D3.4
Trust and reputation in online social learning communities (v2)

<table>
<thead>
<tr>
<th>Core authors</th>
<th>Patricia Gutierrez, Nardine Osman, Carles Sierra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintainers/Editors</td>
<td>Nardine Osman</td>
</tr>
<tr>
<td>PRAISE research group leaders</td>
<td>Mark d’Inverno, François Pachet, Carles Sierra, Luc Steels</td>
</tr>
<tr>
<td>PRAISE project leader</td>
<td>Carles Sierra</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Grant agreement no.</th>
<th>318770</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project acronym</td>
<td>PRAISE - Practice and peRformance Analysis Inspiring Social Educa-</td>
</tr>
<tr>
<td></td>
<td>tion</td>
</tr>
<tr>
<td>Date</td>
<td>July 18, 2014</td>
</tr>
<tr>
<td>Distribution</td>
<td>Public</td>
</tr>
</tbody>
</table>
Disclaimer

The information in this document is subject to change without notice. Company or product names mentioned in this document may be trademarks or registered trademarks of their respective companies.

This document is part of a research project funded by the European Community as project number 318770 acronym PRAISE under THEME 3: FP7-ICT-2011-8 Collaborative project.
Executive Summary

Peers in online social learning communities — whether humans, autonomous agents, or web services — can provide feedback when observing the activities performed by others. This feedback may vary in its form. In this paper, we focus on two types of feedback: (1) advice, which is specified as plans of actions that are proposed by advisers intending to help peers improve their performance; and (2) opinions, which assess the quality of observed activities from the point of view of the opinion holder. An example of an opinion would be a statement such as: I do not think you are playing ‘My Funny Valentine’ on time. An example of an advise would be a statement such as: To get better at playing ‘My Funny Valentine’ on time, you will have to practice it twice a week over a period of four months. This paper visits these two forms of feedback and proposes trust models that help assess each of them.

The first section of this paper illustrates how the trustworthiness of advise and advisers may be assessed. ‘If you go to Ferran Adria’s restaurant you will have the time of your life!’ ‘If you study everyday for two hours you will get very good marks next semester.’ These are examples of advice. We say an advice has two components: a plan to perform and a goal to achieve. In dynamic logic, an advice could be formalised as: \([P_\eta]G\). That is, if \(\eta\) performs plan \(P\), then goal \(G\) will necessarily be achieved. An adviser is an entity which provides such advice. An adviser may be an agent, a planner, or a complex recommender system. The first part of the paper proposes a novel trust model for assessing the trustworthiness of advice and advisers by calculating the expectation of an advice’s outcome. This expectation is based on assessing the probabilities of the advised plan being picked up and performed, and the goal being achieved. These probabilities are learned from an analysis of similar past experiences using tools such as semantic matching and action empowerment.

The second section of this paper illustrates how the trustworthiness of opinions may be assessed, and how such an assessment may be used to produce a collaborative assessment of student’s performances that is based on students’ opinions. We build an automated assessment service for online learning support in the context of communities of learners. The ultimate objective is to introduce automatic tools to support the task of assessing massive number of students as needed in Massive Open Online Courses (MOOC). The final assessments are a combination of tutor’s assessment and peer assessment. We build a trust graph over the referees and use it to compute weights for the assessments aggregations. The model proposed intends to be a support for intelligent online learning applications that encourage student’s interactions within communities of learners and benefits from their feedback to build trust measures and provide automatic marks.
## Contents

### I Trustworthy Advice

1 Introduction ................................................................. 1

2 The CONSUASOR Model .................................................. 2

   2.1 Preliminaries .......................................................... 2

      2.1.1 Experiences ...................................................... 3

      2.1.2 Similarity Measures ........................................... 4

      2.1.3 Probability Distribution Update ............................ 8

      2.1.4 Decay of Information ......................................... 9

   2.2 Probability of an Advice Realising its Goal ......................... 10

      2.2.1 Assumptions .................................................... 12

      2.2.2 Notation ........................................................ 13

      2.2.3 Initialisation ................................................... 14

      2.2.4 Update of the probability distributions on compliance, honour, and goal 
           realisation ....................................................... 14

   2.3 Trust Computation ...................................................... 18

   2.4 Trust Algorithm ......................................................... 19

3 Evaluation ................................................................. 22

   3.1 Experimental platform ............................................... 22

   3.2 Benchmarks ........................................................... 24

   3.3 Experiment 1. Trust on Recommender giving Advice .................. 25

   3.4 Experiment 2. Trust on Recommenders for Goals ....................... 27

   3.5 Experiment 3. Comparison of CONSUASOR strategies .................. 29

4 Demonstration .............................................................. 30

5 Background ............................................................... 33

6 Conclusions and Future Work ............................................ 35

II Collaborative Assessment ................................................ 37

7 Introduction ............................................................... 37

8 Collaborative Assessment ................................................ 38

   8.1 Notation and preliminaries ........................................ 38

   8.2 Computing collaborative assessments ............................... 40

9 Experimental Platform and Evaluation .................................... 43

   9.1 Social Network Generation .......................................... 43

      9.1.1 Random Networks ............................................. 43

      9.1.2 Power Law Networks ........................................... 44
Part I

Trustworthy Advice

‘If you go to Ferran Adria’s restaurant you will have the time of your life!’ ‘If you study everyday for two hours you will get very good marks next semester.’ These are examples of advice. We say an advice has two components: a plan to perform and a goal to achieve. In dynamic logic, an advice could be formalised as: $[P_\eta]G$. That is, if $\eta$ performs plan $P$, then goal $G$ will necessarily be achieved. An adviser is an entity which provides such advice. An adviser may be an agent, a planner, or a complex recommender system. This part of the paper proposes a novel trust model for assessing the trustworthiness of advice and advisers. It calculates the expectation of an advice’s outcome by assessing the probabilities of the advised plan being picked up and performed, and the goal being achieved. These probabilities are learned from an analysis of similar past experiences using tools such as semantic matching and action empowerment.

1 Introduction

Advice is what one relies when making decisions on future actions. As such, advice is crucial in directing actions and interactions. Advice may be provided by a physician to a patient on what they can do to lead a healthier life. It could be provided by a tutor, suggesting the best exercises to solve in order to pass an exam. It could be provided by a personal assistant agent, suggesting an itinerary for a fabulous vacation. It could be provided by recommender systems, suggesting what would be the best movie one can rent.

But how can one choose which advice to follow and which advice to discard? This part of the paper proposes a computational trust model, CONSUASOR, that assesses the trustworthiness of advice and their advisers. We say an advice has two components: a plan to perform and the goal intended to be achieved. In dynamic logic, this may be formalised as $[P_\eta]G$, where $P$ is the recommended plan to $\eta$ in order to fulfil goal $G$. That is, if $\eta$ performs plan $P$, then goal $G$ will necessarily be achieved. We note that the adviser may be a human, an agent, or a recommender system. The proposed model is based on the concept that an adviser is a good adviser if it is knowledgable about three main issues: (1) compliance, which describes how much compliant is the person being advised with following recommendations; (2) honour, which describes how much honourable is the person being advised in performing a recommended plan that he has accepted, and (3) goal realisation, which describes whether the recommended plan actually causes the goal to be fulfilled. Compliance is important, because good advisers are those that are knowledgable about who is willing to accept what advice, and personalising their plans accordingly. Honour is also important, because knowledge about whether the one being advised actually performs the recommended plan is fundamental. Finally, goal realisation is crucial, since a good adviser should be an adviser whose recommended plans can actually fulfil the intended goals.

The proposed model then computes the trustworthiness of advice and advisers based on predicting the outcome of advice. The model calculates the expectation of an advice’s outcome by assessing the probabilities of the advised plan being picked up and performed, and the goal being realised. These probabilities are learned from an analysis of similar past experiences using tools
such as semantic matching and action empowerment.

The remainder of this part of the paper is divided as follows. Section 2 presents our proposed trust model, CONSUASOR. Section 3 presents our experimental platform, benchmarks and evaluation. Section 4 describes a demonstration of our proposal, where we have built an online website that allows users to give advice and provide feedback on given advice. Finally, Section 5 provides a brief comparison to related work, before concluding with Section 6.

2 The CONSUASOR Model

The question that this section addresses is: How much should one trust a recommender’s advice? In more precise terms, how much should \( \Gamma \) trust \( \rho \) when \( \rho \) recommends a plan \( [P_\eta | G] \)? We get inspiration from previous work [18], where trust was based on the expectation of a particular observation given a commitment, which was specified as a conditional probability:

\[
p(\text{Observe}(\Gamma, \phi^I) \mid \text{Commit}(\rho, \phi))
\]

where the term \( \text{Commit} \) had two arguments — the one making the commitment (\( \rho \)) and the action he was committing to (\( \phi \)) — and the term \( \text{Observe} \) had two arguments — the one observing the outcome of the commitment (\( \Gamma \)) and the outcome of the commitment describing what \( \rho \) actually performed (\( \phi^I \)). The idea was that past commitments helped in assessing the expected outcome of similar current commitments. For example, if a seller has always delivered good quality goods, then one may expect the seller’s next delivered goods to be of good quality as well.

In this section, we adopt the basic idea that a trust measure is based on the expectation of observing the possible outcomes of a commitment. When assessing the trustworthiness of advice, this expectation is specified as a conditional probability of observing an advice \( [P_\eta | G] \) realising its goal:

\[
p(\text{Observe}(\Gamma, [P_\eta | G]) \mid \text{Commit}(\rho, [P_\eta | G]))
\]

where \( P \) is the plan recommended by \( \rho \) for \( \eta \) in order to fulfil goal \( G \), and \( \Gamma \) represents the party that observes the realisation of the advice \( [P_\eta | G] \)’s goal.

The remainder of this section is divided as follows. Section 2.1 presents the preliminaries needed for understanding the proposed model, Section 2.2 presents how the probability \( p(\text{Observe}(\Gamma, [P_\eta | G]) \mid \text{Commit}(\rho, [P_\eta | G])) \) may be computed by relying on past similar experiences, Section 2.3 illustrates how the probability distribution is used to compute a final trust measure, and Section 2.4 closes this section with a trust algorithm that provides one example of a concrete implementation of our proposed model.

2.1 Preliminaries

CONSUASOR is an experienced-based trust model that relies on past experiences to predict future outcomes. As such, calculating the similarity between experiences is crucial. In this section, we present the preliminaries of our proposed model by defining experiences (Section 2.1.1), similarity measures (Sections 2.1.2), and the update of probabilities and probability distributions (Section 2.1.3) in the light of new experiences. Additionally, this section also presents the general
concept of information decay (Section 2.1.4), which is a basic notion that underlies our work, as it describes how information loses its value over time.

2.1.1 Experiences

A Single Experience The advice that we are interested in assessing are conditional statements of the form: ‘if the recommended plan is performed, then the intended goal will be realised’. As such, past experiences should not only keep track of advice and their realised goals, but of the fulfilment of the conditional part of the advice as well. This is because the adviser might have good advise, but the one being advised has not been fulfilling its duties in carrying out the recommended plans. As such, an experience should keep note of several issues:

- **The advice.** We interpret an advice as a commitment made by the adviser $\rho$ that the goal $G$ will be realised if $\eta$ performs plan $P$. An advice is specified as $Commit(\rho, [P_\eta]G)_t$, where $t$ specifies the time at which the advice $[P_\eta]G$ was recommended by $\rho$.

- **The accepted plan.** We interpret $\eta$ accepting an advice with a plan $P$ as a commitment made by $\eta$ to actually perform $P$. This is specified as $Commit(\eta, P')_{t'}$, where $t'$ describes the time at which $\eta$ accepts plan $P'$.

- **The performed plan.** When $\eta$ actually performs a plan $P''$, some entity $\beta$ needs to observe (or verify) this performance, which is specified as $Observe(\beta, P''_\eta)_{t''}$, where $t''$ describes the time at which $\beta$ observed $P''_\eta$, or $\eta$’s performed plan.

- **The realised goal.** The realised goal needs to be observed by some entity $\alpha$, and this observation is specified as $Observe(\alpha, G')_{t'''}$, where $t'''$ describes the time at which $\alpha$ observed the goal $G'$ being realised.

A single experience $\mu$ is then recorded as follows:

$$\mu = \langle Commit(\rho, [P_\eta]G)_t, Commit(\eta, P')_{t'}, Observe(\beta, P''_\eta)_{t''}, Observe(\alpha, G')_{t'''} \rangle_{t < t' < t'' < t'''}$$

Note that $\eta$ may commit to a variation of the plan: $P' \neq P$. For example, assume an advice stating that one should “practice his piano twice a day”, the one being advised may decide to commit to a variation of this plan, say “practicing his piano once a day”. We also say that what may be observed may also be a variation of what has been committed to: $P'' \neq P'$ and $G' \neq G$.

In the general case, observers will be different from each other, and different from the adviser and the one being advised. Although it is possible to have particular cases where $\beta = \alpha$, or $\rho = \beta$, or $\eta = \alpha$, and so on.

Each element of the experience should have a different time-stamp. An integrity constraint is then needed to check that a plan is accepted (by committing to it) after it has been recommended by an adviser, and that the plan has been performed (and observed) after it has been accepted, and that the goal has been realised (and observed) after the plan has been performed. This integrity constraint is specified by the condition $t < t' < t'' < t'''$. 

318770 July 18, 2014 3
History of Experiences  Numerous and different history of experiences may exist, and we use the notation \( H_\alpha = \{ \mu, \mu', \ldots \} \) to describe \( \alpha \)'s history of experiences.

Populating the history of experiences needs to address numerous issues. For instance, how is information collected? In other words, when an adviser makes an advice or when one accepts an advice, how is this information recorded? Also, who is trusted to observe a plan being performed or a goal being realised, and how are such observations carried out and recorded? How is the relation between elements recognised? For example, recognising that observing goal \( G' \) being realised is the result of plan \( P'' \) being performed, or that observing plan \( P'' \) being performed is the result of \( \rho \) honouring its commitment to \( P' \), or that committing to \( P' \) is the result of \( \rho \)'s compliance with the advice \([P_\rho]G\).

In this part of the paper, we do not dwell much on how a history of experiences is populated, as this could be context dependent. For instance, the entity maintaining a given history of experiences, whether this history is centralised or not, will need to specify who is trusted to record elements of an experience, and how are these elements recorded. The entity maintaining the history of experiences may also specify how experiences may be shared, and how reliable are shared experiences. As an example, consider an online classroom where the history is maintained by the online system. The tutor may advise its student to “focus on hand posture when playing the piano to improve performance”. It is then the student’s duty to confirm its willingness to follow its tutor’s advice. For example, an “okay” from the student may signal its commitment to the advice. The student should then confirm whether they performed the recommended plan or not. For example, the student may say “I had troubles focusing on hand posture”. The teacher may then assess the student’s performance by listening and marking their uploaded performance. These marks may provide an indication on whether the goal of “improving performance” has been realised or not. As such, different signals may be considered for recording experiences in different scenarios, and each system will need to define its signals.

2.1.2 Similarity Measures

When assessing the level of similarity between a past experience and a current one, we need to take into consideration a number of similarity measures, such as the similarity of plans, or the similarity of goals, which we define next.

Plan and Goal Similarity  We assume there is a set of actions \( \mathcal{A} \) that form the taxonomy of actions \( T_\mathcal{A} \). Plans are sets of actions and the set of all possible plans is \( \mathcal{P} = 2^\mathcal{A} \). We assume that there is a semantic similarity relationship between actions \( S: \mathcal{A} \times \mathcal{A} \rightarrow [0, 1] \) that shows the degree of relationship between actions. We also assume there is a set of propositional terms \( \mathcal{T} \) that form the taxonomy of propositional terms \( T_\mathcal{T} \). Goals are sets of propositional terms and the set of all possible goals is \( \mathcal{G} = 2^\mathcal{T} \). We assume that there is a semantic similarity relationship between propositional terms describing goals (overloading symbol \( S \)) \( S: \mathcal{T} \times \mathcal{T} \rightarrow [0, 1] \) that shows the

\[1\] A plan is usually understood as a temporal set of actions, describing the detailed steps (along with their conditions) needed for achieving a given goal. In this part of the paper, we simplify the notion of a plan by reducing it to a set of actions. The only impact of this simplification is keeping the definition of plan similarity relatively straightforward. Adopting the definition of a plan as a temporal set of actions and redefining plan similarity accordingly is left for future work.
degree of relationship between goals.

Plan similarity and goal similarity are then computed in the same manner accordingly:

\[
 Sim(Q, Q') = \frac{1}{2} \left( \min_{\phi \in Q} \max_{\phi' \in Q'} \{ S(\phi, \phi') \} + \min_{\phi' \in Q} \max_{\phi \in Q} \{ S(\phi, \phi') \} \right) \tag{2}
\]

where \( \phi \in Q \) describes either an action of plan \( Q \) (if \( Q \) was a plan), or it describes a propositional term in goal \( Q \) (if \( Q \) was a goal), and \( S \) describes the semantic similarity between actions, or propositional terms.

In other words, Equation 2 states that to calculate the semantic similarity between two plans (goals), we first measure the semantic similarity between each action of the first plan (propositional terms of the first goal) with all the actions of the second plan (propositional terms of the second goal), and only the actions of the second plan (propositional terms of the second goal) that result with maximum similarity are then considered. This provides the maximum similarity measure that each action of the first plan (propositional term of the first goal) can have with the second plan (second goal). To aggregate those maximum similarity measure, we then take the minimum of those similarity measures. This describes the similarity between the first plan (goal) and the second plan. Then, to ensure that the function \( Sim \) is symmetric, we repeat the same process but in reverse order of plans (goals) — that is, we calculate the similarity between the second plan (goal) and the first — and we take the average of the two similarity measures between the two plans (goals). We note that the range of \( Sim \) is \([0, 1]\).

But what is the motivation behind choosing the min operator when calculating \( Sim \)? The basic idea behind this approach is that when considering the similarity of two entities, we need to consider how do the elements composing each entity relate to that entity. In our case, we say the entity (whether a plan or a goal) may be viewed as a set composed of a conjunction of elements. In mathematical terms, as illustrated by Figure 1, there are a number of conjunctive operators that may be used, such as the product operator (\( \prod \)) and the minimum operator (min). We adopt the minimum operator, which describes an optimistic approach. For instance, if we are comparing \( a \land b \) to \( c \) and \( S(a, c) = 0.3 \) and \( S(b, c) = 0.2 \), then we have \( \min \{S(a, c), S(b, c)\} = 0.2 \). For a more pessimistic approach, one can replace the minimum operator (min) with the product operator (\( \prod \)).

In this case, \( \prod \{S(a, c), S(b, c)\} = 0.06 \), which is drastically smaller than considering the minimum. We note that the choice of operator will be domain dependent. Alternative methods for calculating \( Sim \) may be used as long as symmetry is maintained.

Finally, we adopt the following definition of semantic similarity for \( S \) [12]:

\[
 S(\phi, \phi') = e^{-\alpha l} \frac{e^{\kappa_l h} - e^{\kappa_l h}}{e^{\kappa_l h} + e^{\kappa_l h}} \tag{3}
\]
D3.4 Trust and reputation in online social learning communities (v2)

where \( e \) is Euler’s number, \( l \) is the length (i.e. number of hops) of the shortest path between the terms \( \phi \) and \( \phi' \) in a taxonomy, \( h \) is the depth of the deepest concept subsuming both concepts, and \( \kappa_1 \) and \( \kappa_2 \) are parameters scaling the contribution of shortest path length and depth, respectively. Essentially, \( \kappa_1 \) and \( \kappa_2 \) are parameters that \( \alpha \) could use to customise the weight given to \( l \) and \( h \), respectively. The function \( S \) is symmetric (i.e. \( S(\phi, \phi') = S(\phi', \phi) \)), and its range is \([0, 1]\).

The basic idea of semantic similarity is that the concepts within a taxonomy are closer, semantically speaking, depending on how far away are they in the taxonomy’s graph. Equation 3 calculates the semantic similarity between two concepts based on the path length (more distance in the graph means less semantic similarity), and the depth of the subsumed concept (common ancestor) in the shortest path between the two concepts (the deeper in the hierarchy, the closer the meaning of the concepts). We note, however, that we provide Equation 3 just as an example. As such, we refer the interested reader to the work by Li et al. [12] for further details on Equation 3 and we stress that alternative approaches can be used to replace this equation. There is no universal measure for semantic similarity, and it usually depends on the structure of the taxonomy, amongst other things. Different contexts and different taxonomies may require different approaches and equations. Similarly, different systems may also prefer different equations for their own taxonomies.

**Plan Empowerment**   When the capability of performing similar actions is relevant, we say measures of empowerment are needed, as opposed to semantic similarity measurements. For example, driving a truck and driving a car may be similar. However, if \( \alpha \) is capable of driving a truck then it will be capable of driving a car, but not vice versa. As such, when considering the capabilities of performing actions we are not only interested in similar actions, but whether one action empowers another. As illustrated by the truck/car driving example, empowerment measures need not be symmetric. We say, while similarity measures are computed by considering taxonomies (based on the *is-a* relation), empowerment measures are computed by considering meronomies (based on the *empowered-by* relation, and we argue that the *empowered-by* relation is just another form of the *part-of* relation).

To compute the empowerment measure between two nodes of a meronomy, we make use of the OpinioNet algorithm [19]. OpinioNet highlights the importance of the structural relations (based on the *part-of* relation) linking related entities and their use in indicating the flow of opinions from one entity to another. OpinioNet’s mechanism allows a single agent, after it has formed opinions about a few entities (nodes) in a structural graph, to be able to infer its opinion concerning unfamiliar related entities. For example, say a new coffee machine is now out in the market and it has not been rated yet. What can an interested customer infer about this new item’s reputation? Clearly, the reputation of other coffee machines of the same brand, or even other products of this brand in general, could be of help here. Hence, OpinioNet highlights the need for representing the structural relations linking those entities together. A structural graph may then be used, and the brand may be represented as one node in this graph, the brand’s coffee machines as a child node to the former, the new coffee machine model as a child node to the latter, and so on. Such a representation will not only facilitate the flow of opinions amongst related entities, but also permit raters to choose the granularity level at which they would prefer to leave their opinions at. For instance, while one agent might be interested in rating this specific model in the future, it might also be interested in providing a rating for the brand’s coffee machines in general.
Consider, for example, the simple meronomy of actions on music performance presented by Figure 2. The arrows describe the empowered-by relations. For instance, “Practice Scales” may be thought of as empowered by “Practice Piano”. In other words, for one to practice their piano, they should already know how to practice the scales (or if one is capable of practicing the piano, then they are capable of practicing the scales). In general, we say the child node is empowered by the parent node.

![Diagram of music actions meronomy](image)

**Figure 2:** A meronomy describing actions about music practice

In this part of the paper, we interpret the opinions of OpinioNet to describe the capability of performing an action, specified as a node in a meronomy. In other words, if a node in a meronomy receives the best opinion possible (specified as the probability distribution $B$), this is interpreted as the full capability of performing that node, or action. The propagation of an opinion from a node $\phi$ to a node $\phi'$ is then interpreted as deducing what the capability of performing $\phi'$ is, given the capability of performing $\phi$. We then say the difference between the original opinion at $\phi$ (describing capability of performing $\phi$) and the propagated opinion at $\phi'$ (describing the deduced capability of performing $\phi'$, given the capability of performing $\phi$) specifies the empowerment of $\phi$ on $\phi'$. For example, if the full capability of $\phi$ implies a strong capability of $\phi'$ (where the distance between those two capabilities would be very small), then this describes that $\phi$ greatly empowers $\phi'$.

Let us say “Practice Piano” of Figure 2 receives the best opinion possible (describing the full capability of practicing the piano). Propagating this opinion to other nodes in the meronomy, OpinioNet can help us deduce that one is also very much capable of practicing the scales (the distance between the best opinion possible at “Practice Piano” and the propagated opinion to “Practice Scales” is 0.9), but only half as capable when it comes to practicing band improvisation (the distance between the best opinion possible at “Practice Piano” and the propagated opinion to “Practice Band Improvisation” is 0.9).

Formally, the empowerment of $\phi$ on $\phi'$ is defined accordingly:

$$\phi \triangleright \phi' = 1 - |emd(B, opinioNet(M, B, \phi, \phi'))|$$

where $opinioNet$ is a function that returns the propagated opinion at $\phi'$ (describing the capability of performing $\phi'$) by propagating the best opinion possible $B$ from $\phi$ (describing the full capability...
ity of performing $\phi$) in meronomy $M$ following the OpinioNet propagation algorithm \cite{19}; and $emd$ is the earth mover’s distance that calculates the distance (whose range is $[1, 0]$) between two probability distributions $^2$.

We note that as long as a meronomy does not change, empowerment measures remain fixed. Empowerment measures between terms may then be computed in advance (or when the meronomy changes) for every pair of nodes.

Based on Equation 4, we define the empowerment of plan $P$ on plan $P'$ accordingly:

$$Emp(P, P') = \min_{a' \in P'} \{ \max_{a \in P} \{ a \triangleright a' \} \}$$

(5)

Note that we use the ‘min-max’ operators, following the same reasoning as that of Equation 2.

### 2.1.3 Probability Distribution Update

Our proposed trust model is based on the idea that the information provided by past experiences can help us assess the outcome, or expectation, of a current experience. We say, given an experience $\mu$, we need to update the probabilities of expectations according to the information presented by $\mu$. To do so, we first update the probability of at least one single expectation ($X = x$) with respect to $\mu$. Then, we update the probability distribution over all possible expectations in a single step, following the minimum relative entropy approach. We explain these two steps in further detail below.

**Updating the probability of a single expectation $X = x$** We say, if the information provided by experience $\mu$ suggests the increase in the probability of expectation $X = x$, then the amount by which this probability is increased should be dependent on the relevance of the experience $\mu$ in the context of updating the probabilities of expectations. Furthermore, we design this increase in such a way that avoids unstable behaviour: that is, even if an experience is fully relevant, a *single* experience cannot result in considerable change in the probability in question. In other words, considerable changes in expected behaviour can only be the result of information learned from an accumulation of experiences (as opposed to information learned from a single experience). As such, we update the probability of expectation $X = x$ accordingly:

$$p^{t_\mu}(X = x) = p^{t_{\text{last}}}(X = x) + (1 - p^{t_{\text{last}}}(X = x)) \cdot \varepsilon \cdot R_\mu(\mathbb{P}(X))$$

(6)

where $p^{t_{\text{last}}}(X = x)$ specifies the probability of expectation $x$ at time $t_{\text{last}}$ (the time when the probability of $x$ was last updated), $p^{t_\mu}(X = x)$ specifies the probability of expectation $x$ after considering the information provided by $\mu$ at time $t_\mu$ (the time of experience $\mu$), $\varepsilon$ specifies the maximum percentage of increase that a probability is allowed (which controls unstable behaviour), and $R_\mu(\mathbb{P}(X))$ specifies the relevance of the experience $\mu$ in the context of updating the probabilities of expectations ($\mathbb{P}(X)$).

$^2$If probability distributions are viewed as piles of dirt, then the earth mover’s distance measures the minimum cost for transforming one pile into the other. This cost is equivalent to the ‘amount of dirt’ times the distance by which it is moved, or the distance between elements of the probability distribution’s support. The range of $emd$ is $[0, 1]$, where 0 represents the minimum distance and 1 represents the maximum possible distance.
Equation 6 states how to calculate the probability of expectation $X = x$ after considering experience $\mu$, whose relevance in this context is $R_\mu(\mathbb{P}(X))$. The update is based on increasing the latest probability $p^{last}(X = x)$ (or the probability of the expectation $x$ that was calculated at an earlier point in time, $t_{last} < t_\mu$, and that did not consider the experience $\mu$) by a factor $\epsilon \cdot R_\mu(x)$ of the potential increase $(1 - p^{last}(X = x))$. This fraction is defined by a fixed percentage ($\epsilon$) tuned by the relevance of $\mu$ ($R_\mu(\mathbb{P}(X))$). In other words, if $R_\mu(\mathbb{P}(X)) = 1$, or if $\mu$ was maximally relevant, then $p^{last}(X = x)$ is increased by $\epsilon$ percent of $1 - p^{last}(X = x)$. For instance, if the probability of $x$ was 0.6, the maximum percentage of increase is 0.1, and the experience $\mu$ updating the probability of $x$ has a relevance of 1, then the new probability of $x$ becomes $0.6 + (1 - 0.6) \cdot 0.1 \cdot 1 = 0.64$.

We note that $\epsilon \in [0, 1]$ and $R_\mu(\mathbb{P}(X)) \in [0, 1]$. However, the ideal value of $\epsilon$ should be closer to 0 than to 1 so that even if an experience is very relevant, a single experience does not result in considerable changes in expected behaviour.

**Updating the probability distribution over all possible expectations** With the probability of one (or more) expectation(s), we update the probability distribution over all possible expectations in one single and simple step, following the entropy-based approach of Sierra and Debenham [28]. The entropy-based approach updates a distribution $\mathbb{P}^{last}(X)$ (where $t_{last}$ describes the time when the probability has been updated last) into $\mathbb{P}^{\mu}(X)$ (where $t_\mu$ describes the time of the experience $\mu$ resulting in this update) such that: (1) the new distribution satisfies the constraint(s) imposed by the new point(s) (that is, if the probability of expectation $X = x$ was updated to $p^{\mu}(X = x)$, then the new distribution’s value at $x$ should be equivalent to $p^{\mu}(X = x)$), and (2) the new distribution’s relative entropy with respect to $\mathbb{P}^{last}(x)$ is minimal. In other words, we look for distributions that satisfy the updated probabilities of expectations and are at a minimal distance from the original distribution $\mathbb{P}^{last}(X)$ (as the relative entropy is a measure of the difference between two probability distributions). This is described accordingly:

$$
\mathbb{P}^{\mu}(X) = \arg \min_{\mathbb{P}(X)} \sum_{i} p^{last}(X = i) \log \frac{p^{last}(X = i)}{p(X = i)}, \text{ such that } \{p(X = x) = p^{\mu}(X = x), \ldots\} \quad (7)
$$

where $\{p(X = x) = p^{\mu}(X = x), \ldots\}$ specifies that one or more constraints of the form $p(X = x) = p^{\mu}(X = x)$ need to be satisfied.

**2.1.4 Decay of Information**

An important notion in our proposal is the notion of the decay of information. We say the integrity of information decreases with time. In other words, the information provided by a probability distribution should lose its value over time and decay towards a default value. We refer to this default value as the decay limit distribution.

Calculating the decay limit distribution is outside the scope of this paper, although we argue that one may have background knowledge concerning the expected integrity of a precept as $t \to \infty$. Such background knowledge may be expressed in terms of an individual’s knowledge, and is represented as a decay limit distribution $\mathbb{D}_x$, where $x$ describes the specific context.\footnote{For example, when calculating the probability distribution of the expected outcome for the goal of submitting one’s work on time, one might expect the default probability of submitting on time to be very high for computer science conferences, whereas the default probability of submitting on time will be much lower for an internal technical report, for instance.}
In summary, given a distribution, \( P \), and a decay limit distribution \( D \), the distribution \( P \) decays from one point in time \((t')\) to a later point in time \((t, \text{ where } t > t')\) by:

\[
P^{t' \rightarrow t} = \Lambda(D_x, P_{t'})
\]  

(8)

where \( \Lambda \) is the *decay function* satisfying the property: \( \lim_{t \rightarrow \infty} P^{t' \rightarrow t} = D_x \).

One possible definition for \( \Lambda \) could be:

\[
P^{t' \rightarrow t} = \nu \Delta(t, t') \cdot P_{t'} + (1 - \nu \Delta(t, t')) D_x
\]  

(9)

where \( \nu \) is the decay rate, and:

\[
\Delta(t, t') = \begin{cases} 
0, & \text{if } t - t' < \omega \\
1 + \frac{t - t'}{t_{\text{max}}}, & \text{otherwise}
\end{cases}
\]

The definition of \( \Delta(t, t') \) above serves the purpose of establishing a minimum *grace* period, determined by the parameter \( \omega \), during which the information does not decay, and that once reached the information starts decaying. The parameter \( t_{\text{max}} \), which may be defined in terms of multiples of \( \omega \), controls the *pace of decay*. The main idea behind this is that after the grace period, the decay happens very slowly; in other words, \( \Delta(t, t') \) decreases very slowly.

Of course, one might also think of either the decay function or the decay limit distribution to be also a function of time, if the context requires this.

### 2.2 Probability of an Advice Realising its Goal

To help assess the trustworthiness of an advice, CONSUASOR calculates the expectation of the advice’s outcome, specified as the probability of an advice realising its goal:

\[
p(\text{Observe}(\Gamma, [P_\eta]G) \mid \text{Commit}(\rho, [P_\eta]G))
\]

Note that we are interested in the probability of the advice being the responsible of the fulfilment of the goal; that is, we rule out accidental fulfilment’s of the goal. As such, CONSUASOR considers that an advice which recommends plan \( P \) to \( \eta \) fulfils its goal \( G \) if three things happen:

1. **Compliance:** \( \eta \) agrees to perform \( P \), specified as \( \text{Commit}(\eta, P) \). This describes \( \eta \)’s compliance with following \( \rho \)’s recommended advice \([P_\eta]G\).
2. **Honour:** \( \eta \) performs \( P \), specified as \( \text{Observe}(\beta, P_\eta) \). This describes \( \eta \)’s honour in following its own commitment to perform the plan it has agreed to.

\[\text{While CONSUASOR does not dwell on the motivation behind one being compliant with an advice, we note that compliance may be influenced by willingness, capabilities, and/or obligations. Social commitments, for instance, may result in obligations, and hence compliance. One may be obliged to accept their superior’s advice. Personal interest in the intended goal and the trustworthiness of the adviser may result in the willingness to accept the recommended plan. In such a case, the probability of compliance will depend on the trust on the adviser, although the probability of compliance will also be used for computing the trustworthiness of advice, and hence, the adviser. This could result in non-linear equations, which will require alternative computational approaches, such as following fixed point methods. We leave this for future work.}\]

\[\text{While CONSUASOR does not dwell on the motivation behind one honouring its commitments, we note that honour may again be influenced by willingness, capabilities, and/or obligations. For instance, one may be willing to perform a given plan, and hence accept it, only to discover later on that they are in fact incapable of performing the accepted plan. Social commitments may also force one to accept a plan that they are in fact not to willing to perform.}\]
3. **Goal Realisation: Plan \( P \) realises goal \( G \), specified as \( \text{Observe}(\alpha, G) \).**

Accordingly, a good adviser is then one who not only knows the causal relation between plans and goals, but also knows about the advisee’s compliance and honour (that is, it can correctly guess what plan will the advisee agree to and perform). In other words, a good adviser becomes one who modifies his advice, taking into consideration its knowledge about the the advisee’s compliance and honour in order to ensure the intended goal is fulfilled.

Given that we define an advice realising its goal as a conjunction of three events where the advisee agrees to and performs the plan and the performed plan realises the intended goal, the probability of an advice realising its goal becomes:

\[
p(\text{Observe}(\Gamma,[P_\eta]|G) \mid \text{Commit}(\rho,[P_\eta]|G)) = \\
p(\text{Observe}(\alpha,G) \text{ and } \text{Observe}(\beta,P_\eta) \text{ and } \text{Commit}(\eta,P) \mid \text{Commit}(\rho,[P_\eta]|G))
\]

where \( \Gamma = \{ \alpha, \beta \} \). In other words, \( \Gamma \) is the coalition that observes that the advise fulfills its goal. Note that \( \Gamma \)'s observation is deduced from \( \alpha \) and \( \beta \)'s observations. If \( \beta \) observes the plan being performed and \( \alpha \) observes the goal being realised, then we can say that the coalition \( \Gamma \) observes the advice fulfilling its goal. This is similar to coalition logic \(^6\), a special type of modal logic. Properties in coalition logic can describe what a coalition, or a group of agents, may achieve as a whole. In other words, it can describe what the group is capable of. In Equation \(10\) the observation on the left hand side describes what the coalition \( \Gamma = \{ \alpha, \beta \} \) observes, and it is deduced from the individual observations on the right hand side of the equation.

Given Equation \(10\) for the probability of an advice fulfilling its goal, we can derive the following proposition: \(^{10}\)

---

\(^6\)The proof for Proposition \(^1\) which is presented below, is a straightforward proof that makes use of the conditional probability definition (or axiom): \( p(A|B) = \frac{p(A \text{ and } B)}{p(B)} \).

**Proof:**

\[
p(\text{Commit}(\eta,P) \mid \text{Commit}(\rho,[P_\eta]|G)) \cdot \\
p(\text{Observe}(\beta,P_\eta) \mid \text{Commit}(\eta,P) \text{ and } \text{Commit}(\rho,[P_\eta]|G)) \cdot \\
p(\text{Observe}(\alpha,G) \mid \text{Observe}(\beta,P_\eta) \text{ and } \text{Commit}(\eta,P) \text{ and } \text{Commit}(\rho,[P_\eta]|G))
\]

\[
= \frac{p(\text{Commit}(\eta,P) \text{ and } \text{Commit}(\rho,[P_\eta]|G))}{p(\text{Commit}(\rho,[P_\eta]|G))} \cdot \\
\frac{p(\text{Observe}(\beta,P_\eta) \text{ and } \text{Commit}(\eta,P) \text{ and } \text{Commit}(\rho,[P_\eta]|G))}{p(\text{Commit}(\eta,P) \text{ and } \text{Commit}(\rho,[P_\eta]|G))} \cdot \\
\frac{p(\text{Observe}(\alpha,G) \text{ and } \text{Observe}(\beta,P_\eta) \text{ and } \text{Commit}(\eta,P) \text{ and } \text{Commit}(\rho,[P_\eta]|G))}{p(\text{Observe}(\beta,P_\eta) \text{ and } \text{Commit}(\eta,P) \text{ and } \text{Commit}(\rho,[P_\eta]|G))}
\]

\[
= \frac{p(\text{Observe}(\alpha,G) \text{ and } \text{Observe}(\beta,P_\eta) \text{ and } \text{Commit}(\eta,P) \text{ and } \text{Commit}(\rho,[P_\eta]|G))}{p(\text{Commit}(\rho,[P_\eta]|G))}
\]

\(= p(\text{Observe}(\alpha,G) \text{ and } \text{Observe}(\beta,P_\eta) \text{ and } \text{Commit}(\eta,P) \mid \text{Commit}(\rho,[P_\eta]|G)) \)

\(\square\)
Proposition 1 (Probability of an advice realising its goal I) The probability of an advice $P_\eta|G$ realising its goal is the product of the probability of compliance $p(\text{Commit}(\eta,P)|\text{Commit}(\rho,[P_\eta]|G))$, the probability of honour $p(\text{Observe}(\beta,P_\eta)|\text{Commit}(\eta,P)\text{ and Commit}(\rho,[P_\eta]|G))$, and the probability of goal realisation $p(\text{Observe}(\alpha,G)|\text{Observe}(\beta,P_\eta)\text{ and Commit}(\eta,P)\text{ and Commit}(\rho,[P_\eta]|G))$. That is:

$$p(\text{Observe}(\gamma,[P_\eta]|G)|\text{Commit}(\rho,[P_\eta]|G)) =$$

$$p(\text{Commit}(\eta,P)|\text{Commit}(\rho,[P_\eta]|G)) \cdot$$

$$p(\text{Observe}(\beta,P_\eta)|\text{Commit}(\eta,P)\text{ and Commit}(\rho,[P_\eta]|G)) \cdot$$

$$p(\text{Observe}(\alpha,G)|\text{Observe}(\beta,P_\eta)\text{ and Commit}(\eta,P)\text{ and Commit}(\rho,[P_\eta]|G))$$

Of course, when $\rho$ recommends $P$ to $\eta$ for realising goal $G$, $\eta$ may agree to a variation of the recommended plan ($P' \neq P$), perform a variation of what it has agreed upon ($P'' \neq P'$), and eventually, a variation of the intended goal ($G' \neq G$) may be realised. As such, the probability distribution that describes all possible expectations of the advice $[P_\eta]|G$ is defined accordingly, by considering all possible plans agreed upon and performed, as well as all possible realised goals:

$$P(\text{Observe}(\gamma,Y|X)|\text{Commit}(\rho,[P_\eta]|G)) =$$

$$\{p(\text{Observe}(\gamma,Y=P''_\eta|X=G')|\text{Commit}(\rho,[P_\eta]|G)), \ldots\} \forall \rho'' \in \mathcal{P}, G' \in \mathcal{G}$$

(11)

where $X$ is a variable and the range over which $X$ varies is $\mathcal{X}$, which describes the set of all possible goals that may be realised; and $Y$ is a variable and the range over which $Y$ varies is $\mathcal{P}$, which describes the set of all possible plans that may be committed to or performed.

2.2.1 Assumptions

Two assumptions are made by CONSUASOR. First, we say the plan that is performed is independent of the original recommended plan, yet dependent on the accepted (or committed to) plan. For example, if the tutor recommends the plan “practice three times a week”, and the student commits to “practicing two times a week”. Then what the student actually performs will not be dependent on what the tutor has recommended, but on what it has committed to.

Second, we say the goal that is realised is independent of the original recommended plan and independent of the accepted (or committed to) plan, yet dependent on the performed plan. This is because the realised goal is the outcome of a causal relation that links the performed plan to the realised goal. For example, assume that the plan “do not submit assignment” results in the goal “fail course”. Then whatever the tutor recommended, and whatever the student accepted (or committed to do), if the student eventually does not submit his assignment then he will fail the course.

We formally define these two assumptions next.

Assumption 1 (Conditional Independence I) The events $\text{Observe}(\beta,P_\eta)$ and $\text{Commit}(\rho,[P_\eta]|G)$ are conditionally independent given $\text{Commit}(\eta,P)$. That is:

$$p(\text{Observe}(\beta,P_\eta)|\text{Commit}(\eta,P)\text{ and Commit}(\rho,[P_\eta]|G)) =$$

$$p(\text{Observe}(\beta,P_\eta)|\text{Commit}(\eta,P))$$
Assumption 1 states that, given knowledge that \( \eta \) agreed to \( P(\text{Commit}(\eta,P)) \), knowledge of whether \( \rho \) recommended \( [P_{\eta}]G\) (\( \text{Commit}(\rho,[P_{\eta}]G) \)) provides no information on the likelihood of \( \eta \) performing \( P(\text{Observe}(\beta,P_{\eta})) \), and the knowledge of \( \eta \) performing \( P(\text{Observe}(\beta,P_{\eta})) \) provides no information on the likelihood of \( \rho \) recommending \([P_{\eta}]G\) (\( \text{Commit}(\rho,[P_{\eta}]G) \)).

**Assumption 2 (Conditional Independence II)** The events \( \text{Observe}(\alpha,G) \) and \( \text{Commit}(\eta,P) \) and \( \text{Commit}(\rho,[P_{\eta}]G) \) are conditionally independent given \( \text{Observe}(\beta,P_{\eta}) \). That is:

\[
 p(\text{Observe}(\alpha,G) \mid \text{Observe}(\beta,P_{\eta}) \text{ and } \text{Commit}(\eta,P) \text{ and } \text{Commit}(\rho,[P_{\eta}]G)) = p(\text{Observe}(\alpha,G) \mid \text{Observe}(\beta,P_{\eta}))
\]

Assumption 2 states that, given knowledge that \( \eta \) performed \( P(\text{Observe}(\beta,P_{\eta})) \), knowledge of whether \( \rho \) recommended \([P_{\eta}]G\) and \( \eta \) agreed to \( P(\text{Commit}(\eta,P)) \) and \( \text{Commit}(\rho,[P_{\eta}]G) \) provides no information on the likelihood of goal \( G \) being realised (\( \text{Observe}(\alpha,G) \)), and the knowledge of goal \( G \) being realised (\( \text{Observe}(\alpha,G) \)) provides no information on the likelihood of \( \rho \) recommending \([P_{\eta}]G\) and \( \eta \) agreeing to \( P(\text{Commit}(\eta,P)) \) and \( \text{Commit}(\rho,[P_{\eta}]G) \)).

Given Assumptions 1 and 2, the probability of an advice realising its goal may then be simplified accordingly.7

**Proposition 2 (Probability of an advice realising its goal II)** The probability of an advice realising its goal, under Assumptions 2 and 2 is:

\[
 p(\text{Observe}(\gamma,[P_{\eta}]G) \mid \text{Commit}(\rho,[P_{\eta}]G)) = p(\text{Commit}(\eta,P) \mid \text{Commit}(\rho,[P_{\eta}]G)) \cdot p(\text{Observe}(\beta,P_{\eta}) \mid \text{Commit}(\eta,P)) \cdot p(\text{Observe}(\alpha,G) \mid \text{Observe}(\beta,P_{\eta}))
\]

### 2.2.2 Notation

For simplification, in the remainder of this part of the paper we will use the notation \( P_{\text{c}}(X \mid [P_{\eta}]G) \) to describe the probability distribution of \( \eta \) committing to a plan \( X \) at time \( t \) given that the advice \([P_{\eta}]G\) has been recommended by \( \rho \). In other words, \( P_{\text{c}}(X \mid [P_{\eta}]G) = P(\text{Commit}(\eta,X) \mid \text{Commit}(\rho,[P_{\eta}]G)) \).

We refer to this as the probability distribution on compliance, which describes the compliance of \( \eta \) with \( \rho \)’s recommended advice \([P_{\eta}]G\).

Similarly, we use the notation \( P_{\text{h}}(X \mid P_{\eta}) \) to describe the probability distribution of \( \eta \) performing plan \( X \) at time \( t \) given that it has committed to the plan \( P \). In other words, \( P_{\text{h}}(X \mid P_{\eta}) = P(\text{Observe}(\beta,X_{\eta}) \mid \text{Commit}(\eta,P)) \). We refer to this as the probability distribution on honour, which describes \( \eta \)’s honour in performing the plan it has committed to perform.

We also use the notation \( P_{\text{r}}(X \mid P) \) to describe the probability distribution of realising goal \( X \) at time \( t \) given that \( P \) has been performed. In other words, \( P_{\text{r}}(X \mid P) = P(\text{Observe}(\alpha,X) \mid \text{Observe}(\beta,P_{\eta})) \). We refer to this as the probability distribution on goal realisation, which describes the realisation of goal \( G \) as a result of performing plan \( P \).

Finally, we will use the notation \( P_{\text{c} \mid \text{r}}(X \mid Y) \) to refer to any of the probability distributions on compliance, honour, or goal realisation.

---

7The proof for Proposition 2 is straightforward: By applying the rules of Assumptions 1 and 2, Proposition 1 is reduced to Proposition 2.
2.2.3 Initialisation

In this part of the paper, we provide a method that calculated a probability distribution $P_{C|H|R}(X|Y)$ incrementally, by updating the probability distributions on compliance, honour, and goal realisation for every experience in the history of experiences $H$. However, whether the update of the probability distributions is triggered every time a new experience is recorded, whether it is triggered on demand every time a trust measure needs to be calculated, or whether it is executed periodically is an implementation detail that we do not discuss in our presentation of the model (although Section 2.4 does provide one approach that we implement and use in our evaluation).

What is important though, is that when a probability distribution is updated, it is updated by considering the oldest experiences first. In other words, the temporal order of the experiences decides which experience needs to be considered first for updating the probability distributions.

At the initial time $t_I$ when no experiences have been considered yet in calculating the distribution, the initial value of the probability distribution may be domain dependent. Alternatively, for domain independent values, one can choose the uniform distribution $F = \{ p(x_1) \mapsto 1/n, \ldots, p(x_n) \mapsto 1/n \}$ to describe ignorance. That is, $P_{C|H|R}(X|Y) = F$. As new experiences are considered, the probability distribution gets reshaped, or updated, by incorporating the information learned from these experiences. We note that the support of $P_C(X|P_\eta)$ and $P_H(X|P_\eta)$ is the set of all plans $\mathcal{P}$, and the support of $P_R(X|P)$ is the set of all possible goals $\mathcal{G}$.8

2.2.4 Update of the probability distributions on compliance, honour, and goal realisation

To calculate the expected outcome of an advice, as defined by Equation [11], CONSUASOR needs to keep track of the probabilities on compliance, honour, and goal realisation. While the previous section presented the initial values of the probability distributions on compliance, honour, and goal realisation, this section illustrates how these distributions are updated when considering new experiences, as these distributions are learnt from similar past experiences.

Every time an experience $\mu$ is considered for updating a probability distribution $P_{C|H|R}(X|Y)$, the following steps are taken:

1. The relevance of $\mu$ w.r.t. $P_{C|H|R}(X|Y)$ is calculated.

The relevance is calculated differently for each context:

a) Compliance: Relevance of $\mu$ with respect to $P_C(X|[P_\eta]G)$.

Given a past experience where $\eta$ committed to $P'$ when $[P_\eta]G'$ was advised, the question then is: how much relevant is this past experience in informing us about the possible commitment of $\eta$ given the current advice $[P_\eta]G$? To define this measure of relevance, we use analogical

---

8There are drawbacks to setting the support of the distributions to the set of all plans or goals. For example, in some cases, the set of all possible goals/plans might be dynamic. In other cases, the set of all possible goals/plans might be massive in size, which would result in an extremely inefficient algorithm. One approach to address such issues is to have the sets of plans and goals learned from experiences. As such, the initial set of goals would be $\mathcal{G} = \{unknown\}$ and the initial set of plans would be $\mathcal{P} = \{unknown\}$, where the probability of unknown is initially 1. This describes that with the lack of any experience, one can expect some unknown thing to happen (specified as unknown). Then, as experiences are formed, they add their own accepted and/or performed plans to $\mathcal{P}$, and their realised goals to $\mathcal{G}$. In other words, the sets of possible plans and goals are updated by learning from experiences. We note that while this is the implementation of our choice, this paper assumes fixed sets of plans and goals to keep the presentation of the CONSUASOR model relatively simple.
reasoning by stating that $\eta$ might behave similarly to how it behaved in experience $\mu$ if $\mu$’s advice $(P^\eta G)$ is similar to the current advice $(P_\eta G)$. As an advice is composed of both a plan and a goal, we compute the similarity between advice $(P^\eta G)$ and $(P_\eta G)$ by relying on the semantic similarity between the advice’s goals $(Sim(G', G))$ and plans $(Sim(P', P))$. For instance, if $\eta$ has been accepting or rejecting plans that have been recommend for a given goal $G$, the it will most likely behave similarly when a plan for a similar goal is recommended. Similarly, if $\eta$ accepts or rejects a plan in the past (whether the reason to accept or reject was based on its willingness or capability of performing the plan), then it will most likely behave similarly when a similar plan is recommended in the future.

As $\mathbb{P}_C(X|P_\eta^G)$ describes the expectation of $\eta$’s compliance with $\rho$’s recommended advice $(P_\eta G)$, we only consider experiences where $\eta$ has been given advice in the past by similar advisers (where the similarity of advisers is specified by $S_A(\rho, \rho')$, and is defined shortly), in order to learn from $\eta$’s compliances in similar past scenarios.

We say, given a new experience $\mu = \langle Commit(\rho', [P_\eta^G, G'), t, -\rangle \rangle$, that $S_A(\rho, \rho') \geq \zeta_\rho$, we calculate $\mu$’s relevance to $\mathbb{P}_C(X|P_\eta^G)$ as follows:

$$R_\mu(\mathbb{P}_C(X|P_\eta^G)) = \frac{\zeta_g \cdot Sim(G', G) + \zeta_p \cdot Sim(P', P)}{\zeta_g + \zeta_p} \tag{12}$$

where $\zeta_g, \zeta_p \in [0, 1]$ are parameters that specify the importance of each similarity measure, and $\zeta_\rho$ of the condition $S_A(\rho, \rho') \geq \zeta_\rho$ describes the similarity level of advisers.

The similarity between two advisers is calculated by comparing their past advice, and it is based on the rationale that two advisers $\rho$ and $\rho'$ are similar if they recommend similar plans for similar goals. As such, we aggregate the similarity of $\rho$ and $\rho'$’s recommended plans, where the weight given to the similarity of each pair of plans is defined by the similarity of the goals for which these plans were recommended:

$$S_A(\rho, \rho') = \sum_{\forall [P_\eta^G \in \mathcal{A}(\rho), P_\eta^G \in \mathcal{A}(\rho')] \exists G} \frac{Sim(P, P') \cdot Sim(G, G')}{\sum_{\forall [P_\eta^G \in \mathcal{A}(\rho), P_\eta^G \in \mathcal{A}(\rho')] \exists G} Sim(G, G')} \tag{13}$$

where $\mathcal{A}(\rho) = \{ [P_\eta G] \mid \langle Commit(\rho, [P_\eta G]), t, -\rangle \rangle \in H \}$ describes the set of advice that $\rho$ has given in the past. Note that following Equation 13 it is sufficient for one pair of advice to share very similar goals for the similarity of their advisers to be dominated by the similarity of the plans of that specific pair of advice. Alternative approaches for calculating the similarity of advisers may also be adopted, such as the collaborative filtering techniques of recommender systems [13].

In the general case, we say that if the similarity between the current adviser $\rho$ and the adviser $\rho'$ of $\mu$ is greater than a certain threshold $\zeta_\rho$, specified as $S_A(\rho, \rho') \geq \zeta_\rho$, then the experience $\mu$ is considered. If $\zeta_\rho = 1$, then we are only considering experiences where advice has been suggested to $\eta$ by $\rho$, which means that the probability will describe the compliance of $\eta$ with $\rho$’s advice. If $\zeta_\rho = 0$, then we are considering experiences where advice has been suggested to $\eta$ by any adviser, which means that the probability will describe the compliance of $\eta$ in general. If $0 \leq \zeta_\rho \leq 1$, then

\footnotesize
\begin{itemize}
  \item In this paper, we use the underscore symbol \_\_, as in Prolog, to refer to an anonymous variable and it means “any term”.
\end{itemize}
we are considering experiences where advice have been suggested to \( \eta \) by advisers that share with \( \rho \) a similarity level of \( \zeta_r \).

b) Honour: Relevance of \( \mu \) with respect to \( \mathbb{P}_H(X|P_\eta) \).

Given a past experience where \( \eta \) committed to \( P' \) and then performed \( P'' \), the question then is: how much relevant is this past experience in informing us about what \( \eta \) will perform if it has committed to \( P' \)? To define this measure of relevance, we again use analogical reasoning by stating that \( \eta \) might behave similarly to how it behaved in experience \( \mu \) if the plan it committed to in \( \mu \) \( (P') \) is similar to the plan it currently committed to \( (P) \). However, the similarity between \( P' \) and \( P \) is not based on semantic similarity as above, but on a measure of empowerment. This is because a necessary condition for performing a plan is to be capable of performing the plan. As such, it is relevant to know how \( \eta \) behaved when it committed to a plan empowered by the plan under consideration. For instance, if it diverted much from it, then it is highly possible that it will divert from its current plan.

As \( \mathbb{P}_H(X|P_\eta) \) describes the expectation of \( \eta \)'s honour in fulfilling what it has committed to, we only consider experiences where \( \eta \) has committed and performed plans in order to learn from \( \eta \)'s past honour. We say given a new experience \( \mu = \langle _,\text{Commit}(\eta, P'), \text{Observe}(\beta, P'_\eta), _, \rangle \), we define \( \mu \)'s relevance to \( \mathbb{P}_H(X|P_\eta) \) as follows:

\[
R_\mu(\mathbb{P}_H(X|P_\eta)) = \text{Emp}(P, P')
\]  

(14)

c) Goal realisation: Relevance of \( \mu \) with respect to \( \mathbb{P}_R(X|P) \).

Given a past experience where \( G' \) was realised as a result of performing plan \( P' \), the question then is: how much relevant is this past experience in informing us about what will be realised if the current plan \( P \) is performed? To define this measure of relevance, we again use analogical reasoning by stating that the goal that will be realised might be similar to the goal that was realised in experience \( \mu \) if the plan that was performed in \( \mu \) \( (P') \) is similar to the currently performed plan \( (P) \). Note that unlike the case on honour above, we now use the semantic similarity between plans \( \text{Sim}(P', P) \) as opposed to plan empowerment. This is because the capability of performing a plan is no longer an issue as plans have already been performed (and observed).

As \( \mathbb{P}_R(X|P) \) describes the expectation of realising a goal given that plan \( P \) was performed, we need to focus on the goals that may be realised as a result of an already performed plan. We assume the identity of who performed the plan is irrelevant. As such, and unlike the cases on compliance and honour above, we look at all experiences and not just those where \( \eta \) has performed the plan. We say given a new experience \( \mu = \langle _,\text{Observe}(\beta, P'_\eta), \text{Observe}(\alpha, G'), _, \rangle \), we calculate its relevance to \( \mathbb{P}_R(X|P) \) as follows:

\[
R_\mu(\mathbb{P}_R(X|P)) = \text{Sim}(P', P)
\]  

(15)

2. The distribution \( \mathbb{P}_G^{\text{last}}_{\text{C}|H|R}(X|Y) \) is retrieved and decayed.

\(^{10}\)We assume that the causal relationship between performed plans and realised goals to be independent of who performs the plan. In other words, the same plan always results in the same goal, regardless of who performs the plan. This might not always be true, however. For example, one student may need to study one hour a day to pass the exam, while another might need to study much more than that. To address such cases, and in order to learn from similar past experiences, we need to look for experiences where advisees with similar profiles were observed performing some plans. This, however, requires a measure of similarity for advisees’ profiles, whose definition is left for future work.
If $P_{\text{CH|R}}^{\text{last}}(X|Y)$ has not been updated by any experiences yet, then the initial distribution is retrieved ($P_{\text{CH|R}}^{\text{last}}(X|Y) = P_{\text{CH|R}}(X|Y)$). Otherwise, the latest probability distribution is retrieved ($P_{\text{CH|R}}^{\text{last}}(X|Y)$) and the distribution is decayed according to the decay approach of Section 2.1.4. This allows us to take into consideration how the information represented by a probability distribution in question has been updated last, $t_\mu$ to refer to the time the experience $\mu$ occurred, and $P_{\text{CH|R}}^{\text{last} \rightarrow \mu}(X|Y)$ to refer to the decayed distribution. Also note that $P_{\text{CH|R}}^{\text{last}}(X|Y)$ is not decayed to the current time, but to the time that the experience $\mu$ occurred ($t_\mu$), since the objective of this update is to update $P_{\text{CH|R}}(X|Y)$ with respect to $\mu$.

Concerning the time the experience $\mu$ occurred, which we refer to as $t_\mu$, recall that an experience $\mu$ has four timestamps, as illustrated earlier in Section 2.1.1. The timestamp that represents the time $\mu$ occurred depends on the probability distribution in question. In the case of updating the probability distribution on compliance ($P_C(X|P_\eta^\mu G)$), $t_\mu$ is the timestamp of the second element of $\mu$, or the time $\eta$ committed to a plan. In the case of updating the probability distribution on honour ($P_H(X|P_\eta)$), $t_\mu$ is the timestamp of the third element of $\mu$, or the time $\eta$ performed a plan. In the case of updating the probability distribution on goal realisation ($P_R(X|P)$), $t_\mu$ is the timestamp of the fourth element of $\mu$, or the time a goal was realised.

### 3. The probability $P_{\text{CH|R}}^{\mu}(X=x|Y)$ is calculated.

The experience $\mu$ is used to decide which expectation needs to have its probability updated and how is this new probability calculated. The decision of which expectation should have its probability updated is based on analogical reasoning, as motivated earlier. For instance, if $\eta$ committed to a very different plan than what was recommended in experience $\mu$, and $\mu$ is highly relevant to the current scenario, then one can expect $\eta$ to commit to a very different plan than what is currently recommended. Similarly, if $\eta$ strongly diverted from the plan that it has committed to in experience $\mu$, and $\mu$ is highly relevant to the current scenario, then one can expect $\eta$ to strongly divert from the currently committed plan too, following a similar behaviour to $\mu$. Similarly, if a goal was realised in experience $\mu$, and the performed plan in $\mu$ is very similar to the plan under consideration, then one can expect a similar goal to be realised. This analogical reasoning is implemented as follows.

How a probability of an expectation $X=x$ is updated follows a similar approach for all the three probability distributions on compliance, honour, and goal realisation. The probability of an expectation $X=x$ is updated based on the relevance of the experience $\mu$ and it follows the probability update of Section 2.1.3 by applying Equation 6 accordingly:

$$P_{\text{CH|R}}^{\mu}(X=x|Y) = P_{\text{CH|R}}^{\text{last} \rightarrow \mu}(X=x|Y) + (1 - P_{\text{CH|R}}^{\text{last} \rightarrow \mu}(X=x|Y)) \cdot \varepsilon \cdot R_\mu(P_{\text{CH|R}}(X|Y))$$

where, following the analogical reasoning presented above, the probability update is carried out for an expectation $X=x$ if:

a) $\text{Sim}(P, x) \approx \text{Sim}(P', P'')$, when updating the probability of committing to a plan ($P_C^{\mu}(X = x|P_\eta G)$) given an experience $\mu = \langle \text{Commit}(\rho', [P_\eta'] G'), \text{Commit}(\eta, P''), t_\mu \rangle$. That is,

---

The value of $\varepsilon$ may be based on considering factors that are not related to the experience $\mu$, and yet they may influence the probability in question. For example, when calculating the probability on compliance, $\varepsilon$ may depend on social commitments. In other words, the stronger the social commitment between $\eta$ and the current recommender $\rho$ then the higher the probability that $\eta$ will commit to $\rho$’s current advice.
we consider an expectation \( X = x \) based on its semantic distance to the recommended plan \( P \) in such a way that this distance is approximately equivalent to the semantic distance between the plan that was recommended in experience \( \mu \) and the plan that \( \eta \) committed to then.

b) \( Sim(P, x) \approx Sim(P', P'') \), when updating the probability of performing a plan \( (p^\mu_{\eta}(X = x|P_\eta)) \) given an experience \( \mu = \langle \ldots, Commit(\eta, P'), Observe(\beta, P''_\eta), \ldots \rangle \). That is, we consider an expectation \( X = x \) based on its semantic distance to the committed plan \( P \) in such a way that this distance is approximately equivalent to the semantic distance between what \( \eta \) has committed to in experience \( \mu \) and what it has performed then.

c) \( Sim(G, x) \approx Sim(P, P') \), when updating the probability of a goal being realised \( (p^\mu_{\eta}(X = x|P)) \) given an experience \( \mu = \langle \ldots, Observe(\beta, P'_\eta), Observe(\alpha, G'), \ldots \rangle \). That is, we consider an expectation \( X = x \) based on its semantic distance to the realised goal \( G \) of \( \mu \) in such a way that this distance is approximately equivalent to the semantic distance between the performed plan \( P' \) of \( \mu \) and the plan \( P \) under consideration.

Note that more than one expectation may satisfy the condition of semantic equivalence, and as a result, the probability of more than one expectation may be updated.

4. The probability distribution \( P^{\mu}_{\mu\eta}X|Y \) is calculated.

Given the probability \( P^{\mu}_{\mu\eta}(X = x|Y) \) (or a set of probabilities \( \{p^{\mu}_{\mu\eta}(X = x|Y), \ldots\} \) ), we calculate the new probability distribution \( P^{\mu}_{\mu\eta}X|Y \) by applying Equation 7 of Section 2.1.3 which follows the minimum relative entropy approach:

\[
\begin{align*}
P^{\mu}_{\mu\eta}X|Y & = \arg \min_{\overline{P}(X)} \sum_{i} t_{\mu\eta}^{-\alpha}(X = i|Y) \log \frac{p^{\mu\eta}_{\mu\eta}(X = i|Y)}{p(X = i)} \quad \text{such that} \quad \{p(X = x) = p^{\mu}_{\mu\eta}(X = x|Y), \ldots\}
\end{align*}
\]

2.3 Trust Computation

With updated probability distributions on compliance, honour, and goal realisation, \( P'(Observe(\alpha, X) | Commit(\rho, [P_\eta|G])) \) is calculated via Equation [11] which we simply refer to as \( P'(X|P_\eta|G) \) for simplification. The question now is: How do we interpret such expectations? In other words: How do we calculate a trust measure given an expectation specified as a probability distribution?

Different trust equations can be implemented, depending on the particular context or interest. For example, a trust measure may be based on the preference of outcomes: given the preferences of possible outcomes, the trust measure will be higher when preferred outcomes are more likely to happen than less preferred outcomes. A trust measure may be based on the certainty of the expected outcomes, where entropy may be used to measure uncertainty: the higher the certainty of outcomes, then the higher the trust measure is, and vice versa. \([29]\) presents a few approaches for calculating trust, including those that are based on preferences or certainty of outcomes.

One alternative straightforward approach, which we present next, is based on the distance between what the advice promised \((P_{\eta}(X|P_\eta|G) = \{1, if X = G; 0, otherwise\})\) and what is expected to be achieved \((P'(X|P_\eta|G))\). The trustworthiness of an advice \([P_\eta|G] \) is then calculated
accordingly:
\[
\]  
(16)

where the function \text{emd}, whose range is \([0,1]\), calculates the earth mover’s distance between two probability distributions.

Concerning the trustworthiness of an adviser \(ρ\), we say a good adviser is one who provides good advice. As such, the trustworthiness of an adviser is defined in terms of the trustworthiness of his advice. That is, the trustworthiness of \(ρ\) on giving advice becomes an aggregation of the trust \(\text{trust}([P^p_ρ]G)\) for every advice \([P^p_η]G\) that \(ρ\) has given in the past:

\[
\text{trust}^t(ρ) = \sum_{[P^p_η]G \in \mathcal{A}(ρ)} \text{trust}^t([P^p_η]G)
\]

(17)

where \(\mathcal{A}(ρ) = \{[P^p_η]G \mid \langle \text{Commit}(ρ,[P^p_η]G),t',\ldots,\ldots \rangle \in H \}\) describes the set of advice that \(ρ\) has given in the past \((t' < t)\).

### 2.4 Trust Algorithm

In this section, we present one approach for calculating the trustworthiness of advice. To calculate the trustworthiness of \(ρ\)’s advice \([P^p_η]G\), the relevant probability distributions on compliance, honour, and goal realisation should first be updated in order to update the probability distribution describing the outcome of \(ρ\)’s advice: \(\mathbb{P}^t_{\text{now}}(X|[P^p_η]G)\), where \(t_{\text{now}}\) describes current time. Every time the trustworthiness of the advice needs to be calculated, the relevant probability distributions are updated by calling Algorithm 1. After updating the necessary distributions and calculating \(\mathbb{P}^t_{\text{now}}(X|[P^p_η]G)\), the final trust measure on \(ρ\)’s advice \((\text{trust}^t_{\text{now}}([P^p_η]G))\) is calculated following Equation 16.

In our proposed approach, trust is calculated on demand. Other implementations that call Algorithm 1 to precompute probability distributions are possible. For example, Algorithm 1 may be executed periodically for all possible advice to ensure that when the trustworthiness of a specific advice is requested, minimal time and effort is spent on updating the relevant probability distributions.

To help update a probability distribution \(\mathbb{P}(X|[P^p_η]G)\), originally specified as \(\mathbb{P}(\text{Observe}(α,X) \mid \text{Commit}(ρ,[P^p_η]G))\), Algorithm 1 updates the relevant probability distributions on compliance, honour, and goal realisation. In Algorithm 1, the similarity between advisers follows Equation 13, the similarity between plans or goals follows Equation 2, and plan empowerment follows Equation 5. Updating the probability of a single expectation follows Equation 6, updating a probability distribution follows Equation 7, the decay follows Equation 9, and the relevance of an experience follows Equations 12, 14, and 15. It also assumes that both the initial distributions as well as the decay limit distributions are equiprobable distributions (i.e., they are equivalent to the uniform distribution \(F\)).

The implementation of Algorithm 1 follows an incremental approach were we use ‘memoization’ techniques, as in dynamic programming, to improve the efficiency of the algorithm. We do this by saving the latest probability distributions and updating older computations. In this way, when a probability distribution needs to be calculated and it has already been calculated in the past, the distribution is modified considering new experiences only.
Algorithm 1 update(\(P^n_θ_i\) | \(G\))

Require: \(H\), which specifies the history (or set) of past experiences.
Require: \(\mathcal{P}\) and \(\mathcal{R}\), which represent the sets of all plans and goals, respectively.
Require: \(\mathcal{P}\), which specifies the set of probability distributions that have been updated. Initially, we have \(\mathcal{P} = \emptyset\).
Require: \(P^n_θ_i = P^n_θ_i = P^n_θ_i = F\) describe the initial value of the probability distributions on compliance, honour, and goal realisation, respectively, and sets them all to the uniform distribution \(P\).
Require: \(\Delta_Ω = \Delta_Ω = \Delta_Ω = F\) describe the decay limit distributions for the probability distributions on compliance, honour, and goal realisation, respectively, and sets them all to the uniform distribution \(P\).
Require: \(ξ, ζ, η \in [0, 1]\), which describe the threshold for considering an experience relevant when updating the probabilities on compliance, honour, and goal realisation, respectively.
Require: \(S_θ, S_θ, S_θ \in [0, 1]\), which describe the threshold for considering two advisers similar.
Require: \(\epsilon, \epsilon, \epsilon \in [0, 1]\), which describes the percentage by which a probability on compliance, honour, and goal realisation, respectively, may increase.
Require: \(\nu \in [0, 1]\), which describes the decay rate.
Require: \(\Delta_Ω : T \times T \rightarrow [0, 1]\), which describes the pace of decay, where \(T\) represents time. (See Section 2.1.4)
Require: \(t_{now} \in T\), where \(t_{now}\) represents current time.

▷ First, the probability distribution on compliance is updated.
▷ Get the experiences that have not been taken into account yet.

\[\text{if } P^n_θ_i (X | \binom{P^n_θ_i}{G}) \in \mathcal{P} \text{ then}\]

\[H^* = \{ \mu | \mu \in H \text{ and } \mu = \langle \text{Commit}(x, y), \ldots, \text{Commit}(x, y), \ldots \rangle \text{ and } t_\mu > t_{now} \}\]

else

\[H^* = H\]

\[\mathcal{P}^{new} (X | \binom{P^n_θ_i}{G}) = \mathcal{P}^C\]

end if

for all \(\mu \in H^* \text{ and } \mu = \langle \text{Commit}(x, y), \ldots, \text{Commit}(x, y), \ldots \rangle \text{ and } S_θ(\rho, \rho') > \xi_θ\) do

▷ The relevance of \(\mu\) with respect to compliance is calculated.

\[R_\mu (\mathcal{P}_C (X | \binom{P^n_θ_i}{G})) = \frac{\xi_θ \cdot \text{Sim}(G', G) + \xi_θ \cdot \text{Sim}(P', P)}{\xi_θ + \xi_θ}\]

if \(R_\mu (\mathcal{P}_C (X | \binom{P^n_θ_i}{G})) > \xi_θ\) then

▷ \(\mathcal{P}^{new} (X | \binom{P^n_θ_i}{G})\) is decayed to \(t_\mu\)

\[P^n_θ_i^{\Delta_Ω - t_\mu} (X | \binom{P^n_θ_i}{G}) = \left( 1 - \Delta_Ω \right) P^n_θ_i (X | \binom{P^n_θ_i}{G})\]

▷ An expectation \(x\) is chosen

\[x \in \{ y | \text{Sim}(P, y) = \text{Sim}(P', P') \}\]

▷ The probability of the expectation \(x\) is calculated.

\[P^n_θ_i (x | \binom{P^n_θ_i}{G}) = P^n_θ_i^{\Delta_Ω - t_\mu} (x | \binom{P^n_θ_i}{G}) + (1 - P^n_θ_i^{\Delta_Ω - t_\mu} (x | \binom{P^n_θ_i}{G}) \cdot \epsilon_θ \cdot R_\mu (\mathcal{P}_C (X | \binom{P^n_θ_i}{G}))\]

▷ The probability distribution is updated

\[P^n_θ_i (X | \binom{P^n_θ_i}{G}) = \arg \min_{\mathcal{P}(X)} \sum_i P_{C}^{\Delta_Ω - t_\mu} (X=i | \binom{P^n_θ_i}{G}) \log \frac{P_{C}^{\Delta_Ω - t_\mu} (X-i | \binom{P^n_θ_i}{G})}{P (X=i)}, \text{ such that } P (X=x) = P^n_θ_i (X=x | \binom{P^n_θ_i}{G})\]

▷ \(\mathcal{P}\) is updated to contain the latest distribution \(P^n_θ_i (X | \binom{P^n_θ_i}{G})\).

\[\mathcal{P} = \mathcal{P}^{new} (X | \binom{P^n_θ_i}{G}) \ominus \mathcal{P}_C^{new} (X | \binom{P^n_θ_i}{G})\]

end if

end for
for all \( P' \in \mathcal{P} \) do

\begin{align*}
\text{if } & \mathbb{P}_H(X|P'_\eta) \in \mathcal{P} \text{ then} \\
& H^* = \{ \mu \mid \mu \in H \text{ and } \mu = \langle x, \ldots, \text{Observ}(x,y) \rangle \text{ and } t_\mu > t_{\text{last}} \} \\
\text{else} \quad & H^* = H \\
& \mathbb{P}_H(X|P'_\eta) = \mathbb{P}_H' \\
\text{end if} \\
\text{for all } & \mu \in H^* \text{ and } \mu = \langle \text{Commit}(\eta, P'_\eta), \text{Observ}(\alpha, P''_\eta) \rangle \text{ do} \\
& R_\mu(\mathbb{P}_H(X|P'_\eta)) = \text{Emp}(P', P'') \\
\text{if } & R_\mu(\mathbb{P}_H(X|P'_\eta)) > \tilde{\xi}_H \text{ then} \\
& \mathbb{P}_H^\text{flow}_{\nu_{\text{last}}} (X|P'_\eta) = \nu_{\text{last}} \cdot \mathbb{P}_H(X|P'_\eta) + (1 - \nu_{\text{last}}) \mathbb{P}_H' \\
& \forall x \in \{ y \mid \text{Sim}(P', y) = \text{Sim}(P'', P'') \} \\
& p_H(x = x'|P'_\eta) = p_H^\text{flow}_{\nu_{\text{last}}} (x = x'|P'_\eta) + (1 - p_H^\text{flow}_{\nu_{\text{last}}} (x = x'|P'_\eta)) \cdot \tilde{\epsilon}_H \cdot R_\mu(\mathbb{P}_H(X|P'_\eta)) \\
& \mathbb{P}_H(X|P'_\eta) = \arg \min_{P(x)} \sum_x p_{\text{flow}_{\nu_{\text{last}}}}(x = i|P'_\eta) \log \frac{p_H(x = i|P'_\eta)}{p(x = i)} \text{, such that } p(x = x) = p_H^\text{flow}_{\nu_{\text{last}}} (x = x'|P'_\eta) \\
& \mathcal{D} = \mathcal{D} \cup \mathbb{P}_H^\text{flow}_{\nu_{\text{last}}}(X|P'_\eta) \\
\text{end if} \\
\text{end for} \\
\text{end for} \\
\text{Third, the relevant probability distributions on goal realisation are updated in a similar manner to those on compliance and honour.} \\
\text{for all } P' \in \mathcal{P} \text{ do} \\
\text{if } \mathbb{P}_R^\text{flow}(X|P') \in \mathcal{P} \text{ then} \\
& H^* = \{ \mu \mid \mu \in H \text{ and } \mu = \langle x, \ldots, \text{Observ}(x,y) \rangle \text{ and } t_\mu > t_{\text{last}} \} \\
\text{else} \quad & H^* = H \\
& \mathbb{P}_R^\text{flow}(X|P') = \mathbb{P}_R' \\
\text{end if} \\
\text{for all } & \mu \in H^* \text{ and } \mu = \langle \text{Observ}(\alpha, P''_\eta) \rangle \text{ do} \\
& R_\mu(\mathbb{P}_R(X|P')) = \text{Sim}(P', P'') \\
\text{if } & R_\mu(\mathbb{P}_R(X|P')) > \tilde{\xi}_R \text{ then} \\
& \mathbb{P}_R^\text{flow}_{\nu_{\text{last}}} (X|P') = \nu_{\text{last}} \cdot \mathbb{P}_R(X|P') + (1 - \nu_{\text{last}}) \mathbb{P}_R' \\
& \forall x \in \{ y \mid \text{Sim}(P', y) = \text{Sim}(P'', P'') \} \\
& p_R(x = x'|P') = p_R^\text{flow}_{\nu_{\text{last}}} (x = x'|P') + (1 - p_R^\text{flow}_{\nu_{\text{last}}} (x = x'|P')) \cdot \tilde{\epsilon}_R \cdot R_\mu(\mathbb{P}_R(X|P')) \\
& \mathbb{P}_R(X|P') = \arg \min_{P(x)} \sum_x p_{\text{flow}_{\nu_{\text{last}}}}(x = i|P') \log \frac{p_R(x = i|P')}{p(x = i)} \text{, such that } p(x = x) = p_R^\text{flow}_{\nu_{\text{last}}} (x = x'|P') \\
& \mathcal{D} = \mathcal{D} \cup \mathbb{P}_R^\text{flow}_{\nu_{\text{last}}}(X|P') \\
\text{end if} \\
\text{end for} \\
\text{end for} \\
\text{Finally, update the probability distribution } \mathbb{P}\text{flow}_{\nu_{\text{last}}}(X|P'_\eta| G). \\
\text{for all } x \in \mathcal{S} \text{ do} \\
& p_{\text{flow}_{\nu_{\text{last}}}}(X = x|P'_\eta| G) = 0 \\
\text{for all } y \in \mathcal{S} \text{ do} \\
\text{for all } z \in \mathcal{S} \text{ do} \\
\text{if } \mathbb{P}_H^\text{flow}_{\nu_{\text{last}}}(Y|z_\eta) \in \mathcal{D} \text{ and } \mathbb{P}_R^\text{flow}_{\nu_{\text{last}}}(X|y) \in \mathcal{D} \text{ then} \\
& p_{\text{flow}_{\nu_{\text{last}}}}(X = x|P'_\eta| G) = p_{\text{flow}_{\nu_{\text{last}}}}(X = x|P'_\eta| G) + (X = z|P'_\eta| G) \cdot \mathbb{P}_C^\text{flow}_{\nu_{\text{last}}}(Z = z|P'_\eta| G) \cdot \mathbb{P}_R^\text{flow}_{\nu_{\text{last}}}(Y = y|z_\eta) \cdot \mathbb{P}_R^\text{flow}_{\nu_{\text{last}}}(X = x|y) \\
\text{end if} \\
\text{end for} \\
\text{end for} \\
\text{end for}
The algorithm uses parameters $\xi_C$, $\xi_H$, and $\xi_R$ to specify the thresholds on when to consider an experience relevant. By fixing the values of these parameters to high values, we can improve the efficiency of the algorithm even further by saying that experiences that would result in ‘small’ modifications to past probability distributions are not to be considered. By reducing the values of these parameters progressively, we can have more realistic and fine grained implementations.

3 Evaluation

In this Section we provide an empirical evaluation based on a simulation of advisers and advisees interaction. We refer to advisers and advisees as recommenders and users, respectively. In the following we describe our experimental platform, define benchmarks for a music learning domain and show a comparison between our CONSUASOR algorithm, the well-known eigentrust algorithm [11], and a random method.

3.1 Experimental platform

The following sets and functions determine a benchmark:

- A set of actions $\mathcal{A}$ to build plans and an action meronomy $\mathcal{M}_a$ used to define particular similarity functions (based on [12]) and empowerment functions (based on OpinioNet [19]).
- A set of propositional terms $\mathcal{T}$ to define goals and a term ontology $\mathcal{O}_\mathcal{T}$ used to define particular similarity functions (based on [12]).
- The set of all plans $\mathcal{P}$ and the set of all goals $\mathcal{G}$.
- A causality function $f : \mathcal{P} \rightarrow \mathcal{G}$, which describes whether a plan achieves a goal.
- A set of users $\mathcal{U}$ where every $\eta \in \mathcal{U}$ is defined by the tuple $(G_t^\eta, d_1^\eta, d_2^\eta, c_\eta^\eta, h_\eta^\eta)$ such that $d_1^\eta, d_2^\eta \in [0, 1]$ and:
  - $G_t^\eta \subseteq \mathcal{G}$ is the set of $\eta$’s goals at every time instant $t$.
  - $c_\eta : \mathcal{P} \rightarrow \mathcal{P}$ and $h_\eta : \mathcal{P} \rightarrow \mathcal{P}$ are functions describing $\eta$ compliance and honor, therefore: when a plan $P$ is recommended to $\eta$, $\eta$ commits to $c_\eta(P)$; when $\eta$ commits to $P$, $\eta$ executes $h_\eta(P)$.
  - $d_1$ and $d_2$ describe the level of compliance and honor of $\eta$ satisfying the following conditions:  
    $\forall P \in \mathcal{P}$:
    - $\text{Sim}(c_\eta(P), P) \geq 1 - d_1$
    - $\text{Sim}(h_\eta(P), P) \geq 1 - d_2$

  Notice that when $d_1 = d_2 = 0$ a user is fully compliant and honorable. When $d_1 > 0, d_2 > 0$, the user may not commit to the recommended plan or execute its commitment, probably achieving a totally different goal.

- A set of recommenders $\mathcal{R}$ where every $\rho \in \mathcal{R}$ is defined by the tuple $(d_3, d_4, d_5, \{c_\eta^{-1}\}_{\eta \in \mathcal{U}}, \{h_\eta^{-1}\}_{\eta \in \mathcal{U}}, f^{-1})$ such that $d_3, d_4, d_5 \in [0, 1]$ and:
- \( c_{\eta}^{-1} \) describes \( \rho \)'s knowledge about \( \eta \)'s compliance, therefore: for \( \eta \) to commit to plan \( P \), \( \rho \) believes that \( c_{\eta}(P)^{-1} \) should be recommended to \( \eta \).

- \( h_{\eta}^{-1} \) describes \( \rho \)'s knowledge about the honor of \( \eta \), therefore: for \( \eta \) to execute plan \( P \), \( \rho \) believes that \( \eta \) should have committed to \( h_{\eta}(P)^{-1} \).

- \( f^{-1} \) describes \( \rho \)'s knowledge about the causality of plan reaching goals, therefore: to achieve goal \( G \), \( \rho \) believes that \( f(G)^{-1} \) must be executed.

- \( d_3, d_4 \) and \( d_5 \) describe the level of knowledge of \( \rho \) about compliance, honor and causality, respectively, satisfying the following conditions:

\[
\forall P \in \mathcal{P}: \quad Sim(c_{\eta}^{-1}(P), P') \geq 1 - d_3 \quad \text{and} \quad c_{\eta}(P') = P \\
Sim(h_{\eta}^{-1}(P), P') \geq 1 - d_4 \quad \text{and} \quad h_{\eta}(P') = P \\
Sim(f(G)^{-1}, P) \geq 1 - d_5 \quad \text{and} \quad f(G) = P
\]

Notice that when \( d_3 = 0 \), \( \rho \) is accurate about what plan \( P' \) must be recommended to \( \eta \) so that \( \eta \) commits to \( P \). When \( d_5 > 0 \), \( \rho \)'s knowledge is not accurate and \( c_{\eta}^{-1}(P) \) may not lead to \( \eta \) committing to \( P \). The same reasoning follows for \( d_4, d_5 \) on honor and causality. A perfect recommender is one with \( d_3 = d_4 = d_5 = 0 \).

We assume determinism and by construct Assumptions 1 and 2 are satisfied. A recommender \( \rho \) suggests a plan \( P \) for user \( \eta \) for goal \( G \) as follows:

\[
P = c_{\eta}^{-1}(h_{\eta}^{-1}(f^{-1}(G)))
\]

If the recommended plan \( P \) is accepted, a new experience \( \mu \) is generated and added to the history of experiences \( \mathcal{H} \). The new generated experience is:

\[
\mu = \langle [P_\eta]G_{t'}, c_{\eta}(P)_{t''}, h_{\eta}(c_{\eta}(P))_{t'''}, f(h_{\eta}(c_{\eta}(P)))_{t'''} \rangle
\]

where time instants \( t' < t'' < t''' < t'''' \) are generated with a pace of 1 between each other.

The success of a recommendation \([P]G\) for user \( \eta \) is defined as:

\[
\text{Succ}_\eta(P,G) = Sim(G,f(h_{\eta}(c_{\eta}(P))))
\]

And the success of a user \( \eta \) up to time \( t \) is defined as:

\[
\text{Succ}^t_\eta = \sum_{\mu} \frac{\text{Sim}(P,G,f(h_{\eta}(c_{\eta}(P))))}{|\mu|} \in \mathcal{H}
\]

where \([P_\eta]G\) is a recommendation selected by user \( \eta \) in time \( t' < t \). A trust strategy determines how to select \([P_\eta]G\) from all the suggestions made by recommenders. A trust strategy will be considered good for user \( \eta \) if \( \text{Succ}^t_\eta(\cdot, \cdot) \) increases over time. That is, if a time frame of size \( n \) is considered, then \((\text{Succ}^t_\eta - \text{Succ}^{t-n}_\eta)/n\) should approach the maximum success, which is determined by the user’s compliance and honour.
3.2 Benchmarks

In our benchmarks we consider an action meronomy $M_{af}$ and goal ontology $O_{af}$ of 10 terms related with music composition, improvisation and instrument performance (Figure 3).

Our set of plans $P$ contains 10 plans, one for every action in $M_{af}$. Causality function $f$ is built such that every plan in $P$ achieves a goal in $G$: $f(\text{Practice all}) = \text{Finest Musician}$, $f(\text{Practice Piano}) = \text{Good Piano Performant}$, $f(\text{Practice Band Improvisation}) = \text{Good Band Improviser}$, etc.

In the experiments there is always a single user $\eta \in U$. Compliance and honor functions $c_\eta(P)$ and $h_\eta(P)$ are built such that for every plan $P \in P$, a plan $P'$ is picked at a distance $d_1$ (for compliance) or $d_2$ (for honor). Notice that $c_\eta$ and $h_\eta$ are not injective functions, therefore when $d_1 \neq 0$ and $d_2 \neq 0$ it may be the case that $\eta$ never commits to a plan $P$ ($\nexists P' s.t. c_\eta(P') = P$) or honors a plan $P$ ($\nexists P' s.t. h_\eta(P') = P$).

The number of recommenders varies from 5 to 30. To generate $\rho$’s knowledge about $\eta$’s compliance ($c_\eta(P)^{-1}$), we first calculate the inverse of $c_\eta(P)$ and then pick a plan in $P$ at a distance $d_3$ from the inverse value. The same logic is followed to generate $h_\eta^{-1}$ and $f^{-1}$.

Experiments run as follows. The user needs advice to achieve goals and he receives advice from the recommenders. Once an advice is selected, an experience is generated where the user may or may not commit to the recommended plan and achieve the desired goal. As explained in Section 3.1, recommendations are generated according to the recommender’s knowledge and experiences are generated according to the user’s compliance and honor. In the experiments we perform several iterations and observe the trust evolution of the recommenders and the user’s success rate over time. Results are given for different values of $d_1, d_2, d_3, d_4, d_5$. For convenience we use the notation $\bar{d} = (d_1 + d_2 + d_3)/3$.

We compare 3 trust strategies: selecting advise randomly, selecting the advice whose adviser is ranked top by the eigentrust algorithm \cite{11,12} and selecting the advice/recommender ranked top by CONSUASOR.

The flexibility of the CONSUASOR model allows to consider several applications where we are able to focus on different aspects of advice. We provide different experimental scenarios. In

\footnote{In the eigentrust case, we compute the normalized local trust. The notion of transitive trust is not needed since neither users nor recommenders provide advise among themselves.}
3.4 Trust and reputation

FP7-ICT-2011- 8 Collaborative project 318770 PRAISE

Experiment 1, we calculate the trust on advices given by recommenders, while in Experiment 2 we calculate the trust on recommenders for a specific goal. Experiment 1 addresses the question: How good is ρ recommending \([P_\eta]G\)? while Experiment 2 addresses the question: How good is recommender ρ giving advice for goal \(G\)? We are interested in showing the flexibility and coherence of the model in cases were different approaches are needed. Finally, Experiment 3 presents a comparison of different CONSUAOR trust evaluations focusing on different aspects of advice. In every case, we present averaged results of 50 experiments with a time frame \(n = 5\).

3.3 Experiment 1. Trust on Recommender giving Advice

In these experiments the user \(\eta\) has a single goal (randomly picked from the goals ontology) which is fixed over time. There is a set of recommenders providing advice. Every experiment performs 100 iterations (time instants) of the following steps: (1) The user asks advice to all recommenders for its goal; (2) Each recommender provides a recommendation, as defined in Section 3.1; (3) The user selects one advice (in the random case, randomly, in the eigentrust and CONSUAOR cases, the most trusted one); and, (4) An experience is generated and stored, the trust of recommenders is updated and the success of the user is updated.

Figure 4 shows the CONSUAOR trust values evolution in time of 10 recommenders (each line represents the trust in a recommender providing a particular advice at time instant \(t\)). Recommenders knowledge vary from fully knowledgeable (small \(\bar{d}\)) to ignorant (large \(\bar{d}\)). Case (a) present results with a good user \((d_1 = d_2 = 0)\) which is fully compliant and honorable. In this case, recommenders with \(0 \leq \bar{d} \leq 0.6\) overlap. This is an effect of our small ontology: since we have just a few terms, the semantic distance between them is relatively large. We need to increase the distance beyond 0.6 to obtain recommenders that are not perfect. The trust on perfect recommenders \((0 \leq \bar{d} \leq 0.6)\) increases over time and reaches 1, while the trust on recommenders which are not fully knowledgeable \((0.7 \leq \bar{d} \leq 1)\) fluctuates up and down (a bad recommender may sometimes give a good recommendation but most of the times he does not). As expected, the worst the recommender, the lower its trust line is. Case(b) present results with a bad user \((d_1 = d_2 = 1)\). We observe the same behavior of Case (a) but trust measures are diminished (trust on perfect recommends reaches 0.5 approximately instead of 1). This is because a bad user does not necessarily commit to or honor every plan (because \(c_\eta\) and \(h_\eta\) are not injective functions). In such cases,
it is not possible to obtain a plan such that the user is willing to commit to (or execute) and that causes the goal fulfillment, according to the recommender’s knowledge ($\exists P''$ s.t. $c_\eta(P'') = P$ and $h_\eta(P') = P$ and $f(P) = G$). In other words, there is no good plan for this bad user. In such cases, the recommender does not provide any advice to the user and the trust on that recommender/advice is not measured (since the advice does not exist), then we simply put a zero as the trust value in that iteration and when executions are averaged this is reflected in the graphics. If we increase $d_1$ and $d_2$ ($\eta$’s compliance and honor), we see how the trust lines increase until we obtain a graph like (a) for a fully compliant and honorable user.

Figures 5 and 6 show $\eta$’s success evolution in time ($\text{Succ}_\eta^t$) for good and bad users, respectively, following the three strategies: random, eigentrust and CONSUASOR. Results in Figure 5 correspond to executions with a perfect user. Case 5(a) includes 30 perfect recommenders. Perfect recommenders always give good advice and $\eta$ always commits and executes that advice, therefore success is always 1. Cases 5(b) and 5(c) includes 30 medium and bad recommenders respectively, we observe how the CONSUASOR algorithm obtains high levels of success while eigentrust and random remain low. Case 5(d) includes 5 different recommenders from fully knowledgeable to ignorant, we observe that eigentrust performs well and that CONSUASOR stabilizes earlier. In the cases of Figure 6, the user $\eta$ is not always compliant and honorable ($d_1 = d_2 = 0.7$). We observe the same behaviour of previous cases but the success rate is diminished, since sometimes there is not any recommendation to achieve $\eta$’s goal that $\eta$ is willing to commit to or execute. For higher

![Figure 5: Experiment 1. Success evaluation in time, for good users.](image-url)
d₁ and d₂ values (less compliant/honorable user), we verified experimentally that the success lines are diminished and for lower d₁ and d₂ values the success lines increase.

These results show that CONSUASOR is able to increase η’s success rate in a short period of time compared with random and eigentrust strategies. Eigentrust obtains high levels of success when there is at least one good recommender, but when recommenders are not very knowledgeable or users are not fully compliant or honorable, its success diminishes considerably (similar to random in many cases). CONSUASOR is able to learn which advices are trustworthy and reaches high levels of success in all observed cases.

3.4 Experiment 2. Trust on Recommenders for Goals

In these experiments the user η has 3 goals that change over time every 40 time instants (G₁,...,₄₀ = Good Piano Performant, G₄₁,...,₈₀ = Good Band Improviser, G₈₁,...,₁₂₀ = Good Composer). η is fully compliant and honorable (d₁ = d₂ = 0) and there are 10 recommenders providing advice, each specialized in a different goal. Every experiment performs 100 iterations (time instants) of the following steps: (1) The user calculates the trust on each recommender for its goal Gᵢ (trust(ρ, [Gᵢ], Gᵢ)). (2) The user selects the most trusted recommender and asks him advice; (3) The selected recommender provides a recommendation, as defined in Section 3.1. (4) An experience is generated and stored, the trust of recommenders is updated and the success of the user is updated.
Figures 7 and 8 show the recommenders’ trust evolution and η’s success evolution in time. In Figure 7, each recommender is fully expert in one goal and ignorant in the others (\( \bar{d} = 0 \) when he is asked for its goal of expertise and \( \bar{d} = 1 \) otherwise). We observe that every time the user changes its goal, the trust on the recommender which is expert in that goal grows. The trust evolution is slower when the user changes from one goal to the other, since more elements are included in the probability distributions and increments in such distributions require more time. The success evolution of CONSUASOR is better than the eigentrust case. In Figure 8, each recommender is moderately knowledgeable in one goal and ignorant in the others (\( \bar{d} = 0.7 \) when he is asked for its goal of expertise and \( \bar{d} = 1 \) otherwise). As in the previous case, the recommender which is more knowledgeable at every time instant stands above the rest, although the trust measure is diminish since the recommender is not fully expert and sometimes he might not give the best advice. Success evolution of the CONSUASOR strategy is also better than eigentrust in this case.

These results show that CONSUASOR is able to adapt to a different scenario (calculate the most trusted recommender for a given goal) and learn which is the most suitable recommender when a user pursues different goals in time. Experiments where advices are asked first to recommenders, and then selecting the most trusted advice were also tested, with similar results trends.

![Figure 7: Experiment 2. Trust and success in time: \( d_1 = d_2 = 0 \) (good user) and each recommender fully expert (\( d = 0 \)) in a goal](image1)

![Figure 8: Experiment 2. Trust and success in time: \( d_1 = d_2 = 0 \) (good user) and each recommender moderately expert (\( d = 0.7 \)) in a goal](image2)
The adequacy of CONSUASOR to provide reliable trust measures at every time instant results in a higher success rates than in the eigentrust case.

3.5 Experiment 3. Comparison of CONSUASOR strategies

Finally we present different strategies that can be used with CONSUASOR and how these strategies behave in time for a particular context. In these experiments a good user $\eta (d_1 = d_2 = 0)$ has 5 goals that change in time randomly. Recommenders are fully expert in one particular goal and ignorant in the others. In this context, $\eta$ will focus on three specific questions: (1) How good is recommender $\rho$ giving advice $[P_\eta]G$? (2) How good is recommender $\rho$ giving advice for my goal $G$? and (3) How good is recommender $\rho$?

Strategy 1 consists of the following steps: (1) The user asks advice to all recommenders for its goal $G'$; (2) Each recommender provides a recommendation; (3) The user calculates the trust on each advice ($\text{trust}(\rho, [P_\eta]G')$) and selects the most trusted one; and, (4) An experience is generated and stored, the trust of recommenders is updated and the success of the user is updated.

Strategy 2 consists of the following steps: (1) The user calculates the trust on each recommender for its goal $G'$ ($\text{trust}(\rho, [_.]G')$); (2) The user selects the most trusted recommender and asks him advice; (3) The selected recommender provides a recommendation; (4) An experience is generated and stored, the trust of recommenders is updated and the success of the user is updated.

Strategy 3 consists of the following steps: (1) The user calculates the general trust on each recommender $\rho$ ($\text{trust}(\rho, [_.])$); (2) The user selects the most trusted recommender and asks him advice; (3) The selected recommender provides a recommendation; (4) An experience is generated and stored, the trust of recommenders is updated and the success of the user is updated.

Figure 9 shows the success evolution of strategies 1, 2 and 3. We observe that, in this particular context where recommenders are specialized in goals, Strategy 1 is the one that presents a faster learning rate in the first time instants, but eventually Strategy 2 achieves higher levels of success. In this particular context, focusing on good recommender for goals is more effective than focusing on advice, although the latter is more specific. Strategy 3 is not effective, which tell us that to overgeneralize (focusing on good recommenders in general) is not useful in this scenario.

Figure 9: Experiment 3. Success in time, 1 perfect user and 10 recommenders, each fully expert in a goal.
These results stresses the point that the flexibility of the CONSUASOR algorithm in calculating general trust measures is an advantage. However when making such generalizations, decision on what to generalize should be carefully decided, according to the specific context and to the available information.

4 Demonstration

We implemented a website where community members can register, provide advise and share feedback about the experiences resulting of these advices. With the collected data, we will be able to test our model assessing the trustworthiness of the submitted advices.

The website provides the following features:

- **Main page.** This page shows a brief introduction about the website purpose (Figure 10).

  ![Main Page](image)

  **Figure 10: Main Page.**

- **Register and Login.** Users can register and log in with a username and password (Figure 11).

  ![Register and Log in](image)

  **Figure 11: Register and Log in pages.**
• **Plans, Goals and Advisee’s Profile.** Logged users can create, edit and delete plans and goals (Figures 12 and 13). They can also create, edit and delete advisee’s profiles to whom the advices will be recommended to (Figure 14). For instance a plan might be described as ‘Practice piano twice a day’, a goal might be described as ‘Good Piano Performant’ and the advisee profile might be described as ‘Beginner student’.

![Figure 12: Plans.](image12.png)

![Figure 13: Goals.](image13.png)

![Figure 14: Advisee’s profiles.](image14.png)
• **Advices.** Logged users can create, edit or delete advices. Advices are provided to advisees consisting of a plan and a goal (Figure 15 and 16).

![Figure 15: Advices.](image1)

![Figure 16: Create Advice.](image2)

• **Experiences.** Logged users can create, edit or delete experiences about advices. For this, they will select one advise and they will specify what percentage of the advised plan was performed and what percentage of the expected goal was achieved (Figure 17 and 18).

![Figure 17: Experiences.](image3)

As future steps, we plan to use the information gathered in the website to calculate trust measures for advice and provide visual information about their trustworthiness. We will provide a web service to integrate CONSUASOR with the PRAISE platform (such as the Music Circle) that would allow users to make advice and provide feedback on advice.
D3.4 Trust and reputation

FP7-ICT-2011- 8 Collaborative project 318770 PRAISE

(a) First, an advice is selected. (b) Once selected, execution and goal achievement percentages are provided.

Figure 18: Create Experience.

5 Background

The literature on trust is a vast literature with many surveys on the subject (e.g. [21, 23, 16, 27]), where each survey focuses on different dimensions in their classification of existing trust and reputation models. For instance, the survey by Sabater and Sierra [27], the most cited survey, categorises computational trust and reputation models based on six dimensions: (1) paradigm, which describes whether the model follows a cognitive approach or a numerical one; (2) information source, which describes the type of information used, such as direct experiences, witness information, sociological information, and prejudice; (3) visibility, which describes whether a trust measure is a global property that may be viewed by all or a private and subjective property that each individual builds; (4) granularity, which describes whether a model is context dependent or not; (5) cheating behaviour, which describes what type of cheating behaviour is addressed, if any; and (6) type of exchanged information, which describes whether exchanged information is described as boolean or continuous measures.

According to the above classification, CONSUASOR follows a numerical paradigm where trust is defined based on a probabilistic measure, although the model has been designed based on a cognitive model that assumes a good adviser is one who knows about the compliance and honour of advisees as well as the causal relations between plans and goals. Concerning the information source, we say CONSUASOR is based on direct experiences as the model relies on the experiences of a single entity (that which maintains the history of experiences). However, we note that the ‘observers’ that help populate the history of experiences may be trusted third-party entities. Concerning visibility, although a trust measure may be shared, we say the trust measure is subjective as it is local to the history of experiences it relies on. Concerning the granularity, CONSUASOR is context dependent, as it learns from past experiences in similar contexts. Concerning cheating behaviour, CONSUASOR does assume that the adviser or advisee might lie. For example, an advisee may say they will perform a plan (i.e. commit to it) when they are in fact not willing to do so. However, CONSUASOR assumes observers to be truthful. Who to trust as
an observer is a decision left for the entity maintaining the history of experiences (as it has been discussed in Section 2.1.1).

From the large existing literature, the models that may be classified as similar to ours are those that are context dependent, which we focus on next. One of these models is the early model by Marsh [14], which proposed an approach for calculating ‘situational trust’. In this model, the trust that an agent x has on an agent y in a given situation (or context) \( \alpha \) is based on the utility that x gains from situation \( \alpha \), the importance of the situation \( \alpha \) for x, and an estimate of the general trust of x on y that takes into account all possible relevant data in similar past situations. Marsh [14] also has a notion of decay, which is modelled as a time window for experiences.

A more cognitive-based model is that of Castelfranchi and Falcone [4], where trust is viewed as a mental state and a complex attitude where the trust of x on y regarding goal g depends on whether x believes y is both capable and willing of performing g. x’s trust on y regarding goal g also depends on whether g is a goal for x and whether x believes it depends on y in achieving g.

The model by Abdul-Rahman and Hailes [2] makes use of both direct and indirect (or communicated) experiences. The model uses a qualitative degree approach to model trust, taking into account the context as well as the trustworthiness of indirect (communicated) experiences. However, the modelling of uncertainty is somewhat ad-hoc and not based on probabilistic grounds.

In [15], a context-based personalised reputation measure is inferred from propagated ratings through a peer-to-peer network. The model is based on a Bayesian probabilistic framework.

The Regret model [26] also calculates trust based on learning from similar past experiences. Both direct and indirect experiences are considered, and an incorporated credibility model is used to assess the reputation and trust of an information provider, based on social network analysis and prejudices. However, the overall notion of trust does not have a probabilistic meaning and is based on a utility modelling of the interactions that we depart from (see [6] for a discussion).

Like Regret, the FIRE model [10] uses a number of information sources to calculate and integrate four different types of trust and reputation measures: (1) interaction trust, which is based on learning from the similar past experience of direct interactions; (2) role-based trust, which is a trust measure that is influenced by role-based relationships between the agents; (3) witness reputation, which is based on learning from the similar past experience of indirect interactions; and (4) certified reputation, which is built from third-party references provided by the agent itself (this is similar to the recommendation letters used when applying for a job position).

Ramchurn et al. [24] also based reputation on similarity between new contexts and past ones. However, their approach uses the concept of fuzzy sets to compute one’s confidence, based on the notion of assigning utilities to the different aspects of a context. Trust is then built on the concept of the maximum expected loss in utility.

The approach by Simari et al. [30] compares what the agent has committed to to what is actually delivered (which we refer to as observed). Similar to us, trust is then context dependent and based on past performances, although our similarity measures are more general measures that are based on semantic matching and empowerment.

Tavakolifard et al. [31] illustrate how trust information in one context can affect trust information in other contexts. The model uses case-base reasoning to assess trustworthiness in new situations by relying on past similar experiences.
In the domain of recommendations, Nakatsuji et al. [17] illustrate how recommendations may be made across multiple interrelated domains such as music and movies. In this model, similarity is based on comparing users who share (and rate) similar items or who share social connections. Such a similarity measure can then help provide recommendation chains (sequences of transitively associated edges) to items in other domains.

Finally, as illustrated by the introduction of Section 2, Osman et al. [18] uses the same philosophy as this part of the paper. In [18], one compares what has been committed to to what is actually delivered (or observed), and the model is based on probability and information theory (for updating the probabilities of expectations) and semantic similarity (for comparing contexts). The equations on semantic similarity and information decay have first been introduced in [18], whereas the notion of updating a probability distribution following the minimum relative entropy approach has first been introduced in [28].

However, differently from all the above models, CONSUASOR takes into account the specifics of an advice. For computing trust on advice, the model does not treat advice as a single entity, but assesses each of its components: the recommended plan and the intended goal. Carefully analysing an advice’s constituents is what makes this model stand out from the rest. As a result, we say an adviser is a good adviser if the adviser takes into consideration the compliance and honour of the advisee when making his advice, in addition to the causality relations between plans and goals. Additionally, CONSUASOR introduces the notion of empowerment for comparing the similarity of contexts when the capability of performing actions (or plans) is important.

6 Conclusions and Future Work

A good adviser needs to certainly know the causal relationship between performing a plan and realising a goal. However, when suggesting a plan to someone, the adviser also needs to know how complaint and honourable that person is, in order to personalise his advice accordingly. An adviser may need to be more demanding to those who are less complaint or honourable, for instance, in order to guarantee that the target plan is actually performed. Currently existing trust algorithms do not take into account the specifics of plan recommendation. We have proposed a trust model for advice based on the compliance and honour of advisees, as well as the causality between plans and goals. All these models are learned from similar past experiences using probability and information theory. The similarity of experiences is based on semantic similarity and the notion of empowerment, which this part of the paper introduces to help compute the similarity of actions (or plans) when the capability of performing these actions (or plans) is important. Finally, we have proposed a benchmark for the evaluation of trust algorithms on advisers that may be used in the future to compare with other trust algorithms modelling plan recommendations when they become available. Empirical evaluation shows that our model assesses coherent trust measures and reach high levels of success in short periods of time. We leave for future work a more detailed analysis of the sensitivity of the algorithm’s parameters and further experiments on complex scenarios.

Plan for Year 3. Our plan for the following year is to consider extending the implementation to allow us to learn from of other people’s experiences, where the trustworthiness/relevance of other people’s experiences will depend on their social relations.
We will continue to improve the online website (see Section 4). This website will be open to the public, allowing users to provide advice and provide feedback on advice. We will also build a web service that would integrate our work with the PRAISE platform. The Consuasor model can then be used to assess and rank collected advice and advisers.

We will also approach existing companies that might benefit from the concept of assessing the trustworthiness of advice in order to see whether any of those are interested in integrating/using the Consuasor trust model. Specifically, we will approach a wiki-based community (wikihow.com) that has an extensive database of how-to guides. We will also approach a startup company (Cognicor) that provides an automated complaint resolution. We will also approach a startup company (wwwwww.me) providing a platform that allows its users to discuss fashion and create and publish shippable looks to inspire others. Approaching these companies allows us to introduce this work to the industry.
Part II

Collaborative Assessment

In this part of the paper we introduce an automated assessment service for online learning support in the context of communities of learners. The goal is to introduce automatic tools to support the task of assessing massive number of students as needed in Massive Open Online Courses (MOOC). The final assessments are a combination of tutor’s assessment and peer assessment. We build a trust graph over the referees and use it to compute weights for the assessments aggregations. The model proposed intends to be a support for intelligent online learning applications that encourage student’s interactions within communities of learners and benefits from their feedback to build trust measures and provide automatic marks.

7 Introduction

Self and peer assessment have clear pedagogical advantages. Students increase their responsibility and autonomy, get a deeper understanding of the subject, become more active in the learning process, reflect on their role in group learning, and improve their judgement skills. Also, it may have the positive side effect of reducing the marking load of tutors. This is specially critical when tutors face the challenge of marking large quantities of students as needed in the increasingly popular Massive Open Online Courses (MOOC).

Online learning communities encourage different types of peer-to-peer interactions along the learning process. These interactions permit students to get more feedback, to be more motivated to improve, and to compare their own work with other students accomplishments. Tutors, on the other hand, benefit from these interactions as they get a clearer perception of the student engagement and learning process.

Previous works have proposed different methods of peer assessment as part of the learning process with the added advantage of helping tutors in the sometimes daunting task of marking large quantities of students [20, 5].

The authors of [20] propose methods to estimate peer reliability and correct peer biases. They present results over real world data from 63,000 peer assessments of two Coursera courses. The models proposed are probabilistic and they are compared to the grade estimation algorithm used on Coursera’s platform, which does not take into account individual biases and reliabilities. Differently from them, we place more trust in students who grade like the tutor and do not consider student’s biases. When a student is biased its trust measure will be very low and his/her opinion will have a moderate impact over the final marks.

[5] proposes the CrowdGrader framework, which defines a crowdsourcing algorithm for peer evaluation. The accuracy degree (i.e. reputation) of each student is measured as the distance between his/her self assessment and the aggregated opinion of the peers weighted by their accuracy degrees. The algorithm thus implements a reputation system for students, where higher accuracy leads to higher influence on the consensus grades. Differently from this work, we give more weight to those peers that have similar opinions to those of the tutor.
In this part of the paper, and differently from previous works, we want to study the reliability of student assessments when compared with tutor assessments. Although part of the learning process is that students participate in the definition of the evaluation criteria, tutors want to be certain that the scoring of the students’ works is fair and as close as possible to his/her expert opinion.

PRAISE enables online virtual communities of students with shared interests and goals to come together and share their music practice with each other so the process of learning becomes social. We will provide tools for giving and receiving feedback, as feedback is considered an essential part of the learning process. In an online classroom, tutors define lesson plans as pedagogical workflows of activities, such as uploading recorded songs, automatic performance analysis, peer feedback, or reflexive pedagogy analysis. The goal of any lesson plan is to improve student skills, for instance, the performance speed competence or the interpretation maturity level. Assessments of students’ performances have to evaluate the achievement of these skills. Once a lesson plan is defined, students can navigate through the activities, to upload assignments, to practice, to assess each other, and so on. And tutors can monitor what students have done and assess them accordingly. In this work we concentrate on the development of a service that can be included as part of a lesson plan and helps tutors in the overall task of assessing the students participating in the lesson plan. This assessment is based on aggregating students’ assessments, taking into consideration the trust that tutors have on the students’ individual capabilities in judging each others work.

To achieve our objective we propose in this part of the paper an automated assessment method (Section 8) based on tutor assessments, aggregations of peer assessments and on trust measures derived from peer interactions. We experimentally evaluate (Section 9) the accuracy of the method over different topologies of student interactions (i.e. different types of student grouping). The results obtained are based on simulated data, leaving the validation with real data for future work. We then conclude with a discussion of the results (Section 10).

8 Collaborative Assessment

In this section we introduce the formal model of the method and the algorithms for collaborative assessment.

8.1 Notation and preliminaries

We say an online course has a tutor \( \tau \), a set of peer students \( \mathcal{S} \), and a set of assignments \( \mathcal{A} \) that need to be marked by the tutor and/or students with respect to a given set of criteria \( \mathcal{C} \).

The automated assessment state \( S \) is then defined as the tuple:

\[
S = \langle R, \mathcal{A}, \mathcal{C}, \mathcal{L} \rangle
\]

\( R = \{ \tau \} \cup \mathcal{S} \) defines the set of possible referees (or markers), where a referee could either be the tutor \( \tau \) or some student \( s \in \mathcal{S} \). \( \mathcal{A} \) is the set of submitted assignments that need to be marked and \( \mathcal{C} = \{ c_1, \ldots, c_n \} \) is the set of criteria that assignments are marked upon. \( \mathcal{L} \) is the set of marks (or assessments) made by referees, such that \( \mathcal{L} : R \times \mathcal{A} \to [0, \lambda]^n \) (we assume marks to be real numbers between 0 and some maximum value \( \lambda \)). In other words, we define a single assessment
as: $\mu^\rho = \vec{M}$, where $\alpha \in \mathcal{A}$, $\rho \in \mathbb{R}$, and $\vec{M} = (m_1, \ldots, m_n)$ describes the marks provided by the referee on the $n$ criteria of $\mathcal{C}$, $m_i \in [0, \lambda]$.

**Similarity between marks** We define a similarity function $\text{sim} : [0, \lambda]^n \times [0, \lambda]^n \to [0, 1]$ to determine how close two assessments $\mu^\rho$ and $\mu^\eta$ are. We calculate the similarity between assessments $\mu^\rho = \{m_1, \ldots, m_n\}$ and $\mu^\eta = \{m'_1, \ldots, m'_n\}$ as follows:

$$\text{sim}(\mu^\rho, \mu^\eta) = 1 - \frac{\sum_{i=1}^{n} |m_i - m'_i|}{n \sum_{i=1}^{n} \lambda}$$

This measure satisfies the basic properties of a fuzzy similarity \cite{9}. Other similarity measures could be used.

**Trust relations between referees** Tutors need to decide up to which point they can believe on the assessments made by peers. We use two different intuitions to make up this belief. First, if the tutor and the student have both assessed some assignments, their similarity gives a hint of how close the judgements of the student and the tutor are. Similarly, we can define the judgement closeness of any two students by looking into the assignments evaluated by both of them. In case there are no assignments evaluated by the tutor and one particular student we could simply not take that student’s opinion into account because the tutor would not know how much to trust the judgement of this student, or, as we do in this part of the paper, we approximate that unknown trust by looking into the chain of trust between the tutor and the student through other students. To model this we define two different types of trust relations:

- **Direct trust**: This is the trust between referees $\rho, \eta \in \mathbb{R}$ that have at least one assignment assessed in common. The trust value is the average of similarities on the assessments over the same peers. Let the set $A_{\rho, \eta}$ be the set of all assignments that have been assessed by both referees. That is, $A_{\rho, \eta} = \{\alpha \rightarrow \mu^\rho_\alpha \in \mathcal{L} \text{ and } \mu^\eta_\alpha \in \mathcal{L}\}$. Then,

$$T_D(\rho, \eta) = \frac{\sum_{\alpha \in A_{\rho, \eta}} \text{sim}(\mu^\rho_\alpha, \mu^\eta_\alpha)}{|A_{\rho, \eta}|}$$

We could also define direct trust as the conjunction of the similarities for all common assignments as:

$$T_D(\rho, \eta) = \bigwedge_{\alpha \in A_{\rho, \eta}} \text{sim}(\mu^\rho_\alpha, \mu^\eta_\alpha)$$

However, this would not be practical, as a significant difference in just one assessment of those assessed by two referees would make their mutual trust very low.

- **Indirect trust**: This is the trust between referees $\rho, \eta \in \mathbb{R}$ without any assignment assessed by both of them. We compute this trust as a transitive measure over chains of referees for
which we have pair-wise direct trust values. We define a trust chain as a sequence of referees $q_j = \langle \rho_i, \ldots, \rho_i+1, \ldots, \rho_m \rangle$ where $\rho_i \in R$, $\rho_1 = \rho$ and $\rho_{m_j} = \eta$ and $T_D(\rho_i, \rho_{i+1})$ is defined for all pairs $(\rho_i, \rho_{i+1})$ with $i \in [1, m_j - 1]$. We note by $Q(\rho, \eta)$ the set of all trust chains between $\rho$ and $\eta$. Thus, indirect trust is defined as an aggregation of the direct trust values over these chains as follows:

$$T_I(\rho, \eta) = \max_{q_j \in Q(\rho, \eta)} \prod_{i \in [1, m_j-1]} T_D(\rho_i, \rho_{i+1})$$

Hence, indirect trust is based on the notion of transitivity. Ideally, we would like to not overrate the trust of a tutor on a student, that is, we would like that $T_D(a, b) \geq T_I(a, b)$ in all cases. Guaranteeing this in all cases is impossible, but we can decrease the number of overtrusted students by selecting an operator that gives low values to $T_I$. In particular, we prefer to use the product $\prod$ operator, because this is the $t$-norm that gives the smallest possible values. Other operators could be used, for instance the $\min$ function.

**Trust Graph** To provide automated assessments, our proposed method aggregates the assessments on a given assignment taking into consideration how much trusted is each marker/referee from the point of view of the tutor (i.e. taking into consideration the trust of the tutor on the referee in marking assignments). The algorithm that computes the student final assessment is based on a graph defined as follows:

$$G = \langle R, E, w \rangle$$

where the set of nodes $R$ is the set of referees in $S$, $E \subseteq R \times R$ are edges between referees with direct or indirect trust relations, and $w : E \rightarrow [0, 1]$ provides the trust value. We note by $D \subset E$ the set of edges that link referees with direct trust. That is, $D = \{ e \in E | T_D(e) \neq \bot \}$. An similarly, $I \subset E$ for indirect trust, $I = \{ e \in E | T_I(e) \neq \bot \} \setminus D$. The $w$ values will be used as weights to combine peer assessments and are defined as:

$$w(e) = \begin{cases} T_D(e) & \text{if } e \in D \\ T_I(e) & \text{if } e \in I \end{cases}$$

Figure [19] shows examples of trust graphs with $e \in D$ (in black) and $e \in I$ (in red — light gray) for different sets of assessments $\mathcal{L}_e$.

### 8.2 Computing collaborative assessments

Algorithm [2] implements the collaborative assessment method. We keep the notation $(\rho, \eta)$ to refer to the edge connecting nodes $\rho$ and $\eta$ in the trust graph and $Q(\rho, \eta)$ to refer the set of trust chains between $\rho$ and $\eta$.

$T_I$ is based on a fuzzy-based similarity relation $\sim$ presented before and fulfilling the $\otimes$-Transitivity property: $\sim(u, v) \otimes \sim(v, w) \leq \sim(u, w)$, $\forall u, v, w \in V$, where $\otimes$ is a $t$-norm [9].
Algorithm 2 collaborativeAssessments($S = (R, \mathcal{A}, \mathcal{C}, \mathcal{L})$)

Require: $S = (R, \mathcal{A}, \mathcal{C}, \mathcal{L})$ is the automated assessment state, where $R$ is the set of all nodes (referees), $\mathcal{A}$ is the set of submitted assignments that need to be marked, $\mathcal{C}$ is the set of criteria that assignments are marked upon, and $\mathcal{L} = \{\mu_\eta, \ldots\}$ is the set of marks made by referees (where $\mu_\eta$ describes the assessment that $\eta$ made on assignment $\alpha$)

Require: $T_D : R \times R \rightarrow [0, 1]$ is the direct trust measure between two nodes in $R$

Require: $T_I : R \times R \rightarrow [0, 1]$ is the indirect trust measure between two nodes in $R$

Require: $w : R \times R \rightarrow [0, 1]$ is the trust value between two nodes in $R$

Require: $D = \{e \in E | T_D(e) \neq \perp\}$ is the set of edges that link nodes with direct trust

Require: $I = \{e \in E | T_I(e) \neq \perp\} \backslash D$ is the set of edges that link nodes with indirect trust

Require: $Q : R \times R \rightarrow 2^R$ is the set of trust chains between two nodes in $R$

\[\triangleright\text{Initial trust between referees is zero}\]

\[D = I = \emptyset\]

\[\text{for all } \rho, \eta \in R, \rho \neq \eta \text{ do}\]

\[w(\rho, \eta) = 0\]

\[\text{end for}\]

\[\triangleright\text{Update direct trust and edges}\]

\[\text{for all } \rho, \eta \in R, \rho \neq \eta \text{ do}\]

\[A_{\rho, \eta} = \{\beta | \mu_\rho \beta \in \mathcal{L} \text{ and } \mu_\eta \beta \in \mathcal{L}\}\]

\[\text{if } |A_{\rho, \eta}| > 0 \text{ then}\]

\[D = D \cup (\rho, \eta)\]

\[w(\rho, \eta) = T_D(\rho, \eta)\]

\[\text{end if}\]

\[\text{end for}\]

\[\triangleright\text{Update indirect trust and edges between tutor & students}\]

\[\text{for all } \rho \in R \text{ do}\]

\[\text{if } (\tau, \rho) \notin D \text{ and } Q(\tau, \rho) \neq \emptyset \text{ then}\]

\[I = I \cup (\rho, \eta)\]

\[w(\rho, \eta) = T_I(\tau, \eta)\]

\[\text{end if}\]

\[\text{end for}\]

\[\triangleright\text{Calculate automated assessments}\]

\[\text{assessments} = \emptyset\]

\[\text{for all } \alpha \in \mathcal{A} \text{ do}\]

\[\text{if } \mu_\alpha \in \mathcal{L} \text{ then}\]

\[\triangleright\text{Tutor assessments are preserved}\]

\[\text{assessments} = \text{assessments} \cup (\alpha, \mu_\alpha)\]

\[\text{else}\]

\[\triangleright\text{Generate automated assessments}\]

\[R' = \{\rho | \mu_\rho \in \mathcal{L}\}\]

\[\text{if } |R'| > 0 \text{ then}\]

\[\mu_\alpha = \frac{\sum_{\rho \in R} \mu_\rho \rho \ast w(\tau, \rho)}{\sum_{\rho \in R} w(\tau, \rho)}\]

\[\text{assessments} = \text{assessments} \cup (\alpha, \mu_\alpha)\]

\[\text{end if}\]

\[\text{end if}\]

\[\text{end for}\]

\[\text{return assessments}\]

The first thing the algorithm does is to build a trust graph from $\mathcal{L}$. Then, the final assessments are computed as follows. If the tutor marks an assignment, then the tutor mark is considered the final mark. Otherwise, a weighted average $(\mu_\alpha)$ of the marks of student peers is calculated for this assignment, where the weight of each peer is the trust value between the tutor and that peer. Other
forms of aggregation could be considered to calculate $\mu_\alpha$, for instance a peer assessment may be discarded if it is very far from the rest of assessments, or if the referee’s trust falls below a certain threshold.

Figure 19 shows four trust graphs built from four assessments histories that corresponds to a chronological sequence of assessments made. The criteria $\mathcal{C}$ in this example are speed and maturity and the maximum mark value is $\lambda = 10$. For simplicity we only represent those referees that have made assessments in $\mathcal{L}$. In Figure 19(a) there is one node representing the tutor who has made the only assessment over the assignment $ex_1$ and there are no links to other nodes as no one else has assessed anything. In (b) student Dave assesses the same exercise as the tutor and thus a link is created between them. The trust value $w(\text{tutor, Dave}) = T_D(\text{tutor, Dave})$ is high since their marks were similar. In (c) a new assessment by Dave is added to $\mathcal{L}$ with no consequences in the graph construction. In (d) student Patricia adds an assessment on $ex_2$ that allows to build a direct trust between Dave and Patricia and an indirect trust between the tutor and Patricia, through Dave. The automated assessments generated in case (d) are: $(5, 5)$ for exercise 1 (which preserves the tutor’s assessment) and $(3, 7, 3.7)$ for exercise 2 (which uses a weighted aggregation of the peers’ assessments).

Note that the trust graph built from $\mathcal{L}$ is not necessarily connected. A tutor wants to reach a point in which the graph is totally connected because that means that the collaborative assessment algorithm generates an assessment for every assignment. Figure 20 shows an example of a trust graph of a particular learning community involving 50 peer students and a tutor. When $S$ has a history of 5 tutor assessments and 25 student assessments ($|\mathcal{L}| = 30$) we observe that not all nodes are connected. As the number of assessments increases, the trust graph becomes denser and eventually it gets completely connected. In (b) and (c) we see a complete graph.

Figure 19: Trust graph example 1.
9 Experimental Platform and Evaluation

In this Section we describe how we generate simulated social networks, describe our experimental platform, define our benchmarks and discuss experimental results.

9.1 Social Network Generation

Several models for social network generation have been proposed reflecting different characteristics present in real social communities. Topological and structural features of such networks have been explored in order to understand which generating model resembles best the structure of real communities [8].

A social network can be defined as a graph $\mathcal{G}$ where the set of nodes represent the individuals of the network and the set of edges represent connections or social ties among those individu-als. In our case, individuals are the members of the learning community: the tutor and students. Connections represent the social ties and they are usually the result of interactions in the learning community. For instance a social relation will be born between two students if they interact with each other, say by collaboratively working on a project together. In our experimentation, we rely on the social network in order to simulate which student will assess the assignment of which other student. We assume students will assess the assignments of students they know, as opposed to picking random assignments. As such, we clarify that social networks are different from the trust graph of Section 8. While the nodes of both graphs are the same, edges in the social network represent social ties, whereas edges in the trust graph represent how much does one referee trust another in judging others work.

To model social networks where relations represent social ties, we follow three different approaches: the Erdős-Rényi model for random networks [7], the Barabási-Albert model for power law networks [3] and a hierarchical model for cluster networks.

9.1.1 Random Networks

The Erdős-Rényi model for random networks consists of a graph containing $n$ nodes connected randomly. Each possible edge between two vertices may be included in the graph with probability
$p$ and may not be included with probability $(1 - p)$. In addition, in our case there is always an edge between the node representing the tutor and the rest of nodes, as the tutor knows all of its students (and may eventually mark any of those students).

The degree distribution of random graphs follows a Poisson distribution. Figure 21(a) shows an example of a random graph with 51 nodes and $p = 0.5$ and its degree distribution. Note that the point with degree 50 represents the tutor node while the rest of the nodes degree fit a Poisson distribution.

### 9.1.2 Power Law Networks

The Barabási-Albert model for power law networks base their graph generation on the notions of *growth* and *preferential attachment*. The generation scheme is as follows. Nodes are added one at a time. Starting with a small number of initial nodes, at each time step we add a new node with $m$ edges linked to nodes already part of the network. In our experiments, we start with $m + 1$ initial

![Degree Distribution](a) Random Network (aprox graph density 0.5)

![Degree Distribution](b) Power Law Network (aprox graph density 0.5)

![Degree Distribution](c) Cluster Network (aprox graph density 0.2)

Figure 21: Social Network generation examples
nodes. The edges are not placed uniformly at random but preferentially in proportion to the degree of the network nodes. The probability $p$ that the new node is connected to a node $i$ already in the network depends on the degree $k_i$ of node $i$, such that: $p = k_i / \sum_{j=1}^{n} k_j$. As above, there is also always an edge between the node representing the tutor and the rest of the nodes.

The degree distribution of this network follows a Power Law distribution. Figure 21(b) shows an example of a power law graph with 51 nodes and $m = 16$ and its degree distribution. The point with degree 50 describes the tutor node while the rest of the nodes closely resemble a power law distribution. Recent empirical results on large real-world networks often show, among other features, their degree distribution following a power law [8].

9.1.3 Cluster Networks

As our focus is on learning communities, we also experiment with a third type of social network: the cluster network which is based on the notions of groups and hierarchy. Such networks consists of a graph composed of a number of fully connected clusters (where we believe clusters may represent classrooms or similar pedagogical entities). Additionally, as above, all the nodes are connected with the tutor node. Figure 21(c) shows an example of a cluster graph with 51 nodes, 5 clusters of 10 nodes each and its degree distribution. The point with degree 50 describes the tutor while the rest of the nodes have degree 10, since every student is fully connected with the rest of the classroom.

9.2 Experimental Platform

In our experimentation, given an initial automated assessment state $S = \langle R, A, C, L \rangle$ with an empty set of assessments $L = \{\}$, we want to simulate tutor and peer assessments so that the collaborative assessment method can eventually generate a reliable and definitive set of assessments for all assignments.

To simulate assessments, we say each students is defined by its profile that describes how good its assessments are. The profile is essentially defined by the measure, or distance, $d_\rho \in [0, 1]$ that specifies how close are the student’s assessments to that of the tutor.

We then assume the simulator knows how the tutor and each student would assess an assignment. This becomes necessary in our simulation, since we generate student assessments in terms of their distance to that of the tutor’s, even if the tutor does not choose to actually assess the assignment in question. This simulator’s knowledge of the values of all possible assessments is generated accordingly:

- For every assignment $\alpha \in A$, we calculate the tutor’s assessment, which is randomly generated according to the function $f_T : A \rightarrow [0, \lambda]^n$. This assessment essentially describes what mark would the tutor give $\alpha$, if it decided to assess it.

- For every assignment $\alpha \in A$, we also calculate the assessment of each student $\rho \in S$. This is calculated according to the function $f_\rho : A \rightarrow [0, \lambda]^n$, such that: $\text{sim}(f_\rho(\alpha), f_T(\alpha)) \geq d_\rho$.

We note that we only need to calculate $\rho$’s assessment of $\alpha$ if the student who submitted the assignment $\alpha$ is a neighbour of $\rho$ in $N$.
We note that the above only calculates what the assessments would be, if referees where to assess assignments.

9.3 Benchmark

Given an initial automated assessment state $S = \langle R, A, \mathcal{C}, \mathcal{L} \rangle$ with an empty set of assessments $\mathcal{L} = \{\}$, a set of student profiles $Pr = \{d_s\}_{s \in R}$, and a social network $\mathcal{N}$ (whose nodes is the set $R$), we simulate individual tutor and students’ assessments. When does a referee in $R$ assess an assignment in $A$ is explained shortly. However we note here that the value of each generated assessment is equivalent to that calculated for the simulator’s knowledge (see Section 9.2 above).

In our benchmark, we consider the three types of social networks introduced earlier: random social networks (with 51 nodes, $p = 0.5$, and approximate density of 0.5), power law networks (with 51 nodes, $m = 16$, and approximate density of 0.5), and cluster networks (with 51 nodes, 5 clusters of 10 nodes each, and approximate density of 0.2). Examples of these generated networks are shown in Figure 21.

We say one assignment is submitted by each student, resulting in $|S| = 50$ and $|A| = 50$. The range that a referee (tutor or student) may mark a given assignment with respect to a given criteria is [0,10]. And the set of criteria is $\mathcal{C} = \langle speed, maturity \rangle$. The criteria essentially measure the speed of playing a musical piece, and the maturity level of the student’s performance.

An assessment profile is generated for each student $\rho$ at the beginning of the execution, resulting in a set of student profiles $Pr = \{d_s\}_{s \in S}$, where $d \in [0,0.5]$. We consider here two cases for generating the set of student profiles $Pr$. A first case where $d$ is picked randomly following a power law distribution (Figure 22(a)) and a second case where $d$ is picked randomly following a uniform distribution (Figure 22(b)).

With simulated individual assessments, we then run the collaborative assessment method in order to compute an automated assessment. We also compute the ‘error’ of the collaborative assessment method, whose range is $[0,1]$, over the set of assignments $A$ accordingly:

$$\sum_{\alpha \in \mathcal{A}} \frac{\text{sim}(f_T(\alpha), \phi(\alpha))}{|\mathcal{A}|},$$

where $\phi(\alpha)$ describes the automated assessment for a given assignment $\alpha \in \mathcal{A}$

Figure 22: Example of the profile distributions (left) and of $d$ counting averaged over 50 instances (right)
With the settings presented above, we run two different experiments. The results presented are an average over 50 executions. The two experiments are presented next.

In experiment 1, students provide their assessments before the tutor. Each student $\rho$ provides assessments for a randomly chosen $a_\rho$ number of peer assignments (of course, where assignments are those of their neighboring peers in $\mathcal{N}$). We run the experiment for 5 different values of $a_\rho = \{3, 4, 5, 6, 7\}$. After the students provide their assessments, the tutor starts assessing assignments incrementally. After every tutor assessment, the error over the set of automated assessment is calculated. Notice that the collaborative assessment method takes the tutor assessment, when it exists, to be the final assessment. As such, the number of automated assessments calculated based on aggregating students’ assessments is reduced over time. Finally, when the tutor has assessed all 50 students, the resulting error is 0.

In experiment 2, the tutor provides its assessments before the students. The tutor in this experiment will assess a randomly chosen number of assignments, where this number is based on the percentage $a_\tau$ of the total number of assignments. We run the experiment for 4 different values of $a_\tau = \{5, 10, 15, 20\}$. After the tutor provides their assessments, students’ assessments are performed. In every iteration, a student $\rho$ randomly selects a neighbor in $\mathcal{N}$ and assesses his assignment (in case it has not been assessed before by $\rho$, otherwise another connected peer is chosen). We note that in the case of random and power law networks (denser networks), a total number of 1000 student assessments are performed. Whereas in the case of cluster networks (looser network), a total of 400 student assessments are performed. We note that initially, the trust graph is not fully connected, so the service is not able to provide automated assessments for all assignments. When the graph gets fully connected, the service generates automated assessments for all assignments and we start measuring the error after every new iteration.

9.4 Evaluation

In experiment 1, we observe (Figure 23) that the error decreases when the number of tutor assessments increase, as expected, until it reaches 0 when the tutor has assessed all 50 students. This decrement is quite stable and we do not observe abrupt error variations or important error increments from one iteration to the next. More variations are observed in the initial iterations since the service has only a few assessments to deduce the weights of the trust graph and to calculate the final outcome.

In the case of experiment 2 (Figure 24), the error diminishes slowly as the number of student assessments increase, although it never reaches 0. Since the number of tutor assessments is fixed in this experiment, we have an error threshold (a lower bound) which is linked to the students’ assessment profile: the closest to the tutor’s the lower this threshold will be. In fact, in both experiments we observe that when using a power law distribution profile (Figure 22(a)) the automated assessment error is lower than when using a uniform distribution profile (Figure 22(b)). This is because when using a power law distribution, more student profiles are generated whose assessments are closer to the tutors’.

In general, the error trend observed in all experiments comparing different social network scenarios (random, cluster or power law) show a similar behavior. Taking a closer look at experiment 2, cluster social graphs have the lowest error and we observe that assessments on all assignments are achieved earlier (this is, the trust graph gets connected earlier). We attribute this to the topology
Figure 23: Experiment 1

Figure 24: Experiment 2
of the fully connected clusters which favors the generations of indirect edges earlier in the graph between the tutor and the nodes of each cluster. Power law social graphs have lower error than random networks in most cases. This can be attributed to the criteria of preferential attachment in their network generation, which favors the creation of some highly connected nodes. Such nodes are likely to be assessed more frequently since more peers are connected to them. Then, the automated assessments of these highly connected peers are performed with more available information which could lead to more accurate outcomes.

10 Conclusion and Future Work

The collaborative assessment model proposed in this part of the paper is thought of as a support in the creation of intelligent online learning applications that encourage student interactions within communities of learners. It goes beyond current tutor-student online learning tools by making students participate in the learning process of the whole group, providing mutual assessment and making the overall learning process much more collaborative.

The use of AI techniques is key for the future of online learning communities. The application presented in this part of the paper is specially useful in the context of MOOC: with a low number of tutor assessments and encouraging students to interact and provide assessments among each other, direct and indirect trust measures can be calculated among peers and automated assessments can be generated.

Several error indicators can be designed and displayed to the tutor managing the course, which we leave for future work. For example the error indicators may inform the tutor which assignments have not received any assessments yet, or which deduced marks are considered unreliable. For example, a deduced mark on a given assignment may be considered unreliable if all the peer assessments that have been provided for that assignment are considered not to be trusted by the tutor as they fall below a preselected acceptable trust threshold. Alternatively, a reliability measure may also be assigned to the computed trust measure $T_D$. For instance, if there is only one assignment that has been assessed by $\tau$ and $\rho$, then the computed $T_D(\tau, \rho)$ will not be as reliable as having a number of assignments assessed by $\tau$ and $\rho$. As such, some reliability threshold may be used that defines what is the minimum number of assignments that both $\tau$ and $\rho$ need to assess for $T_D(\tau, \rho)$ to be considered reliable. Observing such error indicators, the tutor can decide to assess more assignments and as a result the error may improve or the set of deduced assessments may increase. Finally, if the error reaches a level of acceptance, the tutor can decide to endorse and publish the marks generated by the collaborative assessment method.

Another interesting question for future work is presented next. Missing connections might be detected in the trust graph that would improve its connectivity or maximize the number of direct edges. The question that follows then is, what assignments should be suggested to which peers such that the trust graph and the overall assessment outcome would improve?

Additionally, future work may also study different approaches for calculating the indirect trust value between two referees. In this part of the paper, we use the product operator. We suggest to study a number of operators, and run an experiment to test which is most suitable. To do such a test, we may calculate the indirect trust values for edges that do have a direct trust measure, and then see which approach for calculating indirect trust gets closest to the direct trust measures.
**Plan for Year 3.** Our plan for the following year is, first and foremost, to integrate the collaborative assessment module with the PRAISE platform. Concerning further research, we will also investigate and implement a “recommendation module” (and integrate it with the collaborative assessment module), which will allow the collaborative assessment model not just to compute the trustworthiness of opinionators, but also recommend who should assess what next, in order to minimise the error of automated assessments.

We also plan to evaluate the model using real data from a real-life use case. We plan to collect data from an English classroom at the “Institut Torras i Bages” in Barcelona, where students will be assessing each others work, and based on those assessments, the Collaborative Assessment will compute an automated deduced assessment for those assignments that the tutor has not yet assessed. We will also contact other schools at the Industry day that will take place in Barcelona in September 2014 in order to investigate whether we can test our model in other classrooms.

We will also investigate the possibility of integrating the model with other existing online learning platforms. For instance, we will approach Kaplan, Inc. (a provider of higher education programs, professional training courses, test preparation materials and other services for various levels of education) who are planning to launch a new online learning platform by the end of 2014.

**References**


