 480.2, it has a mean of 125.03 $\pm$ 126.31 among users. These three statistics reflect that we have a diversity in our data in terms of Facebook usage.

In order to collect these data, we have developed a Facebook Application which traverses users' posts, photos and profile pages, and computes some statistics which are rooted in our intuitions. The application also asks users to choose a persona which expresses their sharing motivations best among the personas in Brett's study [29]. The personas are explained to users with the descriptions in the original work.

The statistics we have computed for a user are following:

- number of posts: the number of posts and photos which have been shared by the user on her own profile page
- number of posts with multimedia (normalized over the number of posts): the number of posts which contains a multimedia object such as photo and the number of photos which have been shared by the user herself on her own profile page
- comments per post: the total number of comments of user's own posts and photos divided by the number of posts
- number of posts without comments (normalized over the number of posts): the number of user's own posts which do not have any comments
- comments per post with comments: the total number of comments divided by the number of posts which contain at least one comment
- likes per post: the total count of likes of user's own posts and photos divided by the number of posts
- number of posts without likes (normalized over the number of posts): the number of user's own posts which have not received any likes
- likes per post with likes: the total number of likes divided by the number of posts which have received at least one likes
- reactions per post: the total count of reactions of user's own posts and photos divided by the number of posts. The reactions are a feature of Facebook which allows users to respond a post or photo in various ways: Like, Love, Haha, Wow, Sad or Angry.

- number of posts without reactions (normalized over the number of posts): the number of user's own posts which have not received any reactions
- reactions per post with reactions: the total number of reactions divided by the number of posts which have received at least one reactions
- reactions except like per post: the total count of reactions other than like, other reactions, of user's own posts and photos divided by the number of posts
- number of posts only with like (normalized over the number of posts): the number of user's own posts which have received at least one like but no other type of reactions
- other reactions per post with other reactions: the total number of other type of reactions divided by the number of posts which have received at least one reactions other than like
- frequency of sharing posts (posts per day) (normalized over the number of posts): the number of posts shared by the user on her own profile page divided by the number of days in which the user have been registered on this OSN
- number of posts where someone has been tagged (normalized over the number of posts): the number of posts which have been shared the user on her own profile page and in which at least one user have been tagged
- number of posts shared by others on your profile (normalized over the number of posts): the number of posts which have been shared by other users on the user profile page

The number of people which have chosen Altruist, Boomerang, Careerist, Connector, Hipster and Selective, are, respectively, 21, 1, 1, 3, 1, and 13. In terms of the personas, the data are unbalanced. Additionally, the number of samples is very few to relate with some personas. Thus, we have only worked on the data of Altruists (21 people) and Selectives (13 people) in our sample of 40 people. The data of people from other personas are eliminated since they are not sufficiently informative.

Firstly, we have visualized our data, which can be seen in Figure 3.1. Since our data have more than three features, thus dimensions, we had to reduce its dimension to visualize it. Thus, we have reduced its dimension to two by applying Principal Com-

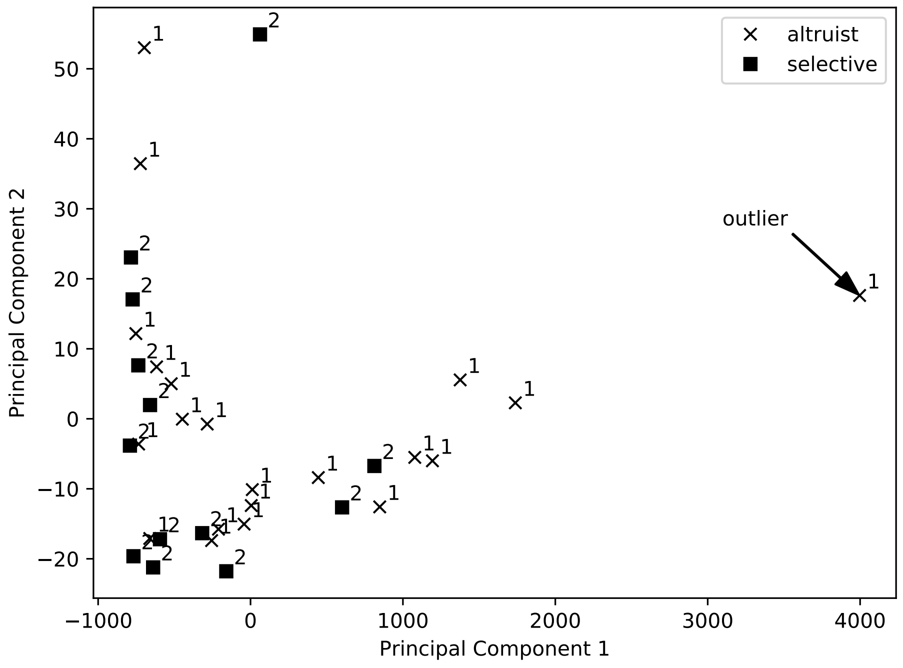

Figure 3.1. The data of Altruists and Selectives which has been reduced to 2 dimension by PCA

ponent Analysis (PCA) on it, which is a dimension reduction method which employs the Eigen vectors of the data as the base vectors. The personas Altruist and Selectives, are, respectively, represented by the following markers and numbers: x marker (1), and square marker (2). In Figure 3.1, it can be seen that there is an outlier, thus we have removed it. After this point, we will not use that outlier. Additionally, Figure 3.1 demonstrates that the data are not linearly separable in this dimension. We cannot observe any significant structure in the data either.

To understand the data more, we have clustered the data with k-means algorithm, a distance-based cluster algorithm. Since we have focused on two personas, we have set the number of clusters parameter of the algorithm to two. After clustering the data, again we have employed PCA in order to visualize it in two dimensions. The result can be seen in Figure 3.2, in which the marker type of a point represents its persona while its annotation number represents the cluster assigned by the clustering algorithm and the diamonds represent the centroids of clusters. That is, Altruists and Selectives are, respectively, represented by x and square markers while 1s and 2s are the cluster

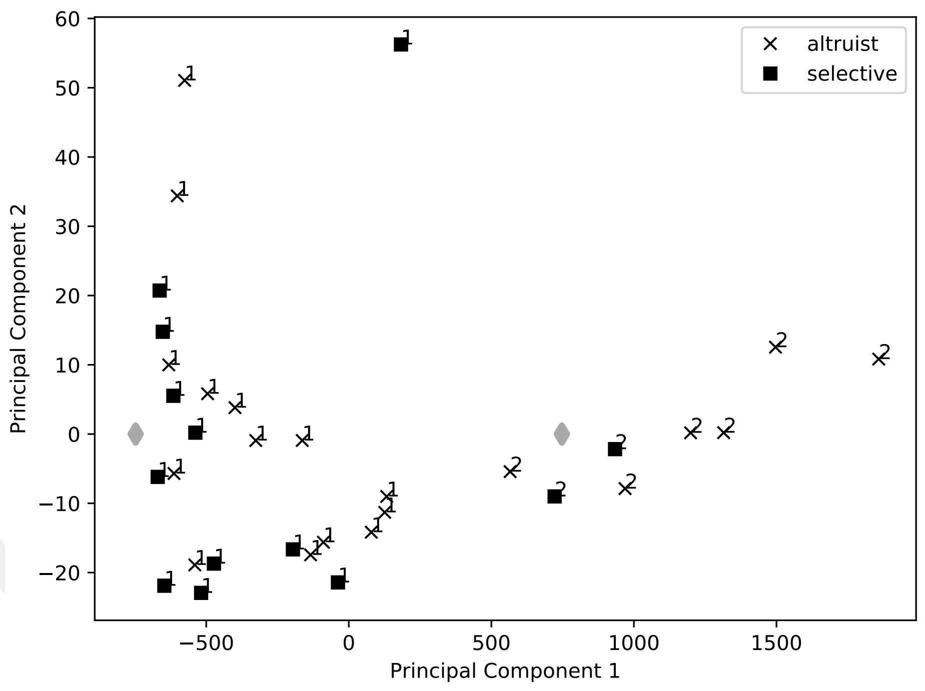

Figure 3.2. The data of Altruists and Selectives which has been clustered to 2 clusters and reduced to 2 dimension by PCA

labels of points. After clustering, clusters do not overlap the persona labels. Both clusters contain samples from different personas. For example, cluster 1 contains 14 altruists, and 11 selectives. Therefore, the persona which this cluster represents is not clear. Thus, we have employed more, namely six, clusters in order to detect smaller structures, which can be seen in Figure 3.3 again numeric annotations for clusters. However, personas were not still separated enough even though it is better than the 2-cluster case. For example, considering again cluster 1, it has four altruists and two selectives, thus its persona cannot be decided.

With the aim of finding smaller structures, we preferred a method which can represent hierarchy in data: decision trees. Another reason to pick them is that their findings are more human-readable compared to other classification methods. First of all, we have trained a decision tree with the data that we have focused on (two personas, one instance excluded, PCA not applied). This decision tree is visualized in Figure 3.4. When a new user enters to our system, the tree is supposed to determine her persona based on her statistics. In order to assign a persona, it begins by checking

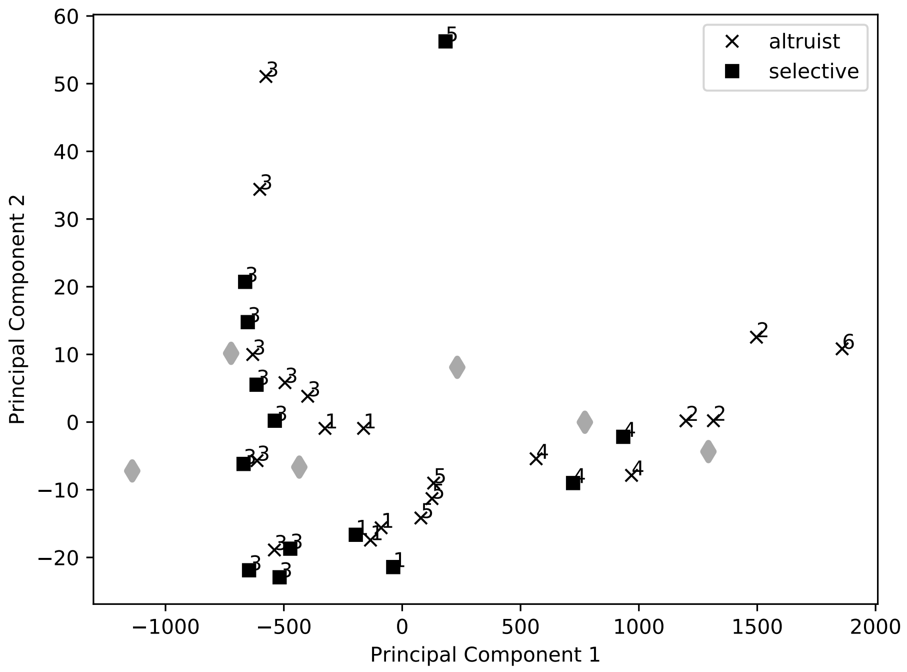

Figure 3.3. The data of Altruists and Selectives which has been clustered to 6 clusters and reduced to 2 dimension by PCA

the condition on the root node. Based on the result, it checks the conditions on the left or right child nodes. It continues until a leaf node is found. Then, class of that leaf node is assigned to that user. However, if we consider the gini impurity values, which is a measure of impurity representing the probability of incorrectly assigning a class based on a node of nodes, this tree contains one node with 0.5, the maximum value of gini impurity for two classes. Thus, this tree has a degree of uncertainty, which we want to minimize. Additionally, there is a leaf node with 17 Altruist samples. Therefore, this tree does not split the data into unbalanced sets. In order to find a better structure, we have removed these features: 'number of posts', 'number of posts without comments', 'comments per post with comments', 'number of posts without likes', 'likes per post with likes ', 'number of posts without reactions', 'reactions per post with reactions ', 'reactions except like per post', 'number of posts only with like', and 'other reactions per post with other reactions'. These features have been eliminated since they are indirectly related with our intuition-based approach while other seven features are directly related. Then another decision tree has been trained with this data, which can be seen in Figure 3.5. This tree also has two nodes with gini impurity

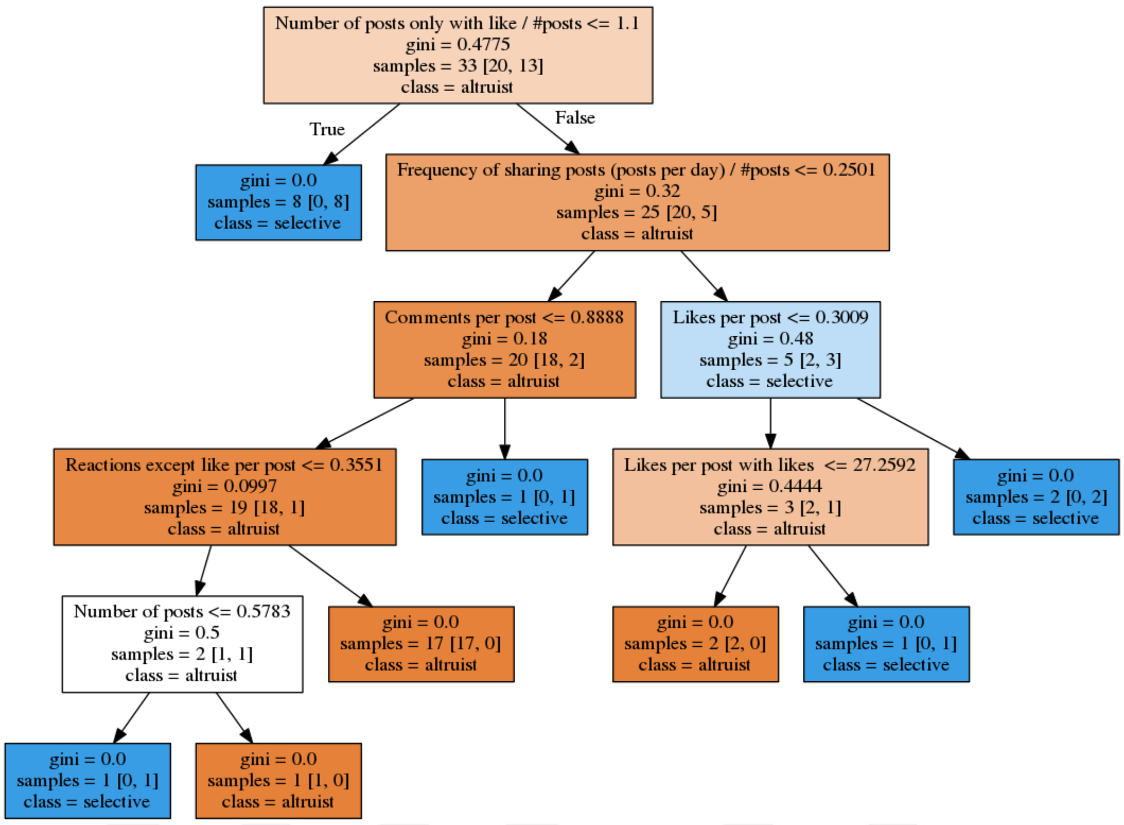

Figure 3.4. The decision tree trained with the data of Altruists and Selectives

value of 0.5. In order to find a better one and test the relation of features complied with our intuition, we have removed all features except these two: 'likes per post' and 'number of posts where someone has been tagged'. The tree trained with this data is displayed in Figure 3.6. This tree does not contain any nodes with 0.5 gini impurity. Compared to other trees, the sets represented by its leaf nodes are better balanced. The maximum number of nodes in a leaf node is seven while the maximums are 10 and 17 in the previous trees. Furthermore, among its ten discriminative intermediate nodes, five nodes supports our intuition-based rules for Altruists and Selectives. As a result, this decision tree is a better choice than others to assign a user to a persona.

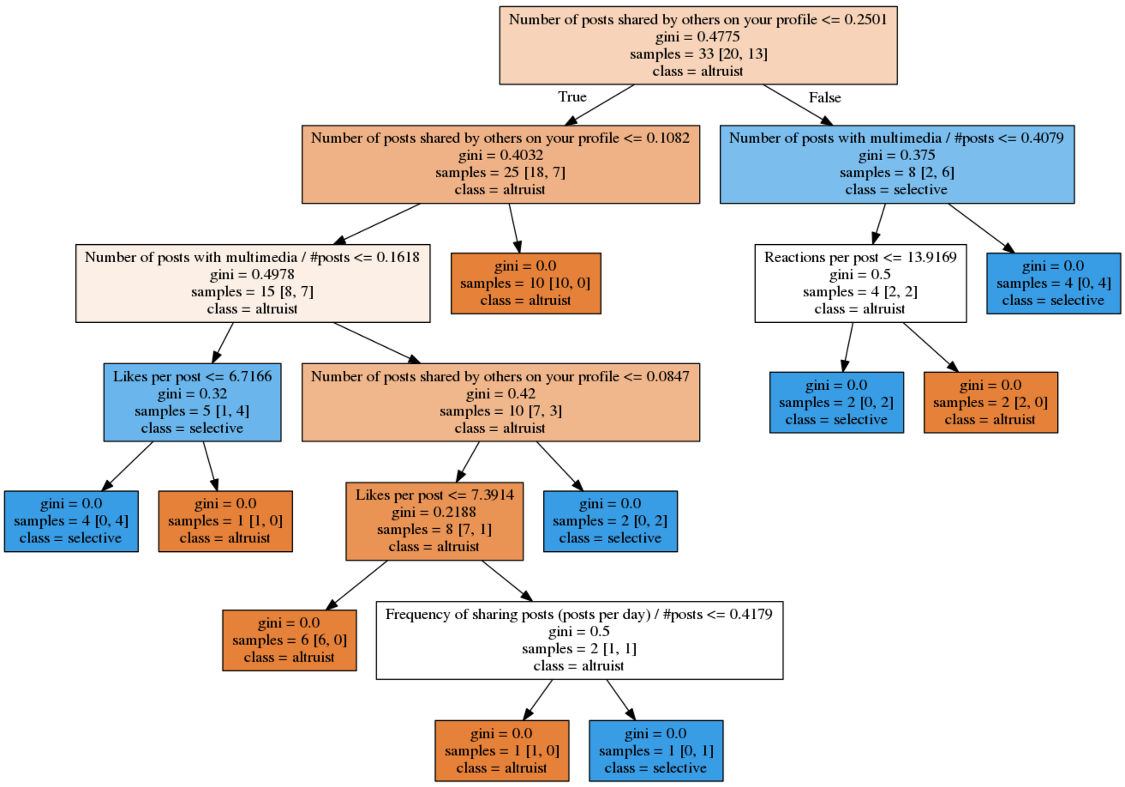

Figure 3.5. The decision tree trained with the data of Altruists and Selectives containing relevant features

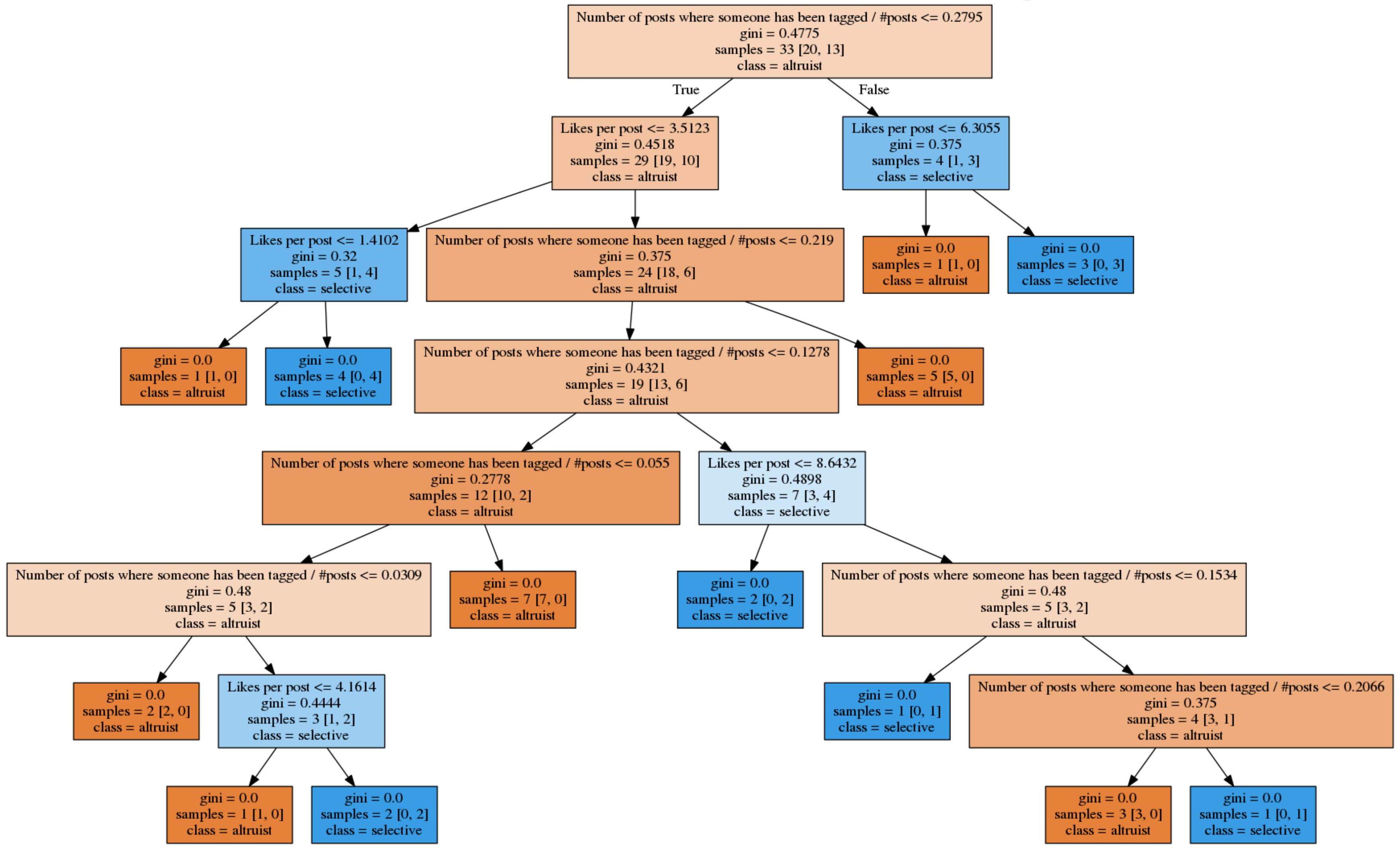

Figure 3.6. The decision tree trained with the data of Altruists and Selectives containing only two relevant features

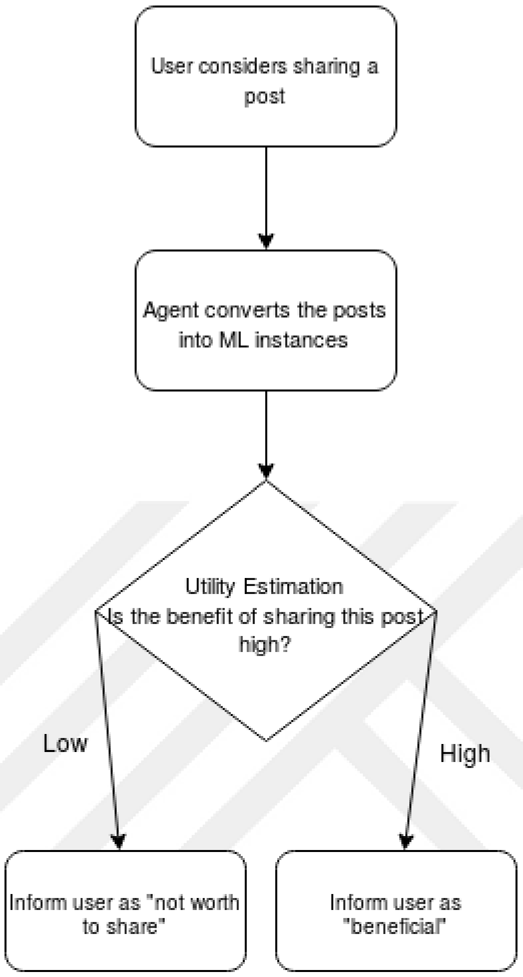
User considers sharing a post
Agent converts the posts into ML instances
Utility Estimation
Is the benefit of sharing this post high?
Low
High
Inform user as "not worth to share"
Inform user as "beneficial"

Figure 3.7. The first approach of utility estimation

Based on the procedure, the user is categorized into one of these personas, which, we assume, reflects her concept of utility in OSNs. Thus, the utility function corresponding to her persona is recognized as her utility function anymore.

3.3. Utility Learning

We approach the learning of the relation between the post content and the utility as a machine learning problem. However, we may treat this problem in two ways. In the first one, we can try to make our system learn directly the relation of the post attributes with the utility of sharing, which is depicted in Figure 3.7. For each of the user's past posts, one instance is created. Its attributes are converted to the input features, which is a step similar to the conversion for the privacy decision problem in Section 2.2. This conversion step is explained in Sections 3.3.1 and 3.3.2. The

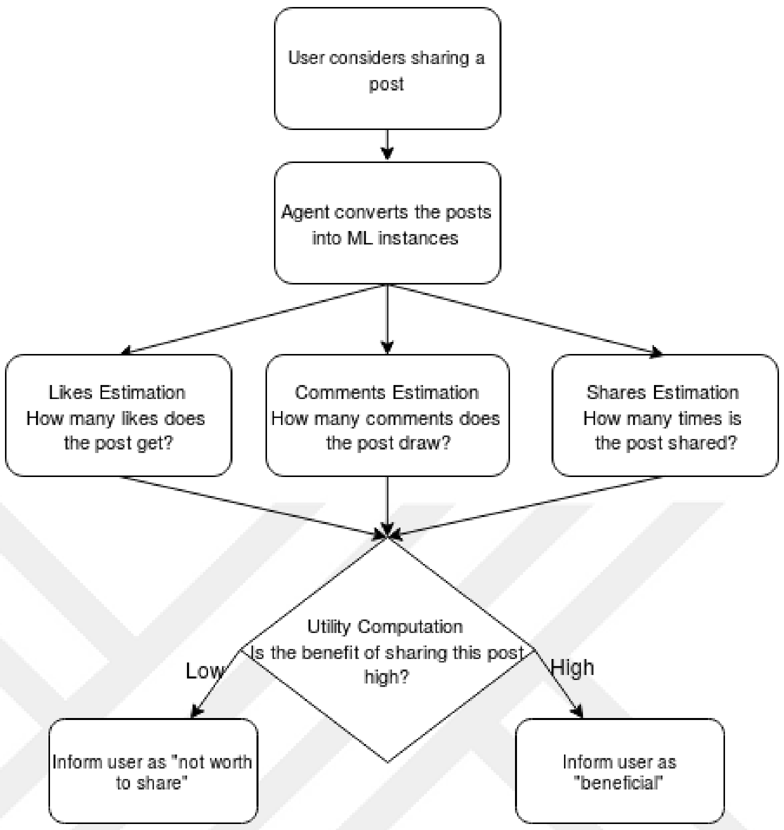
User considers sharing a post
Agent converts the posts into ML instances
Likes Estimation How many likes does the post get?
Comments Estimation How many comments does the post draw?
Shares Estimation How many times is the post shared?
Utility Computation Is the benefit of sharing this post high?
Low
High
Inform user as "not worth to share"
Inform user as "beneficial"

Figure 3.8. The second approach of utility estimation (our approach)

output of the instance is utility value of the post. For each past post, its utility value is computed by giving number of its likes, number of its comments and so on to the utility function of the user, which is assigned according to her persona. When she attempts to share a post, a machine learning instance is created by converting its attributes to the input features. Then, the regression model, a machine learning model which learns the relation of the input features with a continuous output variable, predicts the output value of this instance and hence her utility value of sharing this post. The next approach to this problem is to learn the relations between number of likes of her posts, number of their comments and number of their shares, and their attributes, which is depicted in Figure 3.8. One machine learning instance is created for each one of the past posts of the user. The input features of each instance are set as the result of conversion of the attributes of each post. Since there are multiple relations to consider, there are more than one machine learning problem in this approach. Additionally, we consider them as regression problems since the domains of output values, which are the number of likes, number of comments, and number of shares, are unbounded. Furthermore, the numerical relations between the values such as the numerical difference, the numerical order would be lost if those problems were treated as classification problems. For each of our regression models, an instance is reproduced with its corresponding value such as the number of its likes as the output value. Then each model is trained with its corresponding instances, which results in regression models which are able to predict how many likes a post will get, how many times it will be commented on, and how many times it will be shared. When the user shares a post, our system intervenes the process. It examines the post and it generates one machine learning instance by converting the attributes of the post to the features of the instance. Then the instance is reproduced for each regression model with its output value missing. The models estimate number of its potential likes, number of its potential comments, and how many times it may be shared. Then, the utility value of sharing this post is computed by inputting these estimations to the utility function of the user. In comparison with the first approach, this one requires more computational resource in the training phase due to the multiple number of regressors. However, when sharing a new post, its computation does not take much more time than the computation in the first approach since the prediction involves only few instances. Furthermore, it is more expressive than the first one since the number of likes, the number of comments, and the number of shares are predicted. By taking the opportunity of giving more detailed explanations to users and more computational costs only in the training time into consideration, we prefer the latter approach.

3.3.1. Instance Generation from Posts

3.3.1.1. Training Instance Generation. An instance from a post is generated for number of likes estimator similar to the generation for the privacy classifier in Chapter 2. However, considering the various photo-related and text-related features, we have decided to use separate estimators for posts and photos. For the likes estimator of posts, our agent chooses the following attributes: its sharing time, its sharing year, its location and its text content. The possible usages of sharing time and location are already covered in Chapter 2. Our agent calculates the number of minutes from midnight

(00:00) from the sharing time and it assigns this as a numerical feature, which we call *timeOfDay*. It also employs the year of the sharing time as a feature, which we call *sharingYear*. It is the year when the post has been shared. Since the social circle of a user in the real life and, thus in an OSN, usually grows in time, it brings more potential people to like, comment or share the content that the user has shared. Thus, a newer post of the user may have a more chance of being liked than her older posts. The agent assigns it as a numerical feature such as "2014". Besides the features based on the sharing time of the post, it extracts two categorical features from the location information: the city name and the country name, which is a fine-grained representation of location information. For the posts with no location information, predefined special values NO_CITY and NO_COUNTRY are assigned, respectively, to the city name and the country name features. For the posts with location information but not with a city name, predefined special value NO_CITY is assigned to the city name feature. For the posts with location information but not with a country name, predefined special value NO_COUNTRY is assigned to the country name feature.

The text content of a post can be employed to extract various features in order to represent the post. One of the features which could be extracted from the text is the one based on term frequency-inverse document frequency (tf-idf). Tf-idf is a statistical measure of terms, that is of words, in documents which is calculated for a term t in a document d with Equation 3.10, which is introduced by Jones [30]. To represent posts by employing tf-idf, first a list of all terms appearing in posts, dictionary, is produced. Then a document is represented with a list of tf-idf values of this document and corresponding term in the dictionary.

$$tf - idf(t,d) = \frac{count\ of\ t\ in\ d}{count\ of\ words\ in\ d} * \log \frac{count\ of\ all\ documents}{count\ of\ documents\ with\ t\ in\ it} \tag{3.10}$$

Another way to utilize the text content in the representation of a post is the text embeddings, a feature based on the word embeddings. A word embedding is the vectorial representation of a word in an n-dimensional space, in which the semantically similar words are close to each other. These word embeddings are learned from a training corpus. One of the word embedding technique which has gained a lot of interest due to its success is word2vec, which is introduced by Mikolov *et al.* [31]. This technique utilizes either one of two algorithms: Continuous Bag of Words (CBOW) and Skip-gram model. For CBOW, a shallow neural network which tries to predict the probability of a word given the context, that is, its neighboring words, is trained with the corpus. In this neural network, the input is the one-hot encoded ids of the words in the context while the output is the one-hot encoded id of the target word. At the end, the learned weights of the output layer in this neural network are the word vectors learned. In the latter implementation, Skip-gram, the neural network structure is reversed so that it tries to predict the neighboring words of a word given that word. Hence, its input is the id of a word while the output is the ids of context words. After the training, the learned weight matrix of the input layer is the word vectors learned. In our case, for all of the words in a post, their vector representations can be obtained via either one of these matrices of the trained model. After that, the post may be represented with the average of embeddings of its words so that all posts have the representations of equal size. Another popular word embedding is fastText, introduced by Bojanowski *et al.* in [32]. It is an extension of word2vec method with the main difference that the basic units are words in word2vec while fastText treats the words as they are composed of n-grams of characters, a sequence of n characters. That is, fastText learns the representations for n-gram of characters and constructs the representation of a word by exploiting the representations of its n-grams. For example, the vector of word *apparently* is constructed based on the vectors of n-grams *apparent*, *ly*, etc. Thus, while word2vec cannot deal with the words which is not in its training data, fastText can do so by constructing it from n-grams which exist in the training data. After the training a fastText model, the post may be represented with the average of embeddings of its words, which is done similar to representation of a post with the word2vec embeddings. In our system, we prefer fastText over tf-idf since tf-idf may lead more sparse representations considering the size of its word list, that is, its dictionary, and fastText is superior to it. When working on our system, we have dealt with generally Facebook posts containing Turkish text. Thus, we have preferred fastText rather than word2vec since fastText is better when dealing with morphologically rich languages such as Turkish due to its base of n-grams. Additionally, it can handle words that are not in training data while word2vec cannot. In order to get fastText representations of text content of posts, our agent employs a pre-trained fastText model which is trained on a Turkish wikipedia dump and is provided in a GitHub repository [33]. The size of a word vector in that pre-trained model was 200; hence, we have stored the average vector of words in the text content of a post as 200 numerical features. Our system ignores words of which vectors cannot be determined for a reason, for example, due to non-existing n-grams in the training data of the pre-trained model. If it encounters a post with no text or no words with vector representations, then it assigns a vector of size 200 filled with -1's.

The output is set as the number of likes of the post since this is the number of likes estimator while it is set as the number of comments and the number of shares of the post, respectively, for the number of comments estimator and the number of shares estimator. Then, the training data of estimators is ready.

3.3.1.2. Instance Generation of a New Post. When the user attempts to share a new post, our system intervenes the sharing process. The potential post is converted to a machine learning instance in a similar way to the conversion of the training posts in Section 3.3.1.1. The features of sharing time, the sharing year and the text embeddings are extracted the same way. For categorical features, if a new value, which has not been encountered in the training data (past posts), appears, our system assigns a predefined value representing the case of unknown to that feature. These predefined values are UNKNOWN_CITY, and UNKNOWN_COUNTRY, respectively, for the features of city name, and country name. One instance is generated same as in the process of past posts. Then these instances are sent to estimators to predict their output values. Thus, number of likes that the post will receive, number of comments that the post will draw, and how many times that it will be shared are estimated.

3.3.1.3. Utility Estimation of a New Post. The outputs of estimators are given to the utility function of the user to compute her utility of sharing this post. The utility value is normalized into the range of [0,1] by mapping the past post with the minimum utility value to 0 and the one with the maximum utility value to 1. If the normalized utility value is greater or equal to the utility threshold, 0.5 in our system, it is considered as of high utility. Otherwise, it is of low utility.

3.3.2. Instance Generation from Photos

3.3.2.1. Training Instance Generation. An instance from a photo is generated for number of likes estimator similar to the generation from the post for number of likes estimator in Subsection 3.3.1.1. For the likes estimator of photos, our agent chooses the following attributes: its sharing time, its sharing year, its location and its image content. The sharing time, sharing year and location are used same as in Subsection 3.3.1.1.

The image of a photo object in the OSN can be employed to extract various features in order to represent it. One way to employ the image is the image processing. The image can be processed in various ways in order to extract features. These methods can be activity recognition, cultural event recognition, concept recognition, scene recognition, object detection and so on. For example, they may interpret the image, respectively, as standing, Rio Carnival, graduation, indoor, and car. We prefer a web-based image processing API, a collection of free services offered by the company clarifai [34]. We employ their image tagging and image embedding services. For the first one, we prefer the general model of this service, which is not a domain-specific model. The service is like a combination of object detection and context recognition. For example, if you feed a picture in which a student poses with her diploma in her graduation ceremony to this service, it responses with cap, diploma (name of the objects) and graduation, ceremony (name of the context). Thus, when our agent sends the image to this service, the service returns a list of 20 words, named as tags in this service, which represent the context and the objects in the image. Since we have not defined any semantic distance between these words, we consider this feature as a categorical feature. For each tag assigned to its picture, our system reproduces the instance with that tag value set as picture tag feature. If a picture is not found, only one instance is generated with the predefined special value NO_PICTURE in its picture tag feature. Even though the picture is found, if the tag cannot be generated by the service, only one instance is generated with another predefined special value NO_TAG in its picture tag feature. For the latter service, the image embedding service, we prefer its general embedding model, which is based on the general model in the tagging service; thus, is not a domain-specific model. This image embeddings are similar to the word embeddings in Subsection 3.3.1.1 in the sense that vectors of visually similar images are close to each other in the vector space. Since the service provides vectors of size 1024, we have stored this information in 1024 numerical features. If the service cannot access an image or it cannot produce the embedding for that image, then -1's are assigned for these features.

Another way to use the image is to consider the number of the people in the picture which have been tagged in it, which we call the number of tagged people (NTP). Tagging is attaching the link of a users' profile generally to an area of picture in which her face is located; however, this area sometimes can be arbitrary. Tagging can be done by OSNs' users and OSNs can help the users to tag people by suggesting them the people. Even though an OSN does not support tagging, a human detection method can be employed to count automatically the number of people in the image. We have stored this feature as a numerical feature.

Another feature based on an image is being a profile picture which is based not directly on the image but its meta-data in the OSN. We call the feature *isProfilePicture* (IPP). It is a straight forward feature. It stores that the picture is a profile picture or not. Examining the data has showed that profile pictures tend to gain more attention such as like than the other pictures. Due to its binary structure, this is a boolean feature. That is, it is a categorical feature with the values "True" and "False".

The output is set the same way as the output setting in Subsection 3.3.1.1.

<u>3.3.2.2. Instance Generation of a New Photo.</u> When the user shares a new photo, our system intervenes the sharing process. The potential photo is converted to a machine learning instance in a similar way to the conversion of the training photos in Section 3.3.2.1. The sharing time, the sharing year, the image embeddings, the NTP, and the IPP are extracted the same way. For categorical features, if a new value, which has not been encountered in the training data (past photos), appears, our system assigns a predefined value representing the case of unknown to that feature. These predefined values are UNKNOWN_CITY, UNKNOWN_COUNTRY, and UNKNOWN_TAG, respectively, for the features of city name, country name and image tag name. The instance is also reproduced for each image tag in this conversion. If any tag cannot be generated by the service, only one instance is generated as in the process of past photo objects. Then these instances are sent to estimators to predict their output values. Thus, the number of likes that the photo will receive, and number of comments that the photo will draw are estimated. Since the OSN our current system works with, Facebook, does not provide the number of shares of photos by its API, we eliminated the number of shares estimator of photos.

<u>3.3.2.3. Utility Estimation of a New Photo.</u> Since the number of likes and the number of comments that the new photo will receive have been estimated, its utility estimation can be computed via the utility function of the user. The procedure is same as the utility estimation of a new post, which is explained in Section 3.3.1.3.

3.4. Conclusion

To sum up, when a user begins using our system, her OSN statistics, which are explained in Section 3.2, are computed. Based on these statistics, she is ideally assigned to one of the personas which are explained in Section 3.1. However, we end up with a method which assigns to only two personas, Altruists and Selectives, since the data which we collected to develop an assignment method is not balanced and there is no enough data of other personas to arrive at a decision for them. The method we propose is the decision tree depicted in Figure 3.6. The statistics of the user is given to this tree and the tree decides on the persona of the user. As a result, the user has one of the utility functions which are denoted in Equations (3.4) to (3.9) according to her persona.

In addition to this process, the user's past posts are converted to the machine learning instances as explained in Section 3.3.1.1 and Section 3.3.2.1. Then, comments, likes and shares estimators of the user are trained with these instances. When she attempts to share a post, it is also converted to an instance as explained in Section 3.3.1.2 and Section 3.3.2.2. The estimators estimate comments, likes, shares of this instance. Based on their outputs and the utility function of the user, the agent can compute its utility as explained in Section 3.3.1.3 and Section 3.3.2.3.

4. IMPLEMENTATION & EVALUATION

4.1. Privacy Experiments

While privacy research has been gaining importance, there does not exist any data set with which different approaches can be evaluated. In order to evaluate the performance of the above approach, OSN posts are needed. However, already existing OSN posts cannot be employed since OSNs, such as Facebook, currently do not provide all information regarding posts which our system needs. Additionally, since those posts may contain sensitive data and may be detrimental to privacy of their owners, users tend not to share their own data.

4.1.1. Data Generation

This calls for a method to generate synthetic data that can be used for evaluation of systems, such as this one. However, there does not exist standard techniques for generating these data sets. Here, we propose one such method. Our method consists of two steps:

- Post Generation: In order to talk about privacy, the first step is to have content over which privacy rules can be specified. This content can include text, pictures as well as location information. The generated posts are raw data in the sense that they are not labeled as being shown or not shown to certain individuals.
- Post Labeling: Each post is labeled as "to be shown" or "not to be shown" to a certain individual. This labeling needs to be done from an agent's perspective. Following Example 1, if posts are generated as check-ins to various NY sites and if they are being labeled from *Alice*'s perspective, then *Bob* and *Carol* should be labeled as "denied".

In this work, we realize this method as follows and generate a data set that can be used both in this work and in privacy works [35].

```
{
"postText": {
"text": "BAHIA COCOA REVIEW ... ",
"mentionedPeople": [],
"mentionedLocations": [
{
"com.pcontroller.domain.entity.ontology.location.Location": {
"placeName": "el-salvador"}}]},
"medium": null,
"sharingDate": 541342861079,
"audience": null
}
```

Figure 4.1. An example text post generated

4.1.1.1. Post Generation. Using the above approach, we have artificially generated OSN posts in the following two ways:

Reuters-21578 is a well-known text categorization research dataset [36]. It contains the news from Reuters, an international news agency. At first, this dataset can be seen as not adequate for OSN domain. However, we are accustomed to see users' sharing news in OSNs. For some people, they are the major reason to become OSN users. Thus, they are the fundamental part of the online content shared in OSNs. Furthermore, they may reflect the indirect information of users sharing them such as the political opinion, which may lead to privacy violations. Thus, we think that this dataset is an acceptable fit for our problem. For each news from this dataset, the release date of the news, the places it contains, its title, its content, and the topics, people, organizations, and companies related with it are kept in the dataset. We have used only the text, location information and date when generating text posts. In that way, 200 text posts have been generated, one of which is displayed in Figure 4.1.

Flickr is a famous image sharing platform. Using its API, we have searched the photos with the keyword "people". For each photo, its content, location information and the date of taken have been employed in order to generate an image post. Using that method, 200 image posts have been created.

This enables us to have content that we can work with. However, since privacy constraints are related to audience in particular, we also need to add audience for each post. For the posts of Alice, the agents Bob and Carol have been chosen as the audience.

4.1.1.2. Post Conversion. Even though posts exist, they are not ready to be employed as a dataset since the labels (the sharing decisions) are missing. In order to label posts as to whether they should be shared or not, we assumed there is an agent that has a given set of privacy rules. Those rules are defined in Semantic Web Rule Language (SWRL) [37] rules, which is explained in the next step. To evaluate posts based on those rules, posts are needed to be converted into ontology elements.

An ontology is a conceptualization of a domain. We use the ontology developed by Mester *et al.* to give meaning to various features [38].

Our ontology has the following classes:

- PostRequest: A post in OSNs.
- PostText: Text which is contained in a post.
- Location: Location information which a post mentions.
- Audience: Audience of a post
- Agent: A user in OSNs.

In the first step, an instance of PostRequest is generated in the ontology. In order to identify a PostRequest instance, the timestamp of beginning of the conversion process is employed as a key identifier. Thus, each individual of PostRequest is named as PostRequest_TimestampOfBeginning, which is its Internationalized Resource Identifier (IRI).

The next step depends on whether the post is a text post or a medium post. If it is a text post, a PostText instance is added into the ontology. Its IRI is determined as PostText_key, where key is the key of the PostRequest individual generated in the previous step. Following it, this PostText instance is linked to the PostRequest instance with the object property hasPostText. Since they are connected now, properties of PostText can be added into the ontology. The content of the text is linked to the PostText instance as the data property hasText. The locations included in the post are checked. If their corresponding individuals do not exist in the ontology, then an individual is added to the ontology for each location.

In the evaluation, a set of various agents in terms of privacy rules and the amount of decision noise are employed. These agents decide whether an access of an audience to a post should be accepted or not according to those privacy rules and then they flip their decision based on the probability determined by the amount of decision noise.

4.1.2. Single-Agent Results

In the first setting, we study a case where a single agent (e.g., Alice) uses her previous posts to predict the privacy status of a new post she is about to share.

In order to decrease the effect of structure of the training data on the evaluation results, 10 experiments were conducted for each machine learning algorithm. Each experiment was conducted with a new set of data for both training and testing. In each of them, the model is trained with 400 instances while it is tested again with 400 instances. Over 10 experiments, the means of numbers of *Yes* and *No* instances in the training dataset are, respectively, 131.2 ± 60.19 and 268.8 ± 60.19. For the instances in the test dataset, the means are 147.2 ± 70.53 for *Yes* and 252.8 ± 70.53 for *No*. Then, the accuracy of each ML method (such as SVM) is calculated as the average of accuracy of experiments of that method. Considering that our data are imbalanced, in order to evaluate the performance of ML methods in a fairer way, we can also employ other metrics such as precision, recall or kappa depending on our goal. However, since we compare the performance of classifiers with each other by training and testing them with the same data, accuracy is an adequate metric for this task. We have tried to generate data that resembles the real life. However, it is not clear whether the data of real OSN users will exhibit similar distribution of *Yes* and *No* classes. Thus, these other metrics can be more useful when dealing with real data.

The settings of the ML methods employed in the privacy experiments are as follows:

- Decision Trees (ID3): We did not use pruning: removing subtrees from the tree in order to prevent overfitting to data. Pruning is not employed in the default setting of ID3 in WEKA.
- ELM: 20 hidden neurons was assigned to ELM. The sigmoid function was used as the activation function.
- NB: We have employed normal (or Gaussian) distributions to process numeric features, which is the default setting in WEKA.
- RF: The number of trees, the maximum depth of the trees, and the number of features to consider when splitting, are set as, respectively, 100, unlimited and $\log_2 m + 1$, where m is the number of input features. These are the default parameters of the WEKA implementation of RFs.
- SVM: We use the C-Support Vector Classification (SVC) type of SVM and a linear kernel. The cost parameter C, a parameter to avoid misclassification, is set as 1.0. The tolerance of termination criterion is set as 0.001. The values of parameters other than the kernel type (linear kernel in our case) are default values of WEKA wrapper implementation of LibSVM.

Table 4.1 shows the accuracy of each classifier in terms of predicting the correct label of allow or deny. As it can be seen, SVM outperforms the other methods. In order to observe the limits of that approach, the size of training data (and also the test data) has been decreased. Again, 10 experiments with datasets given in the following table are conducted and accuracy is calculated in the same way.

Table 4.1. Accuracy of various ML classifiers without noise

Method Name	Accuracy
ELM	89.725
ID3	86.15
NB	86.95
RF	86.95
SVM	100.00

Table 4.2. The decrease in accuracy as the training data size decreases

Size of Training (& Test dataset)	Accuracy
400 (400)	100.00
200 (200)	100.00
100 (100)	100.00
50 (50)	100.00
25 (25)	86.00
13 (13)	80.77
7 (7)	64.28

Table 4.2 shows the variation on accuracy when the size of training and test dataset change. When data set has at least 50 training items, the change of size of training data set does not affect (average) accuracy. However, when there are fewer than 50 training instances, the accuracy starts to decrease. If we examine those with lower accuracy than %100 further, it is observed that some of the individual experiments with lower accuracy are caused by the training data in only one class. Of each 10 experiments with 25, 13, and 7 training instances, this behavior is observed in, respectively, 2, 4 and 6 experiments. In those case, the SVM models trained predicts the instances as the only class which they have learned. This affects the accuracy. Since those cases are also possible in practice, their results have taken place here. In order to ignore their effect, the accuracy of the rest of the individual experiments are calculated, which can be seen in Table 4.3.

Table 4.3. The decrease in accuracy as the training data size decreases

Size of Training (& Test dataset)	Accuracy
25 (25)	100.00
13 (13)	100.00
7 (7)	78.57

Only the one with 7 training instances has the accuracy less than 100 percentage.

It has been widely known that users may end up sharing information that they think is private, possibly because they do no think about the consequences up front [39]. It is also common to regret such posts later [40]. Since a user does not always behave in parallel with her privacy preferences in her mind, we have added noise to our dataset in order to reflect that gap. The noise is added at the step of determination of decision of whether access for an OSN post by another user should be allowed or denied. Without noise, an agent evaluates a post by taking privacy rules of its user in the SWRL format into account and accepts the result of reasoning of the post based on SWRL rules as the final decision. However, after adding noise, it decides in the opposite direction of the result of reasoning with a probability p_n. For example, consider adding the noise with the probability (p_n) 0.1 to Example 1. Then, Alice's agent tries to decide whether a post which has the location information USA should be shared with Bob or not. As a consequence of reasoning on Alice's rules, specifically on the rule R_{A_1}, it concludes that the access for that post should be denied. However, with the probability 0.1, it allows the access.

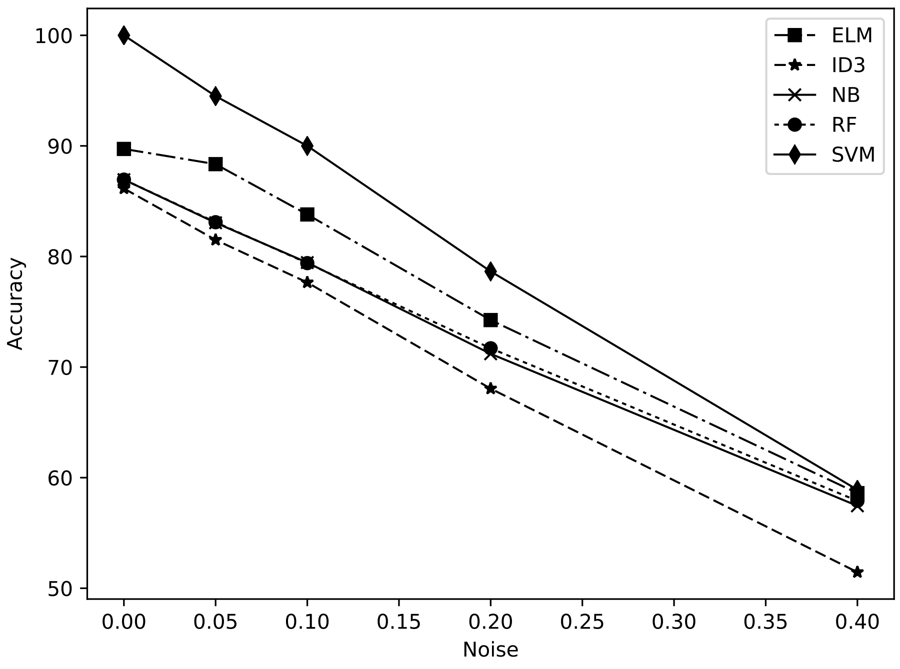

Figure 4.2. The effect of noise on the accuracy of each classifier

To observe the effect of noise on the accuracy of classifiers, we have carried out again 10 experiments with the dataset with 400 training and 400 test instances. Accuracies are again calculated as the average of 10 experiments for each pair of classifier and noise value. As noise values, 0.05, 0.1, 0.2 and 0.4 have been employed. Furthermore, the case of 0 noise (no noise) has been also added in order to develop a comparison of the experiments with the noise, with the previous experiments, which are the ones without noise. Figure 4.2 shows that the accuracy of each classifier decreases as the noise increases. This is expected. However, up to p_n=0.1, except those of SVM and slightly ELM, none of the accuracies of classifiers are acceptable. They are below 85%. After p_n=0.1, none of the classifiers perform well. At that point, there is a trade-off between noise value and accuracy. For the sake of high accuracy, the gap between behaviors and expectations of a user, thus, noise cannot be ignored and cannot be assumed as 0. However, after a point, it may not reveal the actual influence of the gap. When p_n=0.1, the accuracy of SVM does not decrease below the 90%. Thus, we employ a noise value of 0.1 in the following experiments.

4.1.3. Multi-Agent Results

As can be seen in Table 4.3, when the number of training data instances decreases, the accuracy of predicting the label also decreases. When an agent has been trained with a few instances, the agent can guess that its predictions will not be too accurate. In such cases, the agent can use (some of) the other agents to reach a decision.

In this evaluation, in addition to Alice's agent, we have created six more agents that make up the multiagent system. Bob has exactly the same privacy requirements as Alice. Carol's privacy requirement is more restrictive; in addition to denying posts from USA, she also denies posts from Canada. Dave's privacy requirement does not match Alice; he denies posts from Canada and Japan. The fourth agent is a random agent that picks randomly what the outcome will be. The last two agents are YES and NO agents; in the sense that the former always allows posts and the latter always denies them. Note that Alice is not aware of these behaviors.

For those multi-agent experiments, datasets for each agent were created in the same procedure with the single-agent experiments. The noise probability, which is used when determining the decision of sharing, has been chosen as 0.1 due to the reason mentioned in Section 4.1.2. SVM has been selected as the classification method since it had the highest accuracy in the single-agent experiments. Alice's agent has been trained with 14 instances while agents of Bob, Carol and Dave have been trained with 100 instances.

In the first experiment, Alice asks all the agents for the first of her test instances and gets the results as shown in Table 4.4. Since the answers vary and Alice has no other means to decide whom to trust, she applies majority voting, which is explained in Section 2.3.

Table 4.4. Answers for the first test instance of Alice

Agent Name	Decision
Bob	Allow
Carol	Deny
Dave	Allow
Random Agent	Deny
Yes Agent	Allow
No Agent	Deny

Table 4.5. Answers for each of the support posts of Alice

	Decision				
Agent Name	Id of Support Post				
	1	2	3	4	5
Bob	Allow	Allow	Allow	Allow	Allow
Carol	Deny	Deny	Deny	Deny	Deny
Dave	Allow	Allow	Allow	Allow	Allow
Random Agent	Allow	Deny	Allow	Allow	Deny
Yes Agent	Allow	Allow	Allow	Allow	Allow
No Agent	Deny	Deny	Deny	Deny	Deny

In the following experiments, she decides with the Algorithm 1 in Section 2.3. In the algorithm, her connections, Bob, Carol, Dave, Random Agent, Yes Agent and No Agent, are assigned to A (Line 1). Then she employs trust levels to decide. To calculate those levels, Alice's agent sends 1, 2, ..., 13 or 14 support posts to all agents. Their labels for the first five of support posts can be seen in Table 4.5, where each column displays the answers given for each support post by all agents while each row displays the answers given for all support posts by each agent. Thus, the table also reflects the behaviors of agents. For instance, the row of Yes agent contains only Allow values since it always allows. After getting those labels from agents, the trust levels are calculated with Equation (2.6). In that way, the original labels by Alice and the labels by other agents are compared. Thus, she has trust values for each agent. Then she gets the responses of other agents. Following it, Alice's agent calculates a weighted average response by dividing them.

In the experiments, we have used 14 test instances of Alice. For each test instance ins, $\textsc{CollectiveDecision}(Alice, 14, ins)$ has been executed. Since we have also 14 training instances, n is taken as 14. Then their results are compared with the actual labels of those test instances. For each case with different number of support posts, the average accuracy is calculated as the ratio of the number of correct predictions to the number of test instances, which is 14. Additionally, we have removed Random Agent from the set of agents and conducted experiments in order to examine its effect, thus the one of randomness. Since the existence of Random Agent in a real world environment is very possible, its effect is important to take into account. By employing those 14 test instances, we have also calculated average accuracy for majority-voting experiments, those without any trust levels.

Figure 4.3 displays the effect of number of support posts on the average accuracy of collective decisions. We experimented with two settings. The first one is denoted with the dashed line and circle markers. It reflects a static setting, such that the agents act consistently. Thus, the random agent is not taken into account. The second one is denoted with the solid line and diamond markers. It reflects a dynamic setting (e.g., the random agent is included). Since the random agent leads a randomness in the results, we have experimented with this setting 1000 times for each number of support posts. Then, we have averaged the accuracies of each experiment of each number of support posts. In the static setting, up to five support posts, the accuracy is too low to be useful since the correct trust levels cannot be understood with few instances. That is, the agent needs a number of interactions to identify the trusted agent. Then it increases abruptly and gets stable. In the dynamic case, there are few smooth fluctuations. The accuracies when one, two, and three support posts are employed is lower than the accuracy of the case with no support post since few instances again can be misleading to build trusts. After three posts, the accuracy increases and it goes on that level up to seven posts because the agent has some knowledge of other agents but then can go down because Alice's agent cannot model the random behavior of the random agent correctly. It would be an interesting future direction to build and use a trust model that can handle such fluctuations successfully.

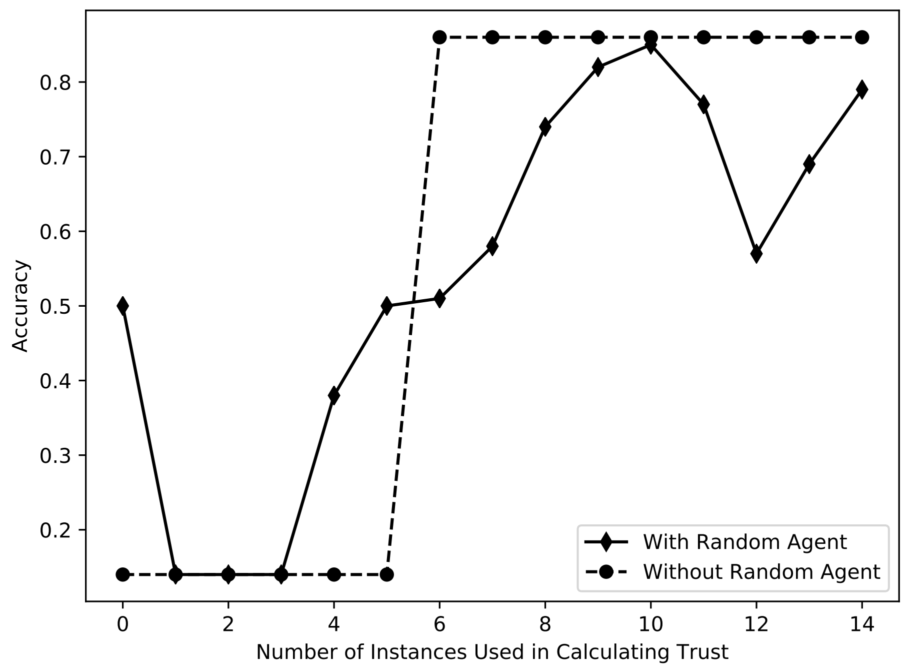

Figure 4.3. Accuracy of collective results with and without Random Agent

To sum up, our agent achieves great performance on single-agent experiments. With SVM classifier, its accuracy is 100% when it is trained with at least 50 instances. In the case of insufficient training data, the user's agent consults to her connections' agents. By employing support posts, the agent computes trust values of those agents, which represent similarities between the privacy expectations of their users and the user's privacy expectation. Based on these trust values, our agent decides on the privacy of the post which is attempted to share. The multi-agent experiments reveals that our system can perform well after a number of support posts, namely five in these experiments. Additionally, the existence of random agent may cause the system to perform worse; however, with the help of our trust model the system may perform well even though there is a random agent on its connections .

4.2. Utility Experiments

Even though the current OSNs do not provide adequate data for the privacy part of our application, their available data is sufficient for our utility experiments. In these experiments, the accuracies of estimations of our comment, like and share regressors are evaluated. In order to conduct those experiments, we have employed the data of users from Facebook with their permissions. Their posts and pictures, which have been shared by them or their connections on their Facebook profile, have been accessed by our application. It is a Facebook application; thus, it can connect to Facebook via its Graph API. Since these data are private, we have worked only with data of one Facebook user.

In our experiments, we first have randomized the order of instances. Then we have tested the estimators with 10-fold cross-validation since it reduces the effect of the way in which the subsets of the data are selected and it is better suited for small datasets compared to the splitting the dataset into training and test sets. The error measure considered in the experiments is root mean squared error (RMSE), which is calculated with Equation (4.1), where n, y_i, and $\hat{y}_i$ represents, respectively, the number of test instances, the actual output value of the ith instance such as its number of likes, and the estimated output value by the estimator for that instance. We prefer RMSE since it penalizes great errors more. If comments, likes and shares are estimated with great errors, RMSE is great, which leads great errors also in the utility computation. As a result, RMSE penalizes great errors in the utility computation. Since these errors can affect sharing decisions of the system, we want to observe them by employing RMSE.

$$RMSE = \sqrt{\frac{1}{n}\sum_{1}^{n}(y_i - \hat{y}_i)^2} \tag{4.1}$$

With the aim of improving performance of the regressors, we have applied the normalization and the standardization on the input data. In regression, the scale of each input feature may be different from each others' scale, which may result in that some features repress others. In order to avoid this situation, the input features can be scaled to a common scale.

Normalization is one of this techniques which scales the values of these features to the range [0, 1] with the following function:

$$normalizedValue(v) = \frac{v - minValue}{maxValue - minValue} \tag{4.2}$$

where minValue and maxValue are, respectively, minimum and maximum values of that feature in that dataset. Thus, minValue and maxValue are transformed to, respectively, 0 and 1. In order to measure the effect of the normalization on the performance, we have experimented with and without it, which are, respectively, represented as "NO+", otherwise "NO-".

The standardization is another technique in order to avoid this repress. It transforms the values into a range in which the data has a predefined mean and a predefined standard deviation. In our implementation, we have used the mean as 0.0 and the standard deviation as 1.0, which is called as standard normal distribution. In order to measure the effect of the standardization on the performance, we have experimented with and without it, which are, respectively, represented as "ST+", otherwise "ST-".

The settings of the ML methods employed in the utility experiments are as follows:

- Decision Trees (M5P): The minimum number of instances in a leaf is set as 4, which is the default value in the WEKA implementation.
- SVM: We use the epsilon-SVR type of SVM and a radial basis function (RBF) kernel. The cost parameter C, a parameter to avoid misclassification, is set as 1.0. The tolerance of termination criterion is set as 0.001. The epsilon in the loss

function is assigned as 0.1. The gamma parameter of RBF is set as $\frac{1}{k}$ where k is the number of input features. The SVM type and the kernel type that we choose are not the default values of these parameters in the WEKA implementation while others are from the default setting.

4.2.1. Post Experiments

In these experiments, we measured the estimation performances of our regressors for three types of outputs: the number of comments, the number of likes, and the number of shares. For each output type, we have compared the performance of the technique of M5P and SVM, which are, respectively, explained in Section 2.1.1 and in Section 2.1.2. Additionally, we have aimed to measure the effect of the existence of the following features on the performance: sharing year ("SY+" and "SY-", respectively, for with and without it) and text embeddings ("TE+" and "TE-", respectively, for with and without it).

In these experiments, 300 post instances have been employed with 10-fold cross-validation.

Table 4.6. RMSE Values of Comments of Post Experiments When Mean of the Comments is 1.84

		M5P				SVM			
		SY-		SY+		SY-		SY+	
		TE-	TE+	TE-	TE+	TE-	TE+	TE-	TE+
ST-	NO-	3.85*	4.04	3.86	4.15	4.08	4.05	4.06	4.04
	NO+	3.85*	4.09	3.86	4.20	4.15	4.10	4.15	4.10
ST+	NO-	3.85*	4.25	3.86	4.31	4.15	4.01	4.15	4.00
	NO+	3.86	4.04	3.86	3.89	4.15	4.09	4.15	4.06

The RMSE values of the experiments of comment, like, and share estimators are, respectively, displayed in Tables 4.6 to 4.8. In general, zero RMSE is the best case. However, it is hard to achieve and our experiments can not achieve it either. Since

Table 4.7. RMSE Values of Likes of Post Experiments When Mean of the Likes is 12.57

		M5P				SVM			
		SY-		SY+		SY-		SY+	
		TE-	TE+	TE-	TE+	TE-	TE+	TE-	TE+
ST-	NO-	23.56	24.70	23.14	24.02	24.65	24.55	24.64	24.63
	NO+	23.75	24.70	22.98*	24.15	24.64	24.62	24.58	24.62
ST+	NO-	23.52	24.28	23.20	23.96	24.61	24.62	24.45	24.55
	NO+	23.61	24.33	23.01	23.99	24.64	24.62	24.59	24.62

Table 4.8. RMSE Values of Shares of Post Experiments When Mean of the Shares is 0.22

		M5P				SVM			
		SY-		SY+		SY-		SY+	
		TE-	TE+	TE-	TE+	TE-	TE+	TE-	TE+
ST-	NO-	0.75*	0.75*	0.79	0.76	0.77	0.78	0.77	0.77
	NO+	0.75*	0.75*	0.75*	0.76	0.75*	0.75*	0.75*	0.75*
ST+	NO-	0.75*	0.76	0.76	0.76	0.75*	0.75*	0.75*	0.75*
	NO+	0.75*	0.76	0.76	0.75*	0.75*	0.75*	0.75*	0.75*

our results are far from zero, we can say that our estimators do not perform very well in any of the experiments. To interpret how poorly they perform, we think that the means of the output of training data of estimators can help. Thus, we provide them in the captions of tables. Therefore, one experiment can be interpreted alone by comparing it with the mean value. Since all values in Tables 4.6 to 4.8 exceed mean values in their captions, the argument that our estimators perform poorly is supported. Additionally, the best values are marked with asterisk in the tables in order to increase the understanding of the performance of estimators. In Table 4.6, even the best value is over twice the mean, which indicates that our comment estimator for posts performs very poorly. In Table 4.7, the best value is close to twice the mean, which demonstrates that our like estimator has a poor performance too. The sharing estimator is the worst among them since the best value in Table 4.8 is over three times the mean. Since our results are bad in an absolute sense, we can compare the effect of existence of features and the usage of scaling techniques such as the normalization on the results by following their corresponding columns and rows. Standardization and normalization do not lead any noticeable difference in the error as we expect. The feature of sharing year (SY) has a slight improvement in the performance of estimators, which may confirm our intuition that SY affects the interaction of users with a post since the social network around the user grows with time. The text embeddings feature has an interesting trend. While it occasionally causes SVM estimators perform slightly better, it worsens the performance of M5P estimators. In general, RMSE values of all the comment experiments are above the mean value; thus, we can say that they do not perform well. We think that the reasons of this bad performance are following: Text embeddings feature is not enough to represent the text content of a post well; thus, it is not enough to reflect the reasons of interaction with the post. Additionally, the motivations of people to like a post, comment on it or share it again are so complex to learn. Even humans are not always able to understand these motivations. Since we aim personal classifiers for each of users in OSN, our system exploits only own data of each user in order to train these personal classifiers. Thus, considering the number of training instances generated from posts of a user, our training data is also too small to learn that kind of complex relation.

4.2.2. Photo Experiments

In these experiments, we measured the estimation performances of our regressors for two types of outputs: the number of comments, and the number of likes since we were not able to obtain the share counts of photos via Facebook API. For each output type, we have again compared the performances of M5P and SVM techniques, which are, respectively, explained in Section 2.1.1 and in Section 2.1.2. Additionally, we have aimed to measure the effect of the existence of the following features on the performance: sharing year, image embeddings, image tags, being a profile picture, and the number of tagged people. Their existence are represented, respectively, with ”SY+”, ”IE+”, ”IT+”, ”IPP+”, and ”NTP+” while the experiments without them are marked, respectively, with ”SY-”, ”IE-”, ”IT-”, ”IPP-”, and ”NTP-”.

In these experiments except those with image tags (IT+), 208 photo instances have been employed with 10-fold cross-validation. IT+ experiments are supposed to have 4160 (208 photo * 20 tags); however, they have 4141 instances since one photo cannot be tagged due to the access problems and NO_TAG is assigned to its tag feature. Since instances with NO_TAG may occur in real-life scenarios, we have not removed that instance. Due to this difference between IT- and IT+ experiments, the mean of the training data also changes. To interpret the results, we have provided the mean values in the caption of tables in Section 4.2.1. However, we provide two means for each table due to this change. The first one should be considered for IT- experiments while the second one is the mean of training data of IT+ experiments. Additionally, the best three results are marked with asterisk in order to increase readability.

There are lots of comment experiments for photos. Thus, the experiments of comment estimator are divided into two groups: those with M5P and SVM methods, which are, respectively, displayed in Tables 4.9 and 4.10. The best results of comments estimator are over the mean; however, they are not so close to twice the mean. Considering that the best performance of the comment estimator is over its corresponding mean in the post experiments as explained in Section 4.2.1, we can say that our system performs better for comments of photos than for comments of posts. The best three values of SVM experiments for photos, which are marked with asterisk in Table 4.10, are lower than those of M5P for photos, which indicates that SVM method performs better. However, in both of the experiment groups, the best values are still over the mean, which points that our system does not perform well. Another interesting point about the best values is that they are achieved when features of sharing year (SY) and being a profile picture (IPP) are included. This indicates that these features are helpful to find the relation between the photos and the number of their comments. In addition to best values, we can also interpret the effect of features and scaling techniques on the RMSE values. If we compare ST+ rows with ST- rows, and NO+ rows with NO- rows,

Table 4.9. RMSE Values of Comments of Photo Experiments with M5P When Means of the Comments of IT-, IT+ Experiments are, Respectively, 1.83, 1.84

				SY-				SY+			
				IE-		IE+		IE-		IE+	
				IT-	IT+	IT-	IT+	IT-	IT+	IT-	IT+
IPP-	NTP-	ST-	NO-	3.57	4.21	9.85	4.20	3.38	7.52	9.87	4.20
			NO+	3.59	4.21	4.27	4.20	3.40	7.52	7.18	4.20
		ST+	NO-	3.58	4.21	5.36	4.20	3.42	7.52	3.63	4.20
			NO+	3.57	4.22	4.44	4.20	3.43	7.52	4.35	4.20
	NTP+	ST-	NO-	3.63	4.94	7.78	4.07	3.39	4.81	3.61	4.05
			NO+	3.61	4.94	7.36	4.07	3.42	4.81	9.25	4.05
		ST+	NO-	3.62	4.94	8.13	4.07	3.41	4.81	15.11	4.05
			NO+	3.60	4.94	6.44	4.07	3.43	4.81	8.11	4.05
IPP+	NTP-	ST-	NO-	3.52	4.19	3.96	4.08	3.14*	4.68	6.18	4.08
			NO+	3.42	4.19	8.16	4.08	3.20*	4.68	10.21	4.08
		ST+	NO-	3.30	4.19	8.63	4.08	3.22	4.68	7.25	4.08
			NO+	3.37	4.19	4.55	4.08	3.19*	4.68	4.12	4.08
	NTP+	ST-	NO-	3.31	4.30	5.52	4.13	3.23	7.14	4.32	4.13
			NO+	3.32	4.30	18.01	4.13	3.22	7.14	10.45	4.13
		ST+	NO-	3.28	4.30	5.71	4.13	3.25	7.14	6.00	4.13
			NO+	3.48	4.30	7.51	4.13	3.27	7.14	3.93	4.13

Table 4.10. RMSE Values of Comments of Photo Experiments with SVM When Means of the Comments of IT-, IT+ Experiments are, Respectively, 1.83, 1.84

			SY-				SY+			
			IE-		IE+		IE-		IE+	
			IT-	IT+	IT-	IT+	IT-	IT+	IT-	IT+
IPP-	NTP-	ST- NO-	3.82	3.79	3.96	3.62	3.84	3.75	3.95	3.62
		ST- NO+	4.10	3.11	4.13	3.08	4.04	3.07	4.13	3.07
		ST+ NO-	4.11	3.11	3.97	3.09	3.78	2.99*	3.95	3.08
		ST+ NO+	4.09	3.11	4.13	3.08	4.04	3.07	4.13	3.07
	NTP+	ST- NO-	3.83	3.75	3.95	3.64	3.84	3.73	4.00	3.63
		ST- NO+	4.10	3.11	4.13	3.08	4.05	3.07	4.13	3.07
		ST+ NO-	4.08	3.11	3.96	3.09	3.81	2.99*	3.95	3.08
		ST+ NO+	4.10	3.11	4.13	3.08	4.04	3.07	4.13	3.07
IPP+	NTP-	ST- NO-	3.78	3.79	3.96	3.62	3.86	3.75	3.94	3.61
		ST- NO+	3.98	3.00	4.13	3.05	3.90	2.97*	4.13	3.04
		ST+ NO-	3.95	3.01	3.95	3.08	3.70	2.90*	3.95	3.07
		ST+ NO+	3.97	3.00	4.13	3.05	3.89	2.97*	4.13	3.04
	NTP+	ST- NO-	3.83	3.75	3.93	3.64	3.82	3.72	3.94	3.63
		ST- NO+	3.97	3.00	4.13	3.05	3.91	2.97*	4.13	3.04
		ST+ NO-	3.95	3.00	3.96	3.08	3.72	2.90*	3.94	3.07
		ST+ NO+	3.96	3.00	4.13	3.05	3.91	2.97*	4.13	3.04

it can be seen that standardization and normalization do not lead any significant difference in the error. Furthermore, NTP+ and NTP- rows denote that NTP decreases or does not change the error except the few experiments; thus, it complies with our intuition that NPP of a post affects the interaction of users with it. Additionally, it is interesting that either of image tags and image embeddings decreases the performance in the experiments with M5P while they increase the performance in those with SVM. It can be seen by comparing the differences between RMSE values of IE+ (or IT+) rows and IE- (or IT-) rows in Table 4.9 with the differences RMSE values of IE+ (or

IT+) rows and IE- (or IT-) rows in Table 4.10. This may point that the structure of SVM can exploit the information in these features while M5P cannot.

Table 4.11. RMSE Values of Likes of Photo Experiments with M5P When Means of the Likes of IT-, IT+ Experiments are, Respectively, 12.57, 12.47

				SY-				SY+			
				IE-		IE+		IE-		IE+	
				IT-	IT+	IT-	IT+	IT-	IT+	IT-	IT+
IPP-	NTP-	ST-	NO-	26.33	70.08	28.59	35.21	27.23	27.87	26.07	22.01
			NO+	26.10	70.15	38.82	35.21	24.21	27.87	37.82	22.01
		ST+	NO-	26.52	70.08	36.33	35.21	25.22	27.87	52.27	22.01
			NO+	26.25	72.45	35.66	35.21	25.25	27.87	95.95	22.01
	NTP+	ST-	NO-	23.23	34.30	36.85	28.77	21.98	30.49	111.29	22.71
			NO+	23.59	34.26	38.60	28.77	21.89	30.49	31.93	22.71
		ST+	NO-	22.22	34.30	29.02	28.77	22.35	30.10	38.18	22.68
			NO+	22.94	34.30	33.89	28.77	20.84	30.49	29.88	22.68
IPP+	NTP-	ST-	NO-	24.27	25.17	26.18	34.91	20.36	19.27*	200.28	21.92
			NO+	25.11	25.41	30.04	34.91	22.61	19.27*	46.67	21.92
		ST+	NO-	25.00	25.17	28.44	34.91	21.89	19.27*	80.22	21.92
			NO+	24.09	25.17	35.01	34.91	20.49	19.27*	113.47	21.92
	NTP+	ST-	NO-	23.16	39.63	31.06	28.73	19.11*	30.81	68.88	22.55
			NO+	21.89	39.59	31.07	28.73	19.43	30.81	63.25	22.55
		ST+	NO-	20.65	39.63	28.53	28.73	21.19	30.41	32.72	22.52
			NO+	21.27	39.63	48.20	28.73	19.26*	30.81	25.39	22.52

The results of the experiments of like estimator are also divided into two groups: those with M5P and SVM methods, which are, respectively, displayed in Tables 4.11 and 4.12. If we examine their lowest error values, SVM performs again better than M5P model. The lowest error values are achieved again in the experiments with sharing year feature included, which supports the functionality of this feature. The best value in Table 4.12, 16.90, is achieved when all features and standardization is used; thus, it can be said that our features together lead better results than separately. Besides the best values, the effect of features and scaling techniques can be examined by considering their corresponding columns or rows. Focusing on ST-, ST+, NO- and NO+ rows, it can

Table 4.12. RMSE Values of Likes of Photo Experiments with SVM When Means of the Likes of IT-, IT+ Experiments are, Respectively, 12.57, 12.47

				SY-				SY+			
				IE-		IE+		IE-		IE+	
				IT-	IT+	IT-	IT+	IT-	IT+	IT-	IT+
IPP-	NTP-	ST-	NO-	27.91	19.26	27.56	18.62	27.89	19.15	27.69	18.59
			NO+	27.96	18.06	28.20	18.20	27.95	18.07	28.33	18.21
		ST+	NO-	28.00	18.11	28.11	16.98	27.91	18.28	28.11	16.94
			NO+	27.94	18.06	28.24	18.20	27.97	18.07	28.23	18.21
	NTP+	ST-	NO-	27.90	19.26	27.59	18.75	27.90	19.16	27.67	18.76
			NO+	28.00	18.05	28.24	18.20	27.95	18.06	28.30	18.21
		ST+	NO-	27.89	18.00	28.12	16.95	27.93	18.23	28.11	16.92*
			NO+	27.98	18.05	28.27	18.20	27.97	18.06	28.22	18.21
IPP+	NTP-	ST-	NO-	27.89	19.26	27.65	18.62	27.87	19.14	27.57	18.59
			NO+	27.88	18.00	28.18	18.14	27.88	18.02	28.23	18.14
		ST+	NO-	27.85	18.02	28.10	16.96	27.77	18.19	28.10	16.93*
			NO+	27.90	18.00	28.21	18.14	27.89	18.02	28.24	18.14
	NTP+	ST-	NO-	27.91	19.25	27.60	18.75	27.85	19.15	27.62	18.75
			NO+	27.87	18.00	28.18	18.14	27.88	18.01	28.24	18.14
		ST+	NO-	27.82	17.94	28.10	16.94	27.76	18.09	28.10	16.90*
			NO+	27.87	18.00	28.25	18.14	27.90	18.01	28.23	18.14

be seen that the difference in the error due to standardization and normalization is not considerable. Especially in the experiments with image tags, which can be seen in IT+ columns in Table 4.11, it can be seen that neither standardization nor normalization have any effect on the error values. The columns IT+ and IT- of Table 4.12 also show that image tags definitely improve the performance of SVM estimator. While the feature of sharing year (SY) has a slight improvement in the post experiments, it does not exhibit any consistent behavior in the photo experiments.

In general, RMSE values of all the photo experiments (comment and like) are above the mean value; hence, we can say that they do not perform well. We think that the reasons of that bad performance in the post experiments similarly apply also here: the motivations of people to interact with a photo are so complex to learn. Again, even humans are not always able to understand these motivations. Since we aim personalized classifiers for each of users in OSN, our system exploits only own data of each user in order to train these personal classifiers. Thus, considering the number of training instances generated from photos of a user, our training data is also too small to learn that kind of complex relation. To sum up, the relation between posts and photos, and utility of sharing them is too complex to learn with that small amount of data. Other features than ours may also reveal the relation better.

5. DISCUSSION

We propose a method to help users preserve their privacy in OSNs. In this method, an agent represents a user in OSN. It examines the user's own posts that have been shared before and extracts content-based features and privacy preferences for each post from that data. Then, it is trained on that data so that it learns the privacy preferences of the user with the help of machine learning methods. When the data about user's own posts are not enough to learn the privacy preferences accurately, the agent can consult a subset of other agents in the OSN in order to decide about the privacy of a post. After collecting their privacy decisions, the agent aggregates them based on their users' similarities to its own user and determines the final decision. Additionally, content-based features and utilitarian properties such as likes, comments of each post are extracted from that data. The agent is also trained with it in order to learn the relation between post's properties and its utility from the users's perspective.

Fang *et al.* have developed one of the first approaches to privacy preference learning, where a privacy wizard helps a user create her own privacy preferences by reducing the user burden [41]. The wizard collects the profile data visible to the user. Then it selects a feature space. Each of the user's friends can be described using a feature vector within that space. Then, privacy preferences about the friends are asked to the user in decreasing order of their uncertainty score. According to those answers, a privacy preference model of the user is constructed, which is essentially a classifier. The features collected from the neighborhood of the user consist of two type features: community-based features and other profile information. The features based on the profile information are gender, age, education history, work history, relationship status, political views, and religious views. Since their values have limited options, they can be easily converted to features. Additionally, the online activities of users such as tagged photos, groups, fan pages and so on can be used as features. The first group of features is based on the community structure, which is determined by using an algorithm based on edge betweenness. With the aim of maximizing the modularity score, the network is partitioned into communities. When partitioning into communities, hierarchical structure is also aimed. Then for each user, a feature vector where each entry represents being member in the corresponding community or not is assigned as community-based features of that user. After feature extraction, each pair of profile item and friend is asked to the user whether user allows to that friend to see that profile item or not. Since the user may give up at any time during that answering process, the order of asking pairs is important. In that study, pairs are asked in a decreasing order of uncertainty, where uncertainties are calculated by training a classifier and using this classifier to predict distributions of labels of friends. This calculation is made based on the entropy of the predicted class distribution. This step is called uncertainty sampling. After the sampling phase, a classifier is constructed by using the data of labeled friends. The advantage of that instantiation of the privacy wizard is that it supports incremental maintenance. That is, when a user adds new friends, user will have the option to label new friends or to continue without labeling any of them. For advanced users, a tool is presented to visualize the model and the inference behind it, and to enable the user to modify the model. This method does not take advantage of privacy preferences of other users in OSNs while our approach benefits from interacting with other users.

Squicciarini *et al.* propose a privacy policy inference mechanism of images which are shared by users on online social networks [42]. The mechanism classifies images of each user based on their content (texture, symmetry, shape and SIFT descriptor). Following that, images are subclassified according to their meta-data (text-based tags, comments, and captions). In order to find a policy for an image, policies of images in its class are first normalized. Based on interestingness measures, most associated subjects, actions and conditions, and the policies in which they appear are determined. The candidate policies containing those most associated subjects, actions and conditions are created. The one with the most strictness level is chosen as the final candidate policy for that image. Their approach generally assumes that a centralized system can be used to find related policies. In our case, each agent has access to its own data and can request a content to be classified from a selected set of other agents. This enables agents to track who has access to their policies.

Bilogrevic *et al.* develop an information sharing system regarding the location privacy of the user, people around her and her availability [43]. The system consists of a directory to which information requesters and information owners are subscribed. When one of those information is requested, it is shared or not according to user's privacy classifier's output. Additionally, a multi-class classifier is used in order to output the level of information shared. While they just take the privacy of location information into account, we consider the whole structure of a post as our privacy problem. Furthermore, we aim to help users, even when they have few data items by employing trust in a multi-agent setting. Sadeh *et al.* extend a previous study, in which they introduced PeopleFinder, an application helping people to selectively share their location with their friends, family and colleagues [44]. They aim to understand people's attitudes towards privacy with the help of that application and to help them to articulate their privacy preferences ("policies"). Laptop users and mobile phone users can use PeopleFinder. It works as follows: users, when connected to the network, periodically report their location to their PeopleFinder agent in the server. When a user requests the location of another one, agent of the user whose location is requested invokes her Policy Enforcing Agent (PEA) to check whether the request is consistent with the privacy rules specified in her privacy policy, which has been expressed beforehand by the user. (The rules are expressed by using a rule extension of the Web Ontology Language (OWL).) If yes, then the location information is processed if needed and forwarded to the user requesting, and the other user is informed about that. If no, an ambiguous message that does not allow the requester to infer whether it is the decision of the other user or not is returned to the user. When expressing the privacy policy rules, days of the week, times of day, the requestor (particular individuals or groups of users) or the location at the time of the request can be employed as the restrictions of those rules. Auditing functionality has been added to that version of PeopleFinder. It enables users to review previous requests (feedback of the user about her happiness of the system's (non-)disclosure decision) and to see the explanation behind the processing of the rules (why the request is accepted or denied). There are two type of experiments in that study. The first one is the lab experiments. The users are asked to provide information about their daily routines and social networks, and to specify their privacy rules. Additionally, individualized scenarios are asked to them. They are asked to whether they felt comfortable disclosing their location for each of those scenarios. The result of the processing of their current policies are displayed to them. The functionality to refine their policies is also provided. Based on their comfort with individualized scenarios, the accuracies of policies have been evaluated. The accuracy of the cases with initial rules, with the progressively modified rules, with the modified rules applied to all scenarios and with the usage of case-based reasoning with a k-nearest neighbor heuristic increases, respectively. The second part of experiments is field studies. This time, the application is actually used by the people while the previous part involves answering questionnaires and a simulated environment. In that setting, the accuracy of Machine Learning method (a random forest classifier) is higher than the one of the progressively modified rules. The accuracy of the progressively modified rules is higher than the one of the modified rules in the first settings of experiments. The difference between the first and the two settings can be explained with following reasons: users' being more careful in stating their rules in a real setting, the improvements in the system, more data in the real setting, the tendency to examine situations which is not wanted to share location in the lab setting and the insufficient context of participants in the lab setting. While our work focuses on the privacy of OSN posts, this study only focuses on the privacy of location information, which makes the problem less complicated. Since OSN posts may contain the location information, the features used in this study are mainly a subset of the features we use in our work. Thus, the privacy of OSN posts includes the privacy of location information, which makes the problem more complex. Additionally, they provides a user interface to label locations on a map of the university, which would impose a user burden in OSNs since the location of users varies a lot. They also limit the location to the university area, which simplifies the problem too, while we do not limit the location in any way.

Ravichandran *et al.* propose an approach to capture privacy preferences while decreasing the user effort and not decreasing expressiveness of the privacy settings so much [25]. Applying traditional machine learning techniques does not lead intuitive and usable policies since the raw data such as longitude and latitude do not mean a lot for each user. Also, users have more event-based schedules than time-based schedules.

Thus, canonical policies are presented and employed in that study. The features used as input to the learning method are day of the week weekday, weekend, time of the day (six parts of a day), duration and location off-campus, school, on-campus residence, restaurant, mall, and, unclassified. The duration is excluded from the features since it increases both the error and the complexity of policies. Number of the parts within a day is determined after trial-and-error (trying 1, 2, 3, 4, 6, 8, 12-hour parts). The best accuracy is achieved when 1-hour time intervals are used. However, it is best when learning individualized policies while it may cause unintuitive policies. Based on the features mentioned, C4.5 decision-tree algorithm has been used to determine individual policies (from the audit data of each user separately). With the aim of lower test error, various combinations of features are tried. At the end of that step, we have a set of individual policies, each of which has a set of rules. After weighting individual policies by the frequency of occurrence of the rules, K-means clustering algorithm is applied to those policies. After clustering, the rules within the same cluster are concatenated to a single list. Then, a decision tree is applied on that list. Numbers from 1 to 4 are tried as the cluster number. Error rate decreases when the number of clusters is increased. However, more cluster means more complexity. Since there is no marked decrease in the error from three clusters to four clusters, three is chosen. Note that the duration of each rule of the policy is considered as a measure of "intuitiveness". Another important feature of the study is conservativeness ratio of a user, which is defined as the ratio of the user's unhappiness when the system wrongly allows access, to the user's unhappiness when the system wrongly denies access. In other words, the unhappiness for false positive / the one for false negative. That measure is used within the canonicalization of the time and the accuracy loss as a weight. As a conclusion, there is a relationship between conservativeness and accuracy, and between time granularity and accuracy. There is also a trade-off between the intuitiveness of policies and the number of such policies. While the focus of this method is privacy of location information, our approach focuses on one of OSN posts. Since the space of inputs depends on the value space of features, the input space in this work is finite and smaller compared to our approach, which may be insufficient to cover various privacy preferences in OSNs. Since they need the privacy policies of multiple users to cluster them, their approach is more centralized than ours. We work on the individualistic classifiers in our single-agent settings. Thus, it is less likely that a user can make inferences on others' privacy policies by employing her own privacy classifier in our study while she can do so by employing her persona information.

In order to learn the users' privacy preferences and help users with no or few data, Mugan *et al.* also employ machine learning techniques [26]. Their study is focused only on the privacy of sharing the location attribute. Thus, the input to machine learning algorithm is the attributes of those location shares, which are time and location information itself. After examining the data, they have observed some sharing patterns such as the one that most of users share their location only one time window in a day, of which size may vary across them. Furthermore, the meanings of values for those attributes may change for each user. For example, a location coordinate pair can be home for someone while it is not for most of other people. Thus, instead of usage of exact values of attributes, they have mapped them to canonical concepts. Based on share time window of a user, the time slot in a day she shares her location, time of each share is mapped to one of values {*in-window*, *not-in-window*}. Additionally, day in which location is shared is also mapped to one of {*weekday*, *weekend*}. Location information itself is converted to one of {*home*, *work*, *other*}. These mappings result in 12 (2*3*3) possible states for each share. At that point, the machine learning problem is a classification problem, which maps each of those states to a Boolean value (share or not share). That is, a 12-bit vector is aimed to learn. Applying decision trees on the data collected from the users, each user's 12-bit vector, *policy vector* in that study, is learned. Next, all of them are clustered with k-means algorithm. Decision trees are again applied on each cluster of policy vectors, resulting in *default persona*s, which represent the common characteristics of users in terms of privacy. Then, a user can pick a default persona for her account. Since it may not meet her privacy needs completely, it should be personalized. With that aim, incremental suggestions are introduced. They are found via neighborhood search, which iterates all of the policies similar to the current policy and chooses the one causing the best improvement over the test data among them. Since the default persona is generated using a decision tree, it can be represented by a set of rules. At each iteration, the neighborhood search considers a single change at a single rule. At the end, a policy for each user has been generated, which is based on a default persona and has incremental suggestions on top of it. Compared to our approach, again they only focus on the location attribute while we focus on sharing posts which can have location information. Furthermore, they decrease the input space to a small value (12), which is the number of possible states of each share in their work, while input space of our approach is not fixed and it varies from user to user. In our system, we aim to learn separate classifiers for each of the users. In these classifiers, we have categorical features, of which possible values change from user to user. For example, the dimension of location features is the number of distinct values in the training data of this user (and the predefined values such as the one representing no location information). Since the past posts of users varies apparently, the dimensions of the features in their data change, which results in various input spaces for users. While input space of this work may be too simple to cover various possible cases in OSNs, our approach does not limit the possibilities of the input features.

Another method to protect privacy with the help of machine learning techniques is proposed by Tøndel *et al.* [45]. Rather than privacy in OSNs, they focus on the privacy of a user when browsing any web site, which may attain information about the user such as her location, the device used, and so on. In their study, they consider the web sites with a machine-readable privacy policy, which is initiated by the Platform for Privacy Preferences (P3P). They argue that the reason behind the unpopularity of this approach is that users' real privacy preferences are not correctly represented due to the difficulty which users face when expressing their preferences. Thus, they propose a machine-learning based method to determine users' privacy preferences regarding their browsing behavior. The main technique used here is Case-Based Reasoning (CBR). It is similar to humans in terms of reasoning such that when it encounters a new case, it finds the most similar case from the past experience and then takes advantages of that past case. One more advantage of CBR is that it can explain the reasoning process explicitly to the user by expressing the similar cases to the new case. That is, it is human-readable. Another basic method of that study is collaborative filtering, which is the base of Amazon-like recommendation systems. It recommends similar solutions to users who have similar past experience. To define similarity, expert knowledge, similarity metrics and domain knowledge are used. Then, K-Nearest Neighbor (KNN) method is used to find the similar users/cases. When the user visits a new web site, the (machine readable) privacy policy of the web site is sent to reasoning engine together with the context of the user. In the engine, a privacy advice about sharing the information of a user is populated in order to be transmitted to the user, by employing historical cases (which have been learned), expert knowledge and information from the knowledge base of community portal (which uses collaborative filtering). User can accept or ignore that advice. In the case of ignoring, the user can correct the reasoning process or can add an exception to it. That feedback is also used in the learning part, thus it is added into the historical cases and expert knowledge. Additionally, the dynamism of privacy preferences is considered by giving advices based on the current information. Similar to our method, this method takes advantage of privacy knowledge of other agents. However, they aim to protect the privacy when surfing online while our approach is a solution for privacy protection in OSNs. That is, they focus on the control of the information which arises from a user's browsing a website while we aim a user to control the access to content that she has been shared in an OSN.

Calikli *et al.* propose Privacy Dynamics (PD), an architecture which learns privacy rules/norms regarding the sharing behavior of user [46]. The basic concepts of that architecture are conflicts and social identity (SI) map. Social identity (SI) is membership to a group and its meaning from that person's perspective. For example, being a student or being an academician, is a social identity. So SI map is an abstraction in which each social identity is modeled as a social group. Conflicts between two SI groups are defined as situations that an information should not flow between those groups. At the same time, they are representative of privacy norms, which will be learned with the help of this architecture. When bootstrapping PD, user's friends' information, her share history including posts shared and her friends lists are attained from her OSN account. SI Map is initialized based on her users' friend lists. Additionally, Conflicts are stored in PD as conflicting groups with attributes of object which conflict applies. This knowledge is initially empty before learning. When user initiates a share request, request is obtained by the app using PD and forwarded to PD. First of all, PD extracts attributes and potential audience of a post. Those attributes extracted are subject (user in the post), location (seven possible values such as office), time (four possible values such as day_time) and day (weekday and weekend). Then it queries SI map in order to learn the SI group of the subject. Using attribute values and SI group information of subject, it determines which SI groups can be conflicted in this situation. Then the result is compared with the potential audience of the object. The intersection is returned to user as unintended recipient. The learning of privacy norms is resolved with Inductive Logic Programming (ILP). The aim is to learn a logic program H, hypothesis explaining a set of positive and negative examples, in the context of another program B called background knowledge. A subset of Answer Set Programming (ASP), a language to search for that hypothesis in ILP, is employed here. Actually, the employed ASP programs are normal logic programs, containing normal rules, consisting of head and body. The body has two parts. While the first part contains the atoms which are tested against truth, the atoms in the second part are tested against fallacy. The rule means that if all atoms in the first part are true and those in the second part are false, then head is true. The ASP program corresponding to the model described here is made of four component: SI, Conf, Obj and Share. They are social identities, conflicts, objects and general sharing protocol, respectively. In SI, there are atomic facts regarding memberships. Conf program contains the conflicts as ASP normal rules. In Obj, there are facts corresponding to attributes of objects. In Share, there are two rules, which together means that if the SI group of subject of an object (post) conflicts with SI group of the potential sharer according to given conflict rules, then a conflict about sharing of that object by that sharer occurs. If it does not occur, then the object is shared by that person. The background knowledge B in this approach is the union of components except Conf. That is, B consists of SI, Obj and Share. Based on positive and negative examples of Share and the program B, the goal is to learn an H containing all positive examples and no negative examples if possible. If not, then containing positive example as much as possible and negative examples as few as possible. They approach the learning problem with a logic-based method while we do not. Additionally, they also do not employ other users' privacy preferences while ours can do.

Vanetti *et al.* propose a text-based approach to protect the wall of users in OSNs [47]. The method is specifically aimed to eliminate undesirable messages from the walls of users, which are the profile pages on which the messages can be shared by some other users depending on the social network. The approach consists of two phases: classification and filtering. The classification has two levels. The messages are firstly classified as neutral or nonneutral. The nonneutral messages are then softly classified as Violence, Vulgar, Offensive, Hate or Sex. In those classifications, textual features such as bag of words, number of bad words and so on are employed. Following the classification, whether the messages will be blocked or not are determined based on the filtering rules which are entered by users. Online Setup Assistant, a GUI-based software implemented for this study, helps users in generation of these filtering rules. In the filtering process, the properties of a message are compared with the conditions in the rules of the user, which is the owner of the wall. For example, the memberships of the message to the categories such as Violence, Vulgar etc., which is the output of the classification, is compared to the content specification in the rules. Then the action in the matching rule is applied. This approach does not aim to solve privacy problem. Since it is in the domain of OSN and it filters text-based OSN posts, the approach is similar to our method. This approach requires users to state their filtering rules beforehand and manually while ours aims to learn privacy preferences, which corresponds to filtering rules in this study.

Our work also differentiates from the preceding studies in the following way: While they generally focus only on the privacy, our work also considers the trade-off between the right to conceal an information and the gains of disseminating it. More specifically, our work takes the trade-off between the privacy and the utility into consideration. Thus, our system aims to estimate the potential utility of sharing an item by learning the relation of items with their utilities with the help of machine learning methods. Then, in the case of a private item, it informs its user about utility of sharing that item.

Krause and Horvitz present an approach which balances privacy-utility trade off [48]. It aims to maximize utility of an online service by sharing the minimum amount of private information. They focus on benefits, personalization in this scenario, and losses of sharing more personal data with online search engines. They use a probabilistic model in order to predict the target intention web page of the user. Then, the utility of sharing personal attributes of this user is defined as the increase in her information gain. The information gain is referred as the decrease in entropy (reflecting uncertainty). Thus, when the user provides the service with a subset of her personal attributes, the utility is the reduction in entropy of the probability of receiving the web page that she intends with her search query. At the same time, the cost of sharing private data is considered as the identifiability of the data owner. It is quantified as proportional to the probability of prediction of the user given the shared attributes of the user. Then, the goal of maximizing information gain while minimizing of identifiability is formulated as an optimization problem with the objective of the difference of utility and cost on the same scale. However, they claim that the solution to the problem is generally not tractable since the problem is NP-hard. When another new information about user is given to search engine, the utility does not increase as much as its previous increase with the previous addition of information. On the contrary, the cost (identifiability in this domain) increases more than the previous increase in the cost when another information about user is shared since information about user together helps to identify a user easier than only a single piece of information. This nature of the problem enables the usage of a near-optimal solution to the optimization problem. This approach is similar to our work in terms of considering privacy-utility trade off. However, their domain is different than ours, which affects the utility and cost functions.

Yassine and Shirmohammadi propose an agent-based framework to measure users' privacy payoff when disclosing their information [49]. The framework consists of five agents: facilitator agent, reputation agent, database agent, payoff agent and negotiation agent. The facilitator agent basically manages the communication between the internal agents, the one between the external and internal agents, and the one between the user and the internal agents. The reputation agent computes the reputation of a service provider. The reputation reflects the trustworthiness of that service provider. In the case of this study, how a service provider handles the private data of a user is reflected by the reputation. The reputation agent computes the reputation of a service provider based on user-provided reputation factors of each of its past transactions with this service provider. The database agent is responsible of determination of private information records. It firstly accesses the data specification of the user. Then the data is classified into categories and their subsets. For each data category, the user specifies context-dependent risk weights, which are values for private information for each context, by considering contexts and the reputation of service provider provided by the reputation agent. With respect to those weights, the number of subsets and revealed subsets in a category, the weighted privacy risk of dissemination of an information in a category under a context, namely privacy risk, is calculated. The payoff agent computes the payoff value of a private information at the current time. In order to compute the value, the problem is considered as Markov processes, of which the formulation helps to identify the payoff value through dynamic programming. Thus, the payoff value is calculated by employing the privacy weights with the help of Markov processes. Finally, the negotiation agent fundamentally generates proposals to the service provider depending on negotiation strategy of itself and the provider. This work is similar to our work that it explores the cases when the dissemination of private data is reasonable and how much the user (the owner of the data) can benefit from it. However, their valuation of private data is rooted in the values provided by users while our method aims to learn the utilities from past data of users. Additionally, their approach is general while ours is specific to OSNs, which makes possible for us to define specific utilities while privacy values defined by users in this study are general.

Our work here opens up interesting directions for further research. One direction is to add inference to the work. More specifically, in Example 1, Alice tries to achieve location anonymity by hiding her location explicitly from her friends. However, in reality, many other details could give away the fact that Alice is actually in the US. It would be cumbersome for the user herself to think about the ramifications for every post. It would be ideal if the agent could make the necessary inferences to help the user manage her privacy. The detection of privacy violations through inferences are studied in Kokciyan *et al.*'s work [10]. Consider the following example:

Example 6: Alice goes to a concert of her favorite band in NYC before meeting her friends. Since she is very excited, she shares the picture of her ticket with her friends. She has not explicitly checked in or shared her locations; but through the information on the ticket, her location can be revealed.

For an agent to warn Alice before sharing this, the agent needs to be able to process the information on the ticket and make inferences that the ticket will give away her location. In order for the agent to do this, we need to make use of the semantic layer in the system (e.g., the ontology), so that the agent can make further inference beyond traditional machine learning techniques.

Another important direction to consider is to enhance the trust layer among agents. Our current work is based on learning user's preferences and consulting other agents, from whom the information can be obtained. However, the choice of agents can be done in more sophisticated ways by employing trust modeling at various stages [50]. Instead of trying to predict the trustworthiness of an agent by asking known queries, one can model the trustworthiness over time. That way, even in a larger population, the right answers can be found by asking only a few agents. In our future work, we want to study how trust among agents can be used to decide on the agents to choose as well as how trust can be built based on how well privacy is preserved cooperatively.

In order to improve utility estimation, personalized user questionnaires can be designed with the aim of learning the motivation of likes and other interactions with posts. Then questions based on their OSN activities can be asked them. These questions can be general questions or those about the selected activities. After the learning motivations, the features employed in utility learning can be improved.

The procedure of persona assignment can be improved by collecting more data, which may also balance the number of individuals from various personas. Some personas in our current data have only few members; thus, we cannot rely on them. Thus, it would improve to conduct the data collection in a larger scale. Furthermore, the question in the questionnaire can be improved or the number of questions can be increased since users may misinterpret their own behavior when answering the current questionnaire.

REFERENCES

1. Westin, A. F., “Privacy and freedom”, *Washington and Lee Law Review*, Vol. 25, No. 1, p. 166, 1968.

2. Cutillo, L. A., R. Molva and T. Strufe, “Safebook: A privacy-preserving online social network leveraging on real-life trust”, *IEEE Communications Magazine*, Vol. 47, No. 12, 2009.

3. Squicciarini, A. C., H. Xu and X. L. Zhang, “CoPE: Enabling Collaborative Privacy Management in Online Social Networks”, *Journal of the American Society for Information Science and Technology*, Vol. 62, No. 3, pp. 521–534, 2011.

4. Stross, R., “How to lose your job on your own time”, `http://www.nytimes.com/2007/12/30/business/30digi.html?ex=1356670800&en=55ef6410d3cac28e&ei=5088&partner=rssnyt&emc=rss`, 2007, accessed at August 2017.

5. Gonzalez, J., “Cold Eagles sure are thin-skinned”, `http://www.philly.com/philly/sports/eagles/20090309_Gonzo___Cold_Eagles_sure_are_thin-skinned.html`, 2009, accessed at August 2017.

6. Henderson, M., M. De Zwart, D. Lindsay and M. Phillips, “Will u friend me? Legal risks and social networking sites”, `http://newmediaresearch.educ.monash.edu.au/lnm/wp-content/uploads/2015/05/SNSandRisks_EducationResource_0.pdf`, 2011, accessed at August 2017.

7. Bonneau, J., J. Anderson and G. Danezis, “Prying data out of a social network”, *Social Network Analysis and Mining, 2009. ASONAM’09. International Conference on Advances in*, pp. 249–254, IEEE, 2009.

8. Grasz, J., “Forty-five Percent of Employers Use Social Networking Sites to Research Job Candidates, CareerBuilder Survey Finds”,

http://www.careerbuilder.com/share/aboutus/pressreleasesdetail.aspx?id=pr691&sd=4/18/2012&ed=4/18/2099, 2012, accessed at August 2017.

9. Maternowski, K., “Campus police use Facebook”, https://badgerherald.com/news/2006/01/25/campus-police-use-fa/, 2006, accessed at August 2017.

10. Kökciyan, N. and P. Yolum, “PRIGUARD: A semantic approach to detect privacy violations in online social networks”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 28, No. 10, pp. 2724–2737, 2016.

11. Lampinen, A., V. Lehtinen, A. Lehmuskallio and S. Tamminen, “We’re in it together: interpersonal management of disclosure in social network services”, *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 3217–3226, ACM, 2011.

12. Liu, Y., K. P. Gummadi, B. Krishnamurthy and A. Mislove, “Analyzing facebook privacy settings: user expectations vs. reality”, *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pp. 61–70, ACM, 2011.

13. Huhns, M. and M. P. Singh (Editors), *Reading In Agents*, Morgan Kaufmann, San Fransisco, 1998.

14. Witten, I. H., E. Frank, M. A. Hall and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufmann, 2016.

15. Chang, C.-C. and C.-J. Lin, “LIBSVM: a library for support vector machines”, *ACM transactions on intelligent systems and technology (TIST)*, Vol. 2, No. 3, p. 27, 2011.

16. Li, D., “Java Implementation of Extreme Learning Machines”, http://www.ntu.edu.sg/home/egbhuang/elm_codes.html, accessed at December 2015.

17. Breiman, L., J. H. Friedman, R. A. Olshen and C. J. Stone, “Classification and

regression trees Belmont", *CA: Wadsworth International Group*, 1984.

18. Quinlan, J. R., "Induction of decision trees", *Machine learning*, Vol. 1, No. 1, pp. 81–106, 1986.

19. Breiman, L., "Random forests", *Machine learning*, Vol. 45, No. 1, pp. 5–32, 2001.

20. Quinlan, R. J., "Learning with Continuous Classes", *5th Australian Joint Conference on Artificial Intelligence*, pp. 343–348, World Scientific, Singapore, 1992.

21. Wang, Y. and I. H. Witten, "Induction of model trees for predicting continuous classes", *Poster papers of the 9th European Conference on Machine Learning*, pp. 128–137, Springer, Prague, Czech Republic, 1997.

22. Vapnik, V. N., *Statistical learning theory*, Vol. 1, Wiley New York, 1998.

23. John, G. H. and P. Langley, "Estimating continuous distributions in Bayesian classifiers", *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 338–345, Morgan Kaufmann Publishers Inc., 1995.

24. Huang, G.-B., Q.-Y. Zhu and C.-K. Siew, "Extreme learning machine: theory and applications", *Neurocomputing*, Vol. 70, No. 1, pp. 489–501, 2006.

25. Ravichandran, R., M. Benisch, P. G. Kelley and N. M. Sadeh, "Capturing social networking privacy preferences", *International Symposium on Privacy Enhancing Technologies Symposium*, pp. 1–18, Springer, 2009.

26. Mugan, J., T. Sharma and N. Sadeh, *Understandable learning of privacy preferences through default personas and suggestions*, Institute for Software Research Technical Report CMU-ISR-11-112, Carnegie Mellon University, Pittsburgh, PA, 2011.

27. "Company Info - Facebook Newsroom)", `https://newsroom.fb.com/company-info/`, accessed at June 2017.

28. Ugander, J., B. Karrer, L. Backstrom and C. Marlow, "The anatomy of the facebook social graph", *arXiv preprint arXiv:1111.4503*, 2011.

29. Brett, B., "The Psychology of Sharing: Why do People Share Online", *New York Times, New York. http://nytmarketing.whsites.net/mediakit/pos/*, Vol. 22, 2012.

30. Sparck Jones, K., "A statistical interpretation of term specificity and its application in retrieval", *Journal of documentation*, Vol. 28, No. 1, pp. 11–21, 1972.

31. Mikolov, T., K. Chen, G. Corrado and J. Dean, "Efficient estimation of word representations in vector space", *arXiv preprint arXiv:1301.3781*, 2013.

32. Bojanowski, P., E. Grave, A. Joulin and T. Mikolov, "Enriching word vectors with subword information", *arXiv preprint arXiv:1607.04606*, 2016.

33. Park, K., "Word Vectors", `https://github.com/Kyubyong/wordvectors`, 2017, accessed at August 2017.

34. "Clarifai General Image Recognition Model", `https://api.clarifai.com/v2/models/aaa03c23b3724a16a56b629203edc62c/outputs`, accessed at June 2017.

35. Kepez, T. B. and P. Yolum, "Synthetic Dataset of Privacy in Online Social Networks", `http://mas.cmpe.boun.edu.tr/privacylearner/ALICEDataset.arff`, 2016, accessed at June 2016.

36. Lewis, D., "Reuters-21578 dataset", `http://www.daviddlewis.com/resources/testcollections/reuters21578`, 1997, accessed at February 2016.

37. Horrocks, I., P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean *et al.*, "SWRL: A semantic web rule language combining OWL and RuleML", *World Wide Web Consortium Member submission*, Vol. 21, p. 79, 2004.

38. Mester, Y., N. Kökciyan and P. Yolum, "Negotiating Privacy Constraints in Online Social Networks", F. Koch, C. Guttmann and D. Busquets (Editors), *Advances in*

Social Computing and Multiagent Systems, Vol. 541 of *Communications in Computer and Information Science*, pp. 112–129, Springer International Publishing, 2015.

39. Gross, R. and A. Acquisti, "Information revelation and privacy in online social networks", *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, pp. 71–80, 2005.

40. Wang, Y., G. Norcie, S. Komanduri, A. Acquisti, P. G. Leon and L. F. Cranor, "I regretted the minute I pressed share: A qualitative study of regrets on Facebook", *Proceedings of the Seventh Symposium on Usable Privacy and Security*, p. 10, ACM, 2011.

41. Fang, L. and K. LeFevre, "Privacy wizards for social networking sites", *Proceedings of the 19th international conference on World wide web*, pp. 351–360, ACM, 2010.

42. Squicciarini, A. C., D. Lin, S. Sundareswaran and J. Wede, "Privacy policy inference of user-uploaded images on content sharing sites", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 27, No. 1, pp. 193–206, 2015.

43. Bilogrevic, I., K. Huguenin, B. Agir, M. Jadliwala, M. Gazaki and J.-P. Hubaux, "A machine-learning based approach to privacy-aware information-sharing in mobile social networks", *Pervasive and Mobile Computing*, Vol. 25, pp. 125–142, 2016.

44. Sadeh, N., J. Hong, L. Cranor, I. Fette, P. Kelley, M. Prabaker and J. Rao, "Understanding and capturing people's privacy policies in a mobile social networking application", *Personal and Ubiquitous Computing*, Vol. 13, No. 6, pp. 401–412, 2009.

45. Tondel, I. A., Å. A. Nyre and K. Bernsmed, "Learning privacy preferences", *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*, pp. 621–626, IEEE, 2011.

46. Calikli, G., M. Law, A. K. Bandara, A. Russo, L. Dickens, B. A. Price, A. Stuart, M. Levine and B. Nuseibeh, "Privacy dynamics: Learning privacy norms for social software", *Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 47–56, ACM, 2016.

47. Vanetti, M., E. Binaghi, E. Ferrari, B. Carminati and M. Carullo, "A system to filter unwanted messages from OSN user walls", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 25, No. 2, pp. 285–297, 2013.

48. Krause, A. and E. Horvitz, "A Utility-Theoretic Approach to Privacy and Personalization.", *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, Vol. 8, pp. 1181–1188, 2008.

49. Yassine, A. and S. Shirmohammadi, "Measuring users' privacy payoff using intelligent agents", *Computational Intelligence for Measurement Systems and Applications. CIMSA. IEEE International Conference on*, pp. 169–174, IEEE, 2009.

50. Şensoy, M., J. Zhang, P. Yolum and R. Cohen, "Poyraz: Context-Aware Service Selection under Deception", *Computational Intelligence*, Vol. 25, No. 4, pp. 335–366, 2009.