Computational Intelligence, Volume , Number 000,

A generic trust framework for large-scale open systems using machine learning

XIN LIU
École Polytechnique Fédérale de Lausanne (EPFL)

GILLES TREDAN
LAAS-CNRS, University of Toulouse

ANWITAMAN DATTA
Nanyang Technological University

In many large scale distributed systems and on the web, agents need to interact with other unknown agents to carry out some tasks or transactions. The ability to reason about and assess the potential risks in carrying out such transactions is essential for providing a safe and reliable interaction environment. A traditional approach to reason about the risk of a transaction is to determine if the involved agent is trustworthy based on its behavior history. As a departure from such traditional trust models, we propose a generic, machine learning based trust framework where an agent uses its own previous transactions (with other agents) to build a personal knowledge base. This is used to assess the trustworthiness of a transaction based on the associated features, particularly using the features that help discern successful transactions from unsuccessful ones. These features are handled by applying appropriate machine learning algorithms in order to extract the relationships between the potential transaction and the previous ones. Experiments based on real datasets show that our approach is more accurate than other trust mechanisms, especially when the information about past behavior of the specific agent is rare, incomplete or inaccurate.

Key words: trust management, machine learning, features, large-scale systems

1. INTRODUCTION

Trust is an important ingredient facilitating reliable interactions[1] among autonomous participants without global coordination in diverse large-scale systems including e-commence, distributed and peer-to-peer systems, multi-agent systems and dynamic collaborative systems. Due to the large scale and openness of these systems, an agent is often required to interact with other agents with which it has few or no shared past interactions. To assess the risk of such interactions and to determine whether an unknown agent is trustworthy, an efficient trust mechanism is necessary (Jøsang *et al.* (2007); Gambetta (1988)).

Traditional approaches, while effective when necessary information is available, often rely upon the knowledge that may not actually be available locally to the assessor. For instance, many traditional approaches rely on establishing a trust path (*i.e.* a chain of "trust relationships") between the assessor and the target agent. Such a chain is difficult to find in large systems, especially if few feedback is available about the target agent. To overcome such problems, we present a novel trust assessment approach that uses machine learning algorithms, exploiting the assessor's local transaction history to build a model of successful and unsuccessful transactions. Intuitively, if these algorithms manage to model efficiently what a successful (or unsuccessful) transaction is, we can then use them to predict the risks of a potential interaction. This new mechanism, given its use of different kind of information, is meant to complement traditional models. Given its reliance on different set of information, it should not and cannot be compared beyond the purpose of validation. So we view our work as an important ingredient in a decision support system, rather than a standalone and

[1] Throughout this paper, we use the terms interaction and transaction interchangeably.

foolproof security mechanism. Furthermore, the choice of local information to be used is application domain specific. Thus the choice of specific domain and kind of data for the evaluation was just for the purpose of determining the efficacy of the proposed approach. The main contributions of this paper are the concepts, rather than the specific instantiation that has been necessary for conducting experiments.

To explain how our approach works and differs from traditional trust mechanisms, let us use an analogy with human behavior in the real world. When a customer, say Bob, wants to evaluate the trustworthiness of a provider, say Sally's restaurant (i.e., to guess whether he will be satisfied by Sally's food), it is possible to consider two different cases:

- Bob has access to some feedback specifically about Sally's restaurant. For instance, Bob has already eaten there in the past, or maybe one of his friends did, and gave him some feedback about it. Then Bob uses this feedback to decide whether he will eat at Sally's restaurant or not. This is the case that traditional recommendation systems rely upon. More generally these recommendation systems will try to find feedback about Sally's restaurant among Bob's friends, or among the friends of Bob's friends (FoF), or from online restaurant review sites and so on. As we will see in the related works section, this method performs well, but has several drawbacks: *(i)* Bob has to somehow find a chain of friends that will provide him feedback, *(ii)* Bob needs to trust, and to have the same food taste as his feedback providers.
- In all other cases, for example when Bob is abroad or when Sally's restaurant just opened, Bob uses a different mechanism. Indeed, Bob will rely on his own experience about restaurants to make a first estimate on Sally's offer: he will look at the global presentation of the restaurant, the amount of other clients, the menu, the prices and the location. Then he will compare this with what he knows about other restaurants in the same city/country, of the same type, or where he previously had good/bad experience with the food he would like to order. This is essentially how our proposal works. Instead of relying on a chain of recommendation to derive trust, our approach relies on a user's (Bob) personal experience. Of course, the success of this assessment relies on two hypotheses: (i) the user has enough experience, and (ii) it is possible to learn and predict, from these experiences, the outcome of a new transaction. One may note that many product recommendation systems, particularly those applying collaborative filtering, work on a similar premise. Results based on data (from an auction site) driven experiments show that these assumptions do also hold in (some) real scenarios, where our proposed approach achieves good and efficient prediction of auction frauds.

This work demonstrates that machine learning can be used for modeling and quantifying trust. That of-course does not mean that machine learning can always be used, since its usage still depends on the availability of appropriate information, which (the information) however happens to be somewhat different from the kind of information used in traditional trust models. Thus to say, we are exploring ways to exploit information that is often ignored in traditional trust models. The proposed approach is generic in that a multitude of machine learning algorithms can be utilized, either to replace or boost the ones we have currently explored.

We model an agent's local knowledge (i.e., past interactions) by a feature vector. Without loss of generality, we assume two classes: successful and unsuccessful interactions. Note that all interactions have the same feature vector. To present and validate our proposal, we rely on two common machine learning algorithms: linear discriminant analysis (LDA) (McLachlan (2004); Fukunaga (1990)) and decision trees (DT) (Quinlan (1986); Mitchell (1997)). LDA is a well known method for dimensionality reduction and classification. It takes as input a set of events belonging to k ($\geqslant 2$) different classes and characterized by various features, and finds a combination of the features (a classifier) that separates these k classes of events. As an

example, linear discriminant analysis was used to differentiate subspecies of beetles based on the measurements of their physical characteristics. DT is a widely used and practical method for classification and prediction by providing a classifier in the form of a tree structure. Its popularity is due to the ability of generating rules which can be easily expressed visually and in human language. Although both methods are simple, as seen in the real auction data based experiments, they are both effective and efficient. More sophisticated methods (e.g., multiple discriminant analysis) may also be applied but the associated trade-offs are not explored in this paper.

Since different machine learning algorithms may produce different results in the same scenario (due to the variations in their suitability), we also investigate the confidence of each algorithm recommendation to help the trustor make a wise decision. Our framework is designed for decentralized systems: each agent uses its own local knowledge to evaluate the trustworthiness of a transaction he might get involved in. However, it can be also applied in a centralized setting, where all agents' local knowledge is gathered in a system-wide knowledge repository. In this context, multiple local knowledge can be aggregated to issue a better trust assessment.

It is worth mentioning that human intervention, offline policies, as well as legal issues may be considered to guarantee wise decision making. However, in this work, we focus on the abstract computational model, and try to develop an efficient trust framework to automatically derive trust and make recommendations to support decision making by leveraging machine learning algorithms. Thus to say, if there is adequate information specific to the target agent's past behavior, one definitely should use that (as is the case with traditional trust models). Our approach targets the situations where such information is absent or scarce.

Note that since this work is particularly developed for evaluating the trustworthiness of the transactions that are conducted by unknown service providers, we do not discuss how to maintain trust ratings of known agents, which can be easily handled by any traditional trust management mechanism.

The primary contribution of this paper is to define a trust framework based on machine learning for large-scale open systems. We rely on machine learning algorithms (in this paper, LDA and DT), and argue that the trustworthiness of a potential transaction may be efficiently learned using the trustor's local knowledge/information that is often ignored in traditional trust models. The algorithms' recommendation confidence is studied to make decisions wisely. This work should thus be treated as a complimentary mechanism to provide prediction for a potential interaction, especially when sufficient (reliable) global information is not available. Furthermore, compared to existing reputation based trust models, this work is more robust: (1) it mainly relies on the trustor's own local knowledge thus lowering the risk of suffering from inaccurate third party information; (2) the features are difficult to fake since a malicious agent is unaware of (or much efforts are needed to investigate) a trustor's local knowledge, thus the adversary may find it difficult to suitably modify the features to mislead the trustor. Furthermore, if some information (feature) can be easily faked, then from the perspective of machine learning, that feature's discriminating power will be marginal, and it will automatically have very little, if any, impact on the decision process. Evaluation results show that compared to other trust models, this approach is efficient, especially when third party information is not reliable. The evaluation based on datasets from real auctions (Allegro and eBay) also demonstrates its efficacy in a realistic setting, namely, for detecting auction frauds.

The rest of this paper is organized as follows: Machine learning based trust framework is introduced in Section 2. In Section 3, we demonstrate how our proposal works using LDA and DT as the case studies in subsections 3.1 and 3.2 respectively. The confidence of algorithm recommendation is studied in Section 4. A discussion about related issues is presented in Section 5. Section 6 reports results from evaluations using both real online

auction datasets and synthetic dataset. In Section 7 we present related works regarding trust management. Finally we conclude and suggest future research directions in Section 8.

2. A GENERIC TRUST FRAMEWORK

Unlike most existing works on computational trust, our approach does not focus on an agent's past behavior to decide whether or not to interact with him. Instead, it focuses on the characteristics of the transaction currently being proposed by that agent. We investigate such an approach primarily to be used when information about the specific agent's behavior is scarce. This incidentally also avoids the risk of suffering from inaccurate third party information.

2.1. Notation

We refer to a participant in the system as an agent. We denote by $\mathcal{A}$ the set of all agents in the system. Each agent can play two roles: *customer*, which requests services from other agents and *provider*, which provides services. An agent can be both a customer and a service provider, but in a given transaction, it plays only one of the roles.

A *transaction* Θ_{a_x,a_y} happens when a customer a_x accepts a service from provider a_y. To indicate the quality of a service, an agent can rate the transaction. In our model, the rating of a transaction is binary: successful or unsuccessful. Note that without loss of generality, each Θ represents a unique transaction. A transaction Θ is described by a feature vector, denoted by $F_\Theta = \{f_\Theta^1, f_\Theta^2, ..., f_\Theta^d\}$, and an outcome. A potential transaction is a transaction that has not occurred yet, and therefore has no outcome.

Each agent a_x in the system maintains a transaction database $Tr_{a_x} = \{\Theta_{a_x,a_1}, \Theta_{a_x,a_2}, ...\}$, which records all historical transactions it conducted with other agents. This local knowledge is used as training data by machine learning algorithms to predict the trustworthiness of a potential transaction.

2.2. Feature collection

Features can be collected from the transaction partner's profiles or from the transaction's context, depending on specific applications. Let us take online auction as a concrete example to demonstrate how features may be collected. As an example, consider a buyer, Bob, that evaluates a potential transaction such as buying a camera from Sally, a seller. Features representing this transaction may be extracted from 3 categories:

(1) about Sally herself, e.g., age, gender, is she from the same country as bob? physical distance to bob, etc.
(2) about Sally as a user in the system, e.g., system age, the numbers of successful and unsuccessful transactions, has Sally provided a phone number? How complete is her profile? The number of items Sally already sold, the average delivering time, the average time between end of auction and user comment, the number of friends Sally has, etc.
(3) the cameras Sally sells, e.g., the average item price (items in the same category), the number of the same items in stock, the number of comments on these cameras, the number of different buyers that already placed a bid on it, the average age of the buyers, etc.

Let us observe that Sally can cheat and set fake information about herself, but faking information from the $2nd$ and $3rd$ aforementioned categories is harder, since such information is maintained by the service exchange platform. Moreover, if available and appropriate, features can also include some historical information (e.g., numbers of successful and unsuccessful transactions, etc.) that is specific to the target agent. Although the information used

here is similar to the one exploited in traditional trust mechanisms, it is exploited in a quite different way. Indeed the information is not directly aggregated to determine a reputation from which a trust level could be derived, but is instead used as a feature to help extract the relationships between a potential transaction and the past successful or unsuccessful transactions.

We argue that compared to existing reputation based trust models, the feature collection process is more robust against malicious agents. For a traditional trust model, when the trustor requests opinions about the target agent, malicious entities can easily counterfeit the feedback to promote or disparage the target agent. Differently, for our approach, since attackers are unaware of the trustor's local knowledge, it is difficult to fake features associated with the potential transaction to mislead the trustor. Furthermore, features that are easy to fake will ultimately be considered as useless by the machine learning algorithms. Our proposal is thus more resilient to attacks that are common to existing reputation based trust systems.

2.3. Architecture

Fig. 1 depicts the architecture of our proposal. The framework consists in three components: the storage component (**SC**), the trust calculation engine (**TCE**) and the knowledge collector (**KC**).

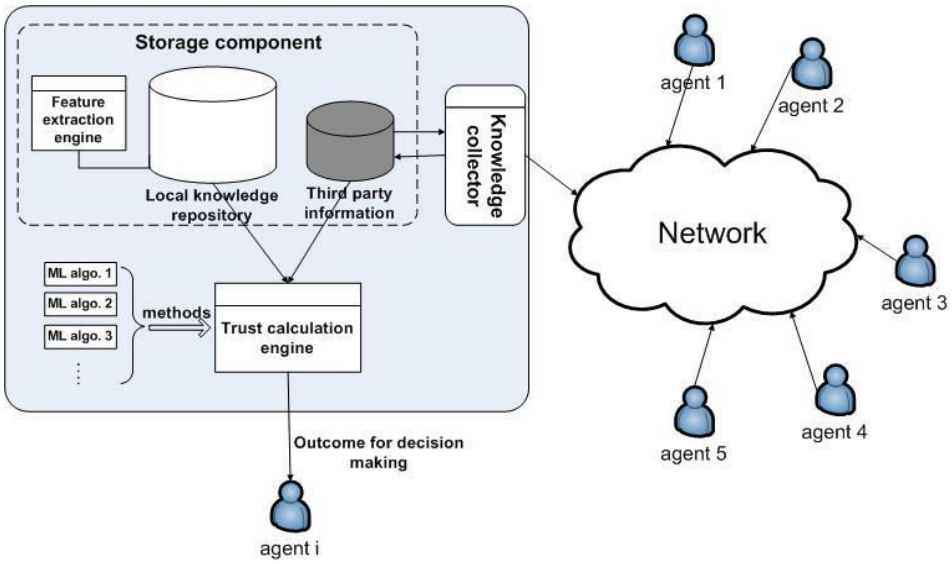


FIGURE 1. The architecture of the trust framework.

The storage component (**SC**) is responsible for managing the local knowledge. It records all transactions conducted by the agent as a (feature vector, outcome) tuple.

The proposed trust framework is generic in two senses. Firstly, since the approach can be applied for different applications and correspondingly diverse features. This includes not only traditional applications of trust like mitigating malicious behavior or intent of other agents, but also applications to reliable distributed systems where trust is treated as an abstraction to learn and adapt the system, without explicitly attempting to formulate or compute objective functions, which may be difficult in large-scale systems, either because of the absence of global knowledge, or any centralized coordinator. Some preliminary results of the utilization of our trust model in optimizing a P2P storage system's robustness can be found in (Liu and Datta (2012)). Secondly, since diverse machine learning approaches may be

TABLE 1. Structure of local knowledge repository

Tran. ID	Tran. partner	outcome	Price of the item	♯ of items already sold	...
Θ_1	a_1	1	6.99	891	...
Θ_2	a_2	0	9.99	337	...
Θ_3	a_3	0	12.99	120	...
.....					

plugged into the framework. **TCE** is responsible for applying machine learning algorithms to predict the trustworthiness of the potential transaction using local knowledge or collected knowledge (by **KC**[2]) as the training data. Depending on the application scenario, **TCE** is able to choose the appropriate machine learning algorithms to conduct trust calculation. For instance, if the features are quantitative, discriminant analysis will be used, otherwise, if several features are qualitative, decision tree may be more suitable. In fact, subject to time/computation restrictions, **TCE** applies as many suitable algorithms as possible. In case the result is not consistent across algorithms, it returns the result that is calculated by the algorithm which has the highest confidence. The measurement of such confidence is algorithm dependent and is described in Section 4. The design of other ways of aggregating different results from different learning algorithms (e.g., weighted sum, majority, applying ensemble learning, etc.) are unexplored, and we defer such a study for the future.

Next we present the details of the machine learning based trust assessment algorithms.

3. MACHINE LEARNING ALGORITHMS FOR TRUST ASSESSMENT

A machine learning algorithm for trust assessment is trained using the (feature vector, outcome) tuples stored by the storage component. It then takes the feature vector of a potential interaction as input, and outputs the likelihood of generating a successful (or unsuccessful) outcome, along with a recommendation confidence level (see Section 4).

It is worth pointing out that among all features, the features that will have an impact on the recommendation are eventually selected based on their discriminating powers, i.e., the extent to which a feature is able to distinguish successful transactions from unsuccessful ones. The benefit of applying machine learning algorithms is to automatically decide which features are useful, and which are not: features that have no discriminating power will simply have little impact on the final recommendation.

Two common but effective machine learning algorithms, linear discriminant analysis (LDA) and decision tree (DT) are used as case studies to demonstrate the framework.

3.1. Case study: Using linear discriminant analysis

Consider a scenario where a service customer a_x encounters a potential service provider a_y and a_x has no prior experience with a_y. We assume that a_x can obtain the features of this potential transaction Θ_{a_x,a_y} (e.g., from a_y's profile and from the transaction offer it received). The vector of such features of Θ_{a_x,a_y} is denoted by $F_{\Theta_{a_x,a_y}} = \{f^1_{\Theta_{a_x,a_y}}, f^2_{\Theta_{a_x,a_y}}, ..., f^d_{\Theta_{a_x,a_y}}\}$.

[2]Knowledge collector component (**KC**) is designed to handle bootstrapping issue by collecting information from other system participants. This issue will be further discussed in Section 5.

To simplify the notation, we use f^i instead of $f^i_{\Theta_{a_x,a_y}}$ when the context is clear. So the potential transaction is represented by a feature vector $p = (f^1\ f^2\ f^3 \dots f^d)$.

We assume that a_x has recorded n historical transactions. Note that we assume an agent is aware of the context of the transaction, for instance, when evaluating an unknown book seller's potential transaction, the buyer only considers past transactions regarding purchasing books, not that regarding purchasing other irrelevant items like digital cameras. To estimate the trustworthiness of the potential transaction, a_x classifies its historical transactions into two disjoint groups, the *successful transaction group* G_s and the *unsuccessful transaction group* G_u, according to the outcomes of these transactions. The two transaction groups are represented as:

$$\mathbf{G_s} = \begin{pmatrix} f^1_{\Theta_{a_x,a_1}}(s) & f^2_{\Theta_{a_x,a_1}}(s) & ... & f^d_{\Theta_{a_x,a_1}}(s) \\ \vdots & \vdots & \vdots & \vdots \\ f^1_{\Theta_{a_x,a_{n_s}}}(s) & f^2_{\Theta_{a_x,a_{n_s}}}(s) & ... & f^d_{\Theta_{a_x,a_{n_s}}}(s) \end{pmatrix}, \tag{1}$$

$$\mathbf{G_u} = \begin{pmatrix} f^1_{\Theta_{a_x,a_1}}(u) & f^2_{\Theta_{a_x,a_1}}(u) & ... & f^d_{\Theta_{a_x,a_1}}(u) \\ \vdots & \vdots & \vdots & \vdots \\ f^1_{\Theta_{a_x,a_{n_u}}}(u) & f^2_{\Theta_{a_x,a_{n_u}}}(u) & ... & f^d_{\Theta_{a_x,a_{n_u}}}(u) \end{pmatrix}. \tag{2}$$

n_s and n_u are the sizes of the successful transaction group and the unsuccessful transaction group respectively, and therefore $n = n_s + n_u$. Let h_d be a $d \times 1$ (column) vector of ones. To perform a LDA, a_x first calculates the centroid of each group. That is, for each feature, we calculate average value of this feature across all transactions in the group:

$$c_s = \frac{1}{n_s} \cdot h^T_{n_s} G_s \text{ and } c_u = \frac{1}{n_u} \cdot h^T_{n_u} G_u. \tag{3}$$

Similarly, the global centroid is calculated by averaging each feature across all past transactions:

$$c = \frac{1}{n} \cdot h^T_n \begin{bmatrix} G_s \\ G_u \end{bmatrix}. \tag{4}$$

In LDA, the internal variance (within-class scatter matrix) and the external variance (between-class scatter matrix) are used to indicate the degree of class separability, i.e., to what extent are the successful transactions distinguished from the unsuccessful ones. The internal variance, which is the expected covariance of each group is given by Eq. 5 and 6:

$$S^s_w = \frac{1}{n_s}(G_s - h_{n_s} c_s)^T (G_s - h_{n_s} c_s), \tag{5}$$

$$S^u_w = \frac{1}{n_u}(G_u - h_{n_u} c_u)^T (G_u - h_{n_u} c_u). \tag{6}$$

Vectors h_{n_s} and h_{n_u} are used to center the values around the centroids c_s and c_u. The overall within-class scatter matrix is calculated as the average of each group's internal variance, weighted by the respective size of each group:

$$S_w = \frac{1}{n}(n_s S^s_w + n_u S^u_w). \tag{7}$$

Then a_x calculates the external variance, which is actually the covariance of the two groups, each of which is represented by its mean vector (Eq. 8):

$$S_b = \frac{1}{n}(n_s(c_s - c)^T(c_s - c) + n_u(c_u - c)^T(c_u - c)). \tag{8}$$

The mixture scatter matrix, i.e., the total variance of the system is obtained by summing up the internal variance S_w and the external variance S_b (Eq. 9) is:

$$S_m = S_w + S_b. \tag{9}$$

LDA finds a projection direction (a transformation) v that maximizes the inter class variance and minimizes the intra class variance. Formally, the criterion function (see Eq. 10) is to be maximized:

$$J(v) = \frac{v^T S_b v}{v^T S_w v}. \tag{10}$$

The projection direction v is found as the eigenvector associated with the largest eigenvalue of $S_w^{-1} S_b$. We then transform the two groups of transactions using v. Similarly, the potential transaction $p = (f^1\ f^2\ f^3 \ldots f^d)$ is also transformed and classified by measuring the distances between the transformed potential transaction and the two groups (i.e., centroid). Eq. 11 and 12 show how to calculate such distances:

$$D_s = v^T p - v^T c_s, \tag{11}$$

$$D_u = v^T p - v^T c_u. \tag{12}$$

If $D_u > D_s$, then transaction p is predicted to be successful, otherwise it is predicted to be unsuccessful.

Note that our approach tries to collect as many features of a transaction as possible, and the algorithm filters out the not-so-relevant variables. That is to say, the feature which is more capable of distinguishing successful transactions from unsuccessful ones will have more impact on the final classification result.

3.2. Case study: Using decision tree

In a nutshell, in a decision tree each node (except the leaf nodes) represents some feature of the object. Each edge linking this node to its children corresponds to a value (or a range) of this feature. Therefore, the feature vector values of a potential transaction define a path in the decision tree, leading from the root to a leaf node, that represents the decision (trustworthy or not).

A decision tree is built using past transactions. From a given past transactions set, it is possible to construct several different decision trees, depending on the order in which the features are tested. Since building an optimal decision tree (optimal with respect to its size) is NP hard (Hyafil and Rivest (1976)), several heuristics have been proposed to build efficient learning trees (e.g., ID3 Quinlan (1986), C4.5 Quinlan (1993), etc.). Without loss of generality, we hereafter present ID3 (Quinlan (1986)) which is a well known algorithm that relies on information gain (MacKay (2003)) to select the classifying features at each node of the tree. Information gain is measured by entropy. Let n_s and n_u be the proportion of successful and unsuccessful past transactions. The entropy of all past transactions is:

$$Entropy(Tr_{a_x}) = -n_s log_2(n_s) - n_u log_2(n_u). \tag{13}$$

Entropy is used to characterize the (im)purity of a collection of examples. From Eq. 13

we can see the entropy will have the minimum value of 0 when all past transactions belong to one class (i.e., successful or unsuccessful) and the maximum value of $log_2 2 = 1$ when past transactions are evenly distributed across the two classes. Using entropy, the information gain of every feature with respect to a set of past transactions Tr can be calculated to determine the best feature to choose for a node in the decision tree. For each feature f_i, we assume it has a set of values (e.g., discrete variable) or intervals (e.g., continuous variable), which is denoted by $V(f_i)$. For each $v \in V(f_i)$, we denote the set of past transactions that are associated with v for feature f_i by Tr_v. The information gain of feature f_i is thus calculated by:

$$IGain(Tr, f_i) = Entropy(Tr) - \sum_{v \in V(f_i)} \frac{|Tr_v|}{|Tr|} Entropy(Tr_v). \tag{14}$$

The information gain of a feature measures the expected reduction in entropy by considering this feature. A high information gain translates into a low entropy: considering the corresponding feature improves the classification.

Algorithm 1 Decision tree construction (code for the trustor a_x).

1: Trust_ID3(Tr_{a_x},F)
2: **if** All past transactions are successful **then**
3: Return single node tree *Root*, with label = +.
4: **end if**
5: **if** All past transactions are unsuccessful **then**
6: Return single node tree *Root*, with label = –.
7: **end if**
8: **if** $F = \emptyset$ **then**
9: Return single node tree *Root*, with label = outcome of the transactions that dominate all past transactions.
10: **end if**
11: Calculate information gain of all features and choose one feature (say, f), which has the highest information gain as the *Root* node.
12: **for** each value v of f **do**
13: Add a new branch below *Root*.
14: Let $Tr_{a_x,f,v}$ be the set of past transactions that have value v for feature f.
15: **if** $Tr_{a_x,f,v} = \emptyset$ **then**
16: Below this branch add a leaf node with label = outcome of the transactions that dominate all past transactions.
17: **else**
18: Below this branch add a sub-tree:
19: Trust_ID3($Tr_{a_x,f,v}$,$F - \{f\}$).
20: **end if**
21: **end for**
22: **Return** *Root*.

A decision tree is iteratively constructed by maximizing information gain at each step. We next describe how the decision tree is constructed (Algo. 1) using ID3 algorithm.

The algorithm's input is the trustor a_x's past transactions Tr_{a_x}, each of which is described by a feature vector F and an outcome. If all transactions are successful or unsuccessful, the algorithm simply returns a single node (i.e., root) tree, labeled as '+' or '–' respectively (Line 2-7). If the feature vector is empty, the algorithm returns a single node

tree, labeled as '+' if most of past transactions are successful or '–' otherwise (Line 8-10). The information gain of each feature is computed using Eq. 14. The feature (say, f) with the highest information is selected as the root of the tree. Next, new branches are added to the root node according to the possible values of f. For each possible value v, we denote the set of past transactions that have v for f by $Tr_{a_x,f,v}$. If $Tr_{a_x,f,v} = \emptyset$, the algorithm adds a leaf node to the corresponding branch with label '+' or '–' if most of past transactions are successful or unsuccessful respectively. Otherwise, the algorithm recursively executes itself with $Tr_{a_x,f,v}$ and $F - \{f\}$ as input (Line 12-21). The algorithm terminates either when all features have been tested or when the tree perfectly classifies all past transactions.

When encountering a potential transaction, a_x tries to classify it as a risky transaction or not by comparing the corresponding features from the root of the decision tree down to a certain leaf node.

4. RECOMMENDATION CONFIDENCE

Given a set of past transactions and an appropriate machine learning algorithm, our trust framework is able to return a recommendation on a potential transaction, namely risky or not. The confidence of such a recommendation relies on the characteristics of input (e.g., discriminating power of the features, number of past transactions) and the performance of the specific machine learning algorithm being used[3]. Let $\sigma \in [0, 1]$ be the algorithm recommendation confidence, where 0 means not confident at all (i.e., the recommendation is equally random) and 1 means completely confident. Algorithm input is denoted by $< F, Tr >$, where $F = \{f_1, f_2, ...\}$ represents the feature vector and $Tr = Tr_s \cup Tr_u$ represents the set of past transactions: $Tr_s = \{\theta_1^s, \theta_2^s, ...\}$ and $Tr_u = \{\theta_1^u, \theta_2^u, ...\}$ are the sets of successful transactions and unsuccessful transactions respectively. We next present how to assess the algorithm's recommendation confidence.

4.1. Measuring recommendation confidence for LDA based algorithm

When applying linear discriminant analysis (refer to Section 3.1), each transaction is described by input feature vector F, i.e., each transaction is represented as a point in a $|F|$-dimensional space. The confidence of the algorithm recommendation largely depends on the discriminating power of the input features. If all features are discriminating enough, LDA is able to clearly separate the successful and unsuccessful transactions. Otherwise, the two transaction groups Tr_s and Tr_u may overlap, therefore reducing the recommendation confidence.

Since the potential transaction is classified as risky or not by measuring its distance to the two transaction group centroids in the $|F|$-dimensional space, we can measure the recommendation confidence based on these distances. Let f be the feature vector of a potential interaction, and let $D_s(f)$ and $D_u(f)$ be the distances between f and the successful/unsuccessful transaction group centroids. The confidence of the algorithm recommendation is then calculated as:

$$\sigma_{LDA}(f) = \frac{|D_s(f) - D_u(f)|}{D_s(f) + D_u(f)}. \tag{15}$$

It is clear from Eq. 15 that when $D_s(f)$ = $D_u(f)$, then $\sigma_{LDA}(f) = 0$, i.e., the potential transaction is classified as risky or not with the same probability. When $D_s(f)$ or $D_u(f)$ is

[3]Given the same input, different machine learning algorithms may generate different recommendations (perhaps due to their suitability in different scenarios).

0, $\sigma_{LDA} = 1$, i.e., the potential transaction is recommended to be safe or not confidently. The trustor can thus use $\sigma_{LDA}(f)$ to reason about how reliable the trust algorithm's recommendation is.

4.2. Measuring recommendation confidence for DT based algorithm

Since a decision tree are constructed using the information gain of each feature (refer to Section 3.2), it is natural to measure the algorithm recommendation confidence using the features' information gains. Recall that the information gain of a feature measures the expected reduction in entropy by considering this feature. If a feature is discriminating enough, the corresponding information gain is high (maximum value being $log_2 2 = 1$), i.e., the values of this feature clearly distinguish between a successful and an unsuccessful transaction. Therefore the potential transaction can be accurately classified by this feature. Otherwise, this feature may not help to classify the potential transaction accurately.

For each feature f_i (with respect to a set of transactions Tr_i), we calculate the corresponding entropy $Entropy(Tr_i)$ and the information gain $IGain(Tr_i, f_i)$ using Eq. 13 and 14 respectively. The entropy reduction r_i obtained using feature f_i is:

$$r_i = \frac{IGain(Tr_i, f_i)}{Entropy(Tr_i)}. \tag{16}$$

r_i can be interpreted as the accuracy of the classification of the potential transaction by feature f_i (i.e., at the node with f_i in the decision tree.). We notice from Eq. 16 that if f_i is extremely discriminating, its information gain is $Entropy(Tr_i)$ (i.e., the second part of Eq. 14) is 0, and thus r_i is 1, which means the classification by f_i is completely accurate. In contrast, if f_i has no discrimination power, its information gain is 0 and thus r_i is 0: the classification of the potential transaction is completely random.

After having the accuracy of the classification by each feature, the algorithm's overall recommendation confidence is calculated as:

$$\sigma_{DT} = \prod_{i=1}^{|F|} r_i. \tag{17}$$

So, if all features are quite discriminating, the algorithm recommendation confidence is approximating 1. Otherwise, any less discriminating feature shall evidently impact the final confidence. This is consistent with the decision rule of the decision tree based algorithm.

5. DISCUSSION: DEALING WITH THE SPARSENESS OF LOCAL INFORMATION

So far, we have presented how our machine learning based trust framework works, along with an analysis of the confidence of the algorithms. Our approach works well under the assumption that the trustor has sufficient local knowledge such that an accurate classifier can be trained. However, in large-scale open systems, an agent may have insufficient local interaction information — this agent is new in the system, or it only occasionally interacts with others. In order to address this bootstrapping issue, some mechanisms are needed to help the inexperienced agents create local knowledge repository for reliable trust assessment. Specifically, there are two obvious straw man approaches to bootstrap agents depending on application scenarios. (1) If the trust framework works on top of centralized systems (i.e., all information is stored and managed by a central entity), an inexperienced agent is able to collect all relevant interaction information from other agents. For instance, an agent may

collect transaction logs in individual categories or from certain agents (e.g., the ones that are highly trustworthy). Such information is combined to construct the agent's local knowledge that is used for classifier training. (2) In the case that no central entity is available, agents may share local knowledge to help bootstrap the inexperienced agents. A traditional method is to resort to a certificate authority (CA) to ensure secure and reliable information exchange. However, such a trusted third party service is not always available in practice. Alternatively, agents may form a self-organized network (e.g., Peer-to-Peer) to support for agents' local knowledge sharing.

It is also worth mentioning that we assume an identical vector of features for all interactions. That is, the local knowledge from multiple agents can be naturally combined. However, in some scenarios, different agents may use different feature vectors to characterize interactions, making local knowledge combination non-trivial. In order to handle this issue, transfer learning technique (Pan and Yang (2010)), which bridges the gap between different domains (with different feature spaces) may be applied to meaningfully merge local knowledge with different feature vectors. We leave as a future work a more detailed exploration on how transfer learning algorithms can be systematically integrated into our trust framework and how this technique influences the performance of our approach.

Thus to conclude, finding robust ways to bootstrap a computational trust system remains an open challenge - and while our work manages to deal with the sparseness of (global) feedback information by harnessing other (locally available) information often ignored in traditional trust models, and can thus help on many occasions, as mentioned at the outset, it in itself is not a self-contained mechanism, but only plays a complementary role to traditional trust systems. In particular, a prominent direction for future work is to explore better mechanisms (than the above outlined straw man approaches) to deal with the situations when locally available information is also sparse.

6. EVALUATION

6.1. Simulation settings

We use data collected from Internet auction sites Allegro (`http://allegro.pl/`) and eBay (`http://www.ebay.com/`) to drive our experiments.

The Allegro dataset contains 10,000 sellers, 10,000 buyers, more than 200,000 transactions and over 1.7 million comments. A transaction is considered to be successful if its feedback is positive, otherwise, it is considered to be unsuccessful. We extract three features empirically (When the ratio between the average value of a feature for successful transactions and unsuccessful transactions is larger than a threshold, this feature is considered to be "useful") from Allegro data, i.e., F_1: the category of the item; F_2: the price of the item and F_3: the number of items already sold by the seller when the transaction occurs. We evaluate the performance of our proposal, i.e., LDA based approach and DT based approach, by studying their capabilities of detecting Internet auction fraud. When a buyer encounters a potential transaction, which is offered by an unknown seller, this buyer will gather past transactions regarding the item category (i.e., F_1) and then perform LDA or DT to estimate the trustworthiness of this transaction using features F_2 and F_3. Note that some related works also applied machine learning techniques to mine electronic auctions (e.g., predicting end prices of online auctions (Ghani and Simmons (2004))). Differently, our approach focuses on predicting a seller's behavior in the potential transactions, and particularly, when the target seller's behavioral history information is unavailable.

In order to assess the applicability of our approach in different systems (but for the same application domain of online auctions), we also conduct experiments using another real dataset collected from eBay. We randomly selected $23,238$ sellers from different categories

and collected their past transactions ($969,213$ with positive ratings, 1914 with neutral ratings and 3590 with negative ratings) conducted from April 10, 2000 to June 4, 2011. The same simulation settings mentioned above (for Allegro data) are applied.

Both Allegro and eBay have a very friendly environment: very small fraction of the transactions are explicitly labeled as negative. To test our approach in a more hostile environment, and also to compare the proposed algorithms with other trust mechanisms, we generates a synthetic dataset consisting of 10,000 nodes. Each node is either *good*, which provides services satisfactorily or gives genuine feedback when requested, or *malicious*, which cheats others in transactions or provides false feedback when requested. To make the simulation environment more realistic, there are no completely *good* or *bad* nodes. We set *good* nodes and *malicious* nodes to behave maliciously with the probabilities of 15% and 85% respectively. We denote the fraction of malicious nodes in the system by P_m. In the simulation, once a transaction is done, the trustor rates the transaction to indicate whether it is successful or not. The trustor stores the related information of each past transaction such as its outcome and the associated features (see Table 1).

We generate four[4] synthetic features to describe each transaction. Let X_s and X_u be the random variables which represent values of features of successful and unsuccessful transactions respectively. We assume that X_s and X_u follow normal distribution, i.e., $X_s \sim \mathcal{N}(\mu, \sigma)$ and $X_u \sim \mathcal{N}(\mu + \theta, \sigma)$. We set $\mu = 1$, $\sigma = 0.1$ and $\theta \in [0, 1]$ is the separability factor that we use to tune the overlap between the values of features for successful transactions and that for unsuccessful ones. When $\theta = 0$, the values of features for successful and unsuccessful transactions overlap completely, while when θ is large enough, the values of features for successful and unsuccessful transactions are separated clearly. In other words, θ allows us to modify the inherent learnability of the transaction classes. Note that since only our approaches are based on features, θ does not impact the performance of other approaches.

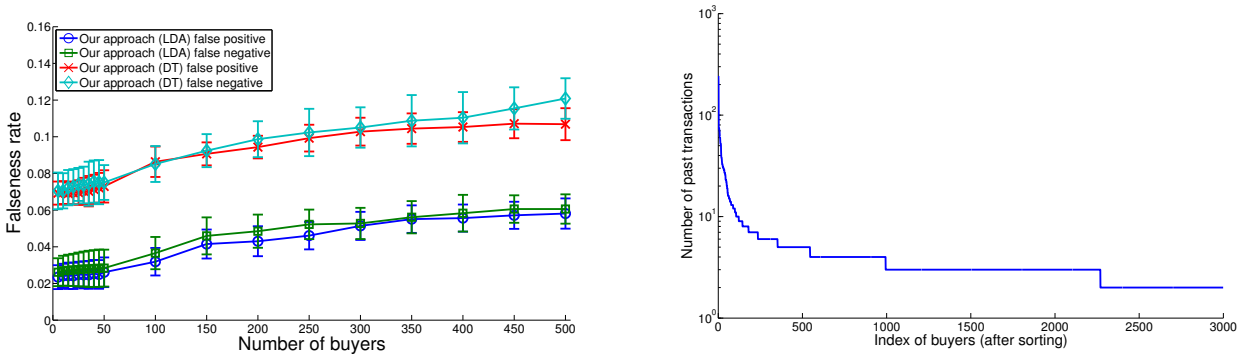
We compare our proposal with three existing trust models: (1) *Random Selection*. The simplest approach is to engage or reject a potential transaction randomly. In the simulation, we set the probability that the trustor accepts a potential transaction to 50%. This model is used as a benchmark. (2) *Feedback Aggregation*. In this model (Jøsang and Ismail (2002); Xiong and Liu (2004); Teacy *et al.* (2006)), if the trustor does not know the target node, it asks other nodes across the network and aggregates the feedback to derive the target node's trustworthiness. False feedback filtering mechanism (from TRAVOS (Teacy *et al.* (2006))) is applied to improve the performance. (3) *StereoTrust* (Liu *et al.* (2009)). The trustor forms groups of agents with which it has past experience according to the features, and derives the trustworthiness of the target agent by combining the trustworthiness of the groups to which the target agent belongs. Please refer to related work section for a brief summary of StereoTrust.

Note that we only compare our proposal with other trust models when synthetic data is used, since we lack the essential information and the ground-truth for Allegro or eBay to apply other approaches.

Each experiment is repeated 10 times, and the error bars indicate the observed deviation. The metrics for performance evaluation include:

- *false positive rate*
  The transaction is unsuccessful but the algorithm predicted that it would be successful.
- *false negative rate*
  The transaction is successful but the algorithm predicted that it would be risky.
- *overall falseness*
  Sum of false positive rate and false negative rate.

[4]We also conducted experiments using other numbers of features and obtained the similar qualitative trends.



(a) Performance of our approach with varying number of buyers.

(b) Numbers of individual buyers' past transaction.

FIGURE 2. Experiments using Allegro dataset.

6.2. Results

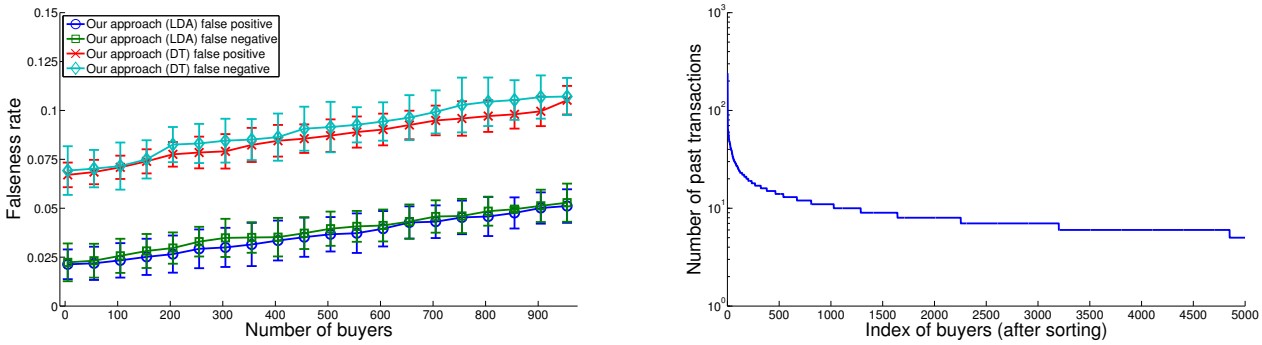
6.2.1. *Real dataset.* We first use Allegro dataset. We rank the 10,000 buyers according to the volumes of their past transactions, i.e., the first buyer has the most past transactions and the last one has the least transactions. We select subset U_b of these buyers starting from the first one. Each buyer evaluates 100 randomly selected transactions (50% are successful and 50% are unsuccessful). We vary the size of U_b to investigate the effect of local knowledge volume.

Fig. 2(a) demonstrates the performance of our approach (LDA and DT based), i.e., how the average rates of false positive and false negative evolve when U_b varies from 5 to 500. As expected, all falseness rates increase when U_b grows. This shows the impact of local knowledge on the proposed algorithms: when U_b is small, it contains only experienced agents, which all have enough past transactions to allow machine learning algorithms to issue accurate predictions. As U_b grows, it contains more inexperienced agents, for which the predictions are less accurate due to less training data. We also notice that DT based algorithm is less accurate than LDA based algorithm. This is because decision tree is more suitable for incorporating qualitative features but the features identified in auction data are quantitative (i.e., it is not easy to obtain the appropriate thresholds to efficiently split the training data.).

Fig. 2(b) shows the distribution of the numbers of individual buyers' past transactions (only first 3000 buyers are shown). Note the logarithmic scale for y-axis: the number of past transactions is quickly decreasing. For instance, less than 100 agents have more than 20 past transactions. Estimating the minimal number of transactions that allow our approach to be precise is challenging, since not all transactions have the same importance (e.g., two very different transactions will be much more useful for our approach than two similar transactions[5]). However, in this set of experiments, we estimate empirically that when the number of transactions is larger than 6, the potential transaction can be relatively reliably predicted (i.e., the overall falseness rate is smaller than 0.1 for LDA based algorithm and 0.19 for DT based algorithm). Experimental results thus show that our approach is capable of "learning" quickly, i.e., with limited amount of input data.

Using eBay dataset (see Fig. 3) we observe the similar trends as those observed using Allegro dataset. The empirically estimated minimum number of past transactions is 10.

[5]If two transactions have similar characteristics, e.g., they have the same category (with the similar feature values) we call them similar transactions. The diversity of transactions naturally aids the machine learning algorithms.



(a) Performance of our approach with varying number of buyers.

(b) Numbers of individual buyers' past transaction.

FIGURE 3. Experiments using eBay dataset.

The results not only help improve the confidence in our approach, but also demonstrate the practicability of our approach across different Internet auction systems.

6.2.2. *Synthetic dataset.* We compare the performance of our approach with other trust models (Fig. 4, 5, 6 and 7). We observe that when P_m (i.e., the fraction of malicious nodes) is low, feedback aggregation model performs quite well. This is because when most nodes in the system are honest, the aggregated feedback can accurately predict the behavior of the target node thus keeping the rates of false positive, false negative as well as overall falseness low. However, with the increase of P_m, various falseness increases sharply (although false feedback filtering strategy is applied). So the accuracy of feedback aggregation model heavily depends on the fraction of malicious nodes and thus is not robust in hostile environments.

One interesting issue about feedback aggregation model is that the false negative rate decreases when P_m is larger than 0.5 (Fig. 4(b), 5(b), 6(b) and 7(b)). A possible reason is that when P_m is high, the amount of false feedback increases, thus increasing the falseness possibility. On the other hand, the total amount of successful transactions decreases in general, so overall the false negative rate decreases.

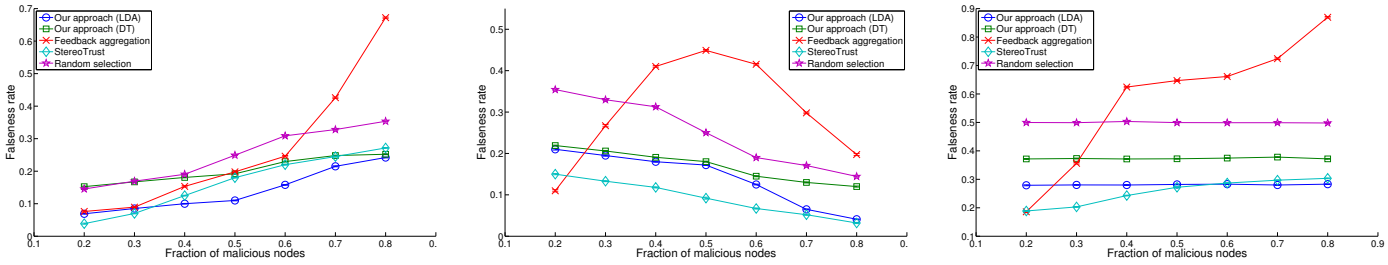
Random selection model, as expected, provides a stable rate of overall falseness of 0.5 because the trustor accepts or rejects a transaction with the same probability. This model is the benchmark of the experiments, demonstrating the falseness rate evolution in the scenario where no trust mechanism is applied.

StereoTrust model outperforms feedback aggregation in general, especially when P_m is high. This is because even though StereoTrust also relies on a few feedback to predict the behavior of the target node, it only requests nodes that are honest (from the trustor's perspective), therefore the probability of receiving false feedback is relatively low.

We observe that DT based approach is less accurate than LDA based approach in all scenarios. This result conforms to the result when real data is used (Fig. 2(a)) and further proves that LDA based algorithm is more suitable to the scenarios where features are quantitative.

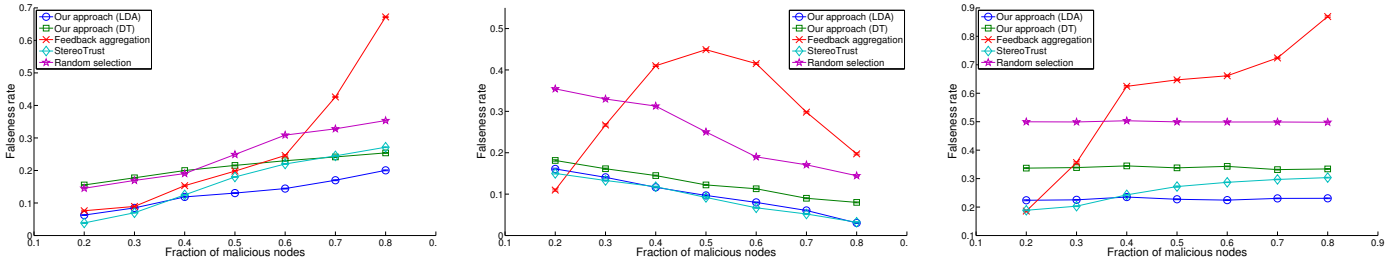
So to sum up, LDA based approach, which has the lowest falseness rates, outperforms all other models (except for the scenarios when θ is quite small, i.e., $\theta \leqslant 0.4$). When θ is small, both LDA based approach and DT based approach produce low accuracy of classification. Finding features that can distinguish successful transactions from unsuccessful ones is thus the key to our approach. We also notice that the performance of our approaches is quite stable (not affected by P_m evidently). This is because our approach only relies on the trustor's local knowledge, which is the most reliable information source.

Note that since our approach relies on different kinds of information, it may not be fair to



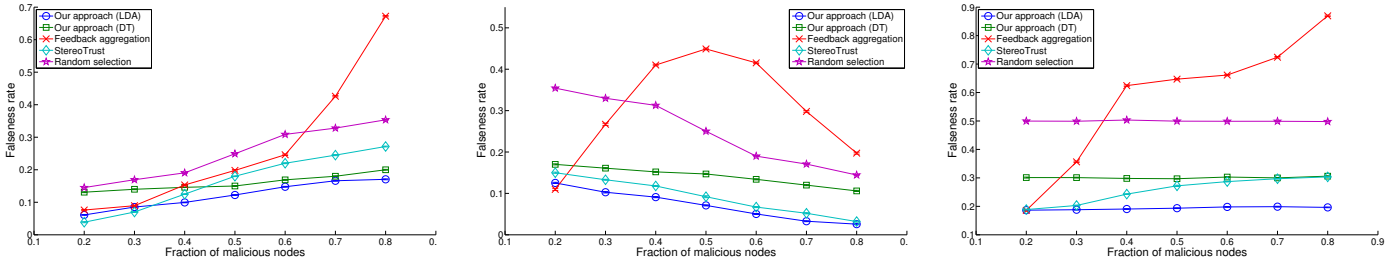
(a) False positive. (b) False negative. (c) Overall falseness.

FIGURE 4. Comparison of our approach with other approaches ($\theta = 0.2$).



(a) False positive. (b) False negative. (c) Overall falseness.

FIGURE 5. Comparison of our approach with other approaches ($\theta = 0.4$).



(a) False positive. (b) False negative. (c) Overall falseness.

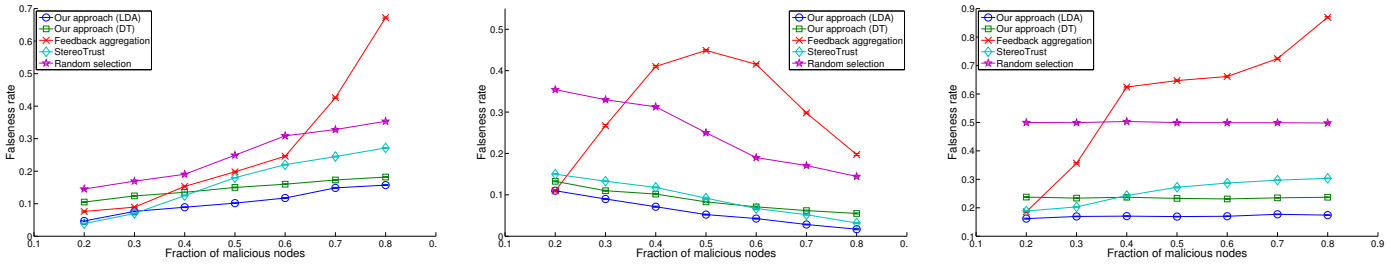
FIGURE 6. Comparison of our approach with other approaches ($\theta = 0.6$).

compare with other models that only use one kind of information (i.e., the target agent's past behavior). Moreover, in some application scenarios, the information used by our approach (e.g., features) may not be always available, rendering the approach unusable. Nevertheless, whenever such information is present, our approach indeed produces reasonable results thus demonstrating its efficacy.

7. RELATED WORK

7.1. Traditional trust mechanisms

Most existing reputation based trust management mechanisms derive an agent's trustworthiness based on its past behavior (e.g., Abdul-Rahman and Hailes (1997); Mui and Mohtashemi (2002); Kamvar *et al.* (2003); Aringhieri *et al.* (2006); Zhou *et al.* (2008); Vogiatzis



(a) False positive. (b) False negative. (c) Overall falseness.

FIGURE 7. Comparison of our approach with other approaches ($\theta = 0.8$).

et al. (2010); Tang *et al.* (2010)). If the trustor has sufficient direct experience, the target agent's future behavior can be reliably predicted (Mui and Mohtashemi (2002)). However, in large-scale distributed settings, direct experience is often unavailable. In this case, the trustor may resort to "indirect experience" – opinions about the target agent obtained from other agents (Abdul-Rahman and Hailes (1997); Jøsang and Ismail (2002); Kamvar *et al.* (2003); Xiong and Liu (2004); Vogiatzis *et al.* (2010)). Simple aggregation (like a seller's reputation score on eBay) relies on the global information like the history of the target agent's behavior. Alternatively, transitive trust models (Abdul-Rahman and Hailes (1997); Jøsang *et al.* (2003)) form chains of trust relationships among agents. However, transitive trust is not always realistic in the real deployment: (i) This method does not handle wrong recommendations properly, which evidently affect the accuracy of the derived trust. (ii) This method does not provide a mechanism for updating trust efficiently in a dynamic system. (iii) Establishing a trust path, even if such a path exists, is nontrivial in a large-scale distributed system.

EigenTrust (Kamvar *et al.* (2003)) is a reputation system developed for P2P networks. EigenTrust uses the transitivity of trust and aggregates trust from peers by performing a distributed calculation to determine the eigenvector of a "trust matrix" over peers. It relies on some pre-trusted peers, which are supposed to be trusted by all peers. EigenTrust (and some other reputation systems like Xiong and Liu (2004); Zhou and Hwang (2007)) is designed based on Distributed Hash Tables (DHT) (Stoica *et al.* (2001); Rowstron and Druschel (2001); Aberer *et al.* (2003)) thus imposing system design complexity and drawbacks, deployment and message overheads (Freedman *et al.* (2005)). Ravichandran and Yoon (2006) proposed a trust system that is built on top of a peer group infrastructure using an EigenTrust-like calculation approach. The groups are formed based on particular interest criterions and the members must follow the set of rules of the group they belong to. The authors assumed that a group leader creates the group and controls the membership. To calculate trust, the authors introduced Eigen Group Trust, which is an aggregative version of EigenTrust. All transactions are coordinated by the group leaders, who are assumed to be trustworthy and resourceful.

GossipTrust (Zhou *et al.* (2008)) is another reputation management system designed for P2P networks. More specifically, it is designed for unstructured P2P networks. The basic idea of GossipTrust is to aggregate global reputation scores using a gossip based algorithm. The gossiping process stops when a gossiping error threshold is reached. Such an approach has the advantage of fast dissemination of global scores with time complexity of $O(log_2(n))$, where n is the number of peers in the network.

The Beta Reputation System (BRS) (Jøsang and Ismail (2002)) is a probabilistic trust model, which is based on beta distribution. In BRS, after each interaction, the service requester gives a binary rating (positive or negative) to indicate the performance of the service

provider. The ratings are then aggregated to estimate the shape parameters that determine the reputation of the assessed agent. However, BRS does not show how it is able to cope with false feedback, which may influence the accuracy of the prediction seriously.

TRAVOS (Teacy *et al.* (2006)) is a trust and reputation model for agent-based virtual organizations. This work also makes use of beta distribution to compute trust but it pays more attention to the issue of unfair feedback. When a buyer evaluates a potential interaction partner (i.e., seller), it first uses its own local knowledge to derive the trustworthiness of the seller. It then estimates the accuracy of the direct trust, which is the probability that the real likelihood of cheating falls in a certain range from the buyer's estimation. If the direct trust is accurate enough, the buyer simply relies on this direct experience based trust. Otherwise, it requests feedback from other buyers who have interacted with the potential seller. To ensure that only accurate feedback is considered, TRAVOS addresses the inaccurate reputation feedback by performing two tasks: (1) Estimating the probability that a feedback provider's opinion of the seller is accurate. This is done by comparing the current feedback with the previous feedback provided by the same buyer. The accuracy of the current feedback is the expected value of beta probability density function representing the numbers of successful and unsuccessful interactions between the buyer and the seller when the buyer is guided by the previous feedback. (2) Adjusting the weight of the feedback according to its accuracy to reduce the effect of inaccurate feedback.

7.2. Bootstrapping trust

Different from the works mentioned above, StereoTrust (Liu *et al.* (2009)) uses another kind of information, i.e., stereotypes to estimate the initial trust of an unknown agent. A stereotype is determined by a feature vector, which can be taken from the profile of the agent. To build stereotypes, the agents that the trustor has interacted with are classified into groups according to the identified features. The stereotypes on each group are calculated by aggregating the trustor's past experience with the members of that group. Then, when facing the target agent, the trustor estimates its trust using stereotypes on the groups to which the target agent belongs. Additionally, when some information about the target agent's past behavior is available, an enhanced StereoTrust model (called d-StereoTrust) uses it to refine the stereotype matching. StereoTrust simply aggregates stereotypes by assigning intuitive weights to derive the trustworthiness of the target agent, thus is not able to tell which stereotypes are more important than other ones. Similar to StereoTrust, Burnett *et al.* (2010) proposed to bootstrap trust of an unknown agent through stereotypes, which are estimated based on M5 model tree learning algorithm (Quinlan (1992)). This work also discussed how to combine initial trust (i.e., based on stereotypes) and reputation based trust using subjective logic (Jøsang (1997)). In contrast, our approach focuses on transactions instead of individual agents. As opposed to the trust models based on stereotypes, this work emphasizes the useful feature selection by performing machine learning algorithms, thus automating the process of determining which features are more useful than others.

TRULLO (Quercia *et al.* (2007)) first gathers the trustor's past ratings (local knowledge) and the associated contextual information to form a rating-context matrix. Then the algorithm applies singular value decomposition (SVD) to decompose the matrix into three matrices: (1) a user-feature matrix that represents the trustor's trust in a certain agent across a set of latent features, (2) a feature-feature matrix (diagonal) that contains each latent feature's contribution to trust estimation in descending order, (3) a feature-context matrix that represents the relevance of each latent feature across the contexts. In order to estimate the initial trust in an unknown agent in a certain context, the trustor combines its trust in that agent across all latent features (the first matrix), where the weight of each feature is assigned according to its influence on assessing trust in general (the second matrix) and its relevance

to the corresponding context (the third matrix). As with most trust models, the issues of cold-start and data sparsity (i.e., the trustor may not know many other agents at all) may degrade the performance of the algorithm.

8. CONCLUSION

This work proposes a machine learning based trust framework, which is designed for large-scale, open systems. It is based on the classification of a potential transaction by learning from a set of past relevant transactions. In this work we instantiate the framework using two common machine learning algorithms, i.e., linear discriminant analysis and decision tree. Unlike many existing trust mechanisms, which rely on a specific agent's past behavior to predict its future behavior, this work uses the trustor's local knowledge and some meta-information about the target, instead of its behavioral history. This makes the approach quite suitable for large-scale distributed environments where the information about the past behavior of the agent in question may be scarce or unavailable, and the third party information may not be reliable or expensive to obtain.

The proposed mechanism has inherent ability to further quantify the trustworthiness of the recommendation itself. Such ability to reason about the recommendation provides one further layer of guidance to an end-user of a decision support system on whether s/he has sufficient and right kind of information locally to derive meaningful recommendation using our approach.

Not surprisingly, experimental results show that the performance of our approach is positively correlated with the discrimination power of the features on successful and unsuccessful interactions. Compared to other trust models, our approach is quite efficient, especially when the third party information is not reliable. Moreover, the performance of the approach is quite stable because it only relies on the trustor's local knowledge. The real Allegro and eBay datasets based experiments show that the proposed approach can be applied to real application scenario (e.g., detecting Internet auction frauds).

An interesting future direction will be to investigate more sophistical machine learning algorithms (e.g., Multiple Discriminant Analysis, etc.) to further improve the accuracy of the approach. Another direction to explore is the mechanisms to bootstrap the system when locally available information is also sparse.

9. ACKNOWLEDGEMENT

The authors are grateful to Dr. Adam Wierzbicki of Polish Japanese Institute of Information Technology for his assistance in providing the Allegro dataset, and the anonymous reviewers for the many helpful and constructive comments. This work was supported by A*Star TSRP grant 102 158 0038 for the pCloud project. Xin Liu contributed to this work while he was a Phd student at NTU Singapore.

REFERENCES

Abdul-Rahman, A. and Hailes, S. (1997). A distributed trust model. In *NSPW '97: Proceedings of the 1997 workshop on New security paradigms*.

Aberer, K., Cudré-Mauroux, P., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., and Schmidt, R. (2003). P-grid: a self-organizing structured p2p system. *SIGMOD Rec.*, **32**(3), 29–33.

Aringhieri, R., Damiani, E., De, S., Vimercati, C. D., Paraboschi, S., and Samarati, P. (2006). Fuzzy techniques for trust and reputation management in anonymous peer-to-peer systems. *Journal of the American Society for Information Science and Technology*, **57**, 528–537.

Burnett, C., Norman, T. J., and Sycara, K. (2010). Bootstrapping trust evaluations through stereotypes. In *Proceedings of 9th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Freedman, M. J., Lakshminarayanan, K., Rhea, S., and Stoica, I. (2005). Non-transitive connectivity and dhts. In *WORLDS'05: Proceedings of the 2nd conference on Real, Large Distributed Systems*.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*. Academic Press Professional, Inc.

Gambetta, D. (1988). Can we trust trust? In D. Gambetta, editor, *Trust: Making and Breaking Cooperative Relations*. Basil Blackwell.

Ghani, R. and Simmons, H. (2004). Predicting the end-price of online auctions. In *In the Proceedings of the International Workshop on Data Mining and Adaptive Modelling Methods for Economics and Management*.

Hyafil, L. and Rivest, R. L. (1976). Constructing Optimal Binary Decision Trees is NP-complete. *Information Processing Letters*, **5**(1), 15–17.

Jøsang, A. (1997). Artificial reasoning with subjective logic. In *Proceedings of the Second Australian Workshop on Commonsense Reasoning*.

Jøsang, A. and Ismail, R. (2002). The beta reputation system. In *Proceedings of the 15th Bled Conference on Electronic Commerce*.

Jøsang, A., Gray, E., and Kinateder, M. (2003). Analysing topologies of transitive trust. In *Proceedings of the Workshop of Formal Aspects of Security and Trust (FAST)*.

Jøsang, A., Ismail, R., and Boyd, C. (2007). A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, **43**, 618–644.

Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H. (2003). The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*.

Liu, X. and Datta, A. (2012). Contextual trust aided enhancement of data availability in peer-to-peer backup storage systems. *Journal of Network and Systems Management*, **20**, 200–225.

Liu, X., Datta, A., Rzadca, K., and Lim, E.-P. (2009). Stereotrust: a group based personalized trust model. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*.

MacKay, D. J. C. (2003). *Information theory, inference, and learning algorithms*. Cambridge University Press.

McLachlan, G. J. (2004). *Discriminant Analysis and Statistical Pattern Recognition*. Wiley-Interscience.

Mitchell, T. (1997). *Machine Learning*. McGraw Hill.

Mui, L. and Mohtashemi, M. (2002). A computational model of trust and reputation. In *Proceedings of the 35th HICSS*.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, **22**(10), 1345–1359.

Quercia, D., Hailes, S., and Capra, L. (2007). Trullo - local trust bootstrapping for ubiquitous devices. In *Proceedings of the 2007 Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking&Services (MobiQuitous)*.

Quinlan, J. R. (1986). Induction of decision trees. *Mach. Learn.*, **1**(1), 81–106.

Quinlan, J. R. (1992). Learning with continuous classes. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, pages 343–348.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Ravichandran, A. and Yoon, J. (2006). Trust management with delegation in grouped peer-to-peer communities. In *SACMAT '06: Proceedings of the eleventh ACM symposium on Access control models and technologies*, pages 71–80, New York, NY, USA. ACM.

Rowstron, A. I. T. and Druschel, P. (2001). Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*.

Tang, J., Seuken, S., and Parkes, D. C. (2010). Hybrid transitive trust mechanisms. In *Proceedings of 9th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Teacy, W. T. L., Patel, J., Jennings, N. R., and Luck, M. (2006). Travos: Trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems*, **12**, 183–198.

Vogiatzis, G., MacGillivray, I., and Chli, M. (2010). A probabilistic model for trust and reputation. In *Proceedings of 9th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Xiong, L. and Liu, L. (2004). Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions on Knowledge and Data Engineering*, **16**, 843–857.

Zhou, R. and Hwang, K. (2007). Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing. *IEEE Trans. Parallel Distrib. Syst.*, **18**, 460–473.

Zhou, R., Hwang, K., and Cai, M. (2008). Gossiptrust for fast reputation aggregation in peer-to-peer networks. *IEEE Transactions on Knowledge and Data Engineering*, **20**, 1282–1295.