

Towards an Inductive Algorithm for Learning Trust Alignment

Andrew Koster
Artificial Intelligence Research
Institute, CSIC
Bellaterra, Spain
andrew@iia.csic.es

Jordi Sabater-Mir
Artificial Intelligence Research
Institute, CSIC
Bellaterra, Spain
jsabater@iia.csic.es

Marco Schorlemmer
Artificial Intelligence Research
Institute, CSIC
Bellaterra, Spain
marco@iia.csic.es

ABSTRACT

Knowing which agents to trust is an important problem in open multi-agent systems. A way to help solve this problem is by allowing agents to relay information about trust to each other. We argue trust is a subjective phenomenon and therefore needs aligning. We present a mathematical framework for communicating about trust in terms of interactions. Based on this framework we present an algorithm based on clustering and inductive logic programming techniques to align agents' trust models.

Keywords

inductive logic programming, trust, alignment, learning

1. INTRODUCTION

In complex, distributed systems, such as multi-agent systems, the artificial entities have to cooperate, negotiate, compete, etc. amongst themselves. Thus the social aspect of these systems plays a crucial role in their functioning. One of the issues in such a social system is the question of whom to trust and how to find this out. There are several systems already in development that model trust and reputation [16], ranging from a straightforward listing of evaluations (such as eBay's [13] reputation system), to complex cognitive models (such as Repage [18]). We anticipate that in an open multi-agent system, there will be a large diversity of models in concurrent use by different agents, depending on the wishes of the programmer and the user. However, even if there is consensus on some model, this is still only a consensus on the computational representation. In a heterogeneous environment it is inevitable that, if the trust model an agent uses is based on cognitive principles, the way different agents interpret their environment will still lead to differences in trust. We will show how, despite agreeing on the ontological underpinnings of the concepts, there is the need to align trust so as to enable reliable *gossip*. With gossip we refer to all communication about trust.

We will emphasize the need to align trust further by considering a simple example of a multi-agent system with three agents.

Alice wants to know if Dave would be a good keynote speaker

for the conference she is organizing. However, she does not know enough about him. She asks Bob. Bob has never collaborated with Dave directly, but they work at the same institute and play squash together. Through these interactions, Bob has trust in Dave and tells this to Alice.

Lets analyse Bob's model. He does not know Dave professionally and bases his trust in Dave on personal interactions. This is a perfectly valid model, but lets assume Alice's model works differently: she only takes academic accomplishments into account. She should therefore disregard Bob's gossip, because it is based on, what she considers, *unreliable* information. We emphasize that we differentiate between the trust she has in Bob and the reliability of the information he sends her. Her trust in Bob is grounded in her trust and reputation model. However, what we want to find out is whether the gossip Bob sends can be interpreted reliably in Alice's model.

This short example shows that even in simple situations the concepts related to trust are highly personal and communication about them is no straightforward matter. In the case that two agents wish to exchange information about trust it is therefore important to clarify what trust means to each of them. This can be done in an alignment process, based on similar protocols in ontology alignment, concept formation and other related fields. Some work has been done in defining common ontologies for trust [14, 7], however in practice these ontologies do not have the support of many of the different trust methodologies in development. Even if support were added for all systems and a common ontology emerged, we could still not use it to communicate effectively. Trust is an inherently personal phenomenon and has subjective components which cannot be captured in a shared ontology. An adaptable approach that takes the different agents' points of view into account is needed.

Abdul-Rahman and Hailes' reputation model [1] approaches the problem from another direction, by defining the trust evaluations based on the actual communications. The interpretation of gossip is based on previous interactions with the same sender. The problem with this, however, is that it is incomplete: firstly it assumes all other agents in the system use the same model, which in a heterogeneous environment will hardly ever be the case. Secondly, it uses a heuristic based on prior experiences, called the semantic distance, to "bias" received messages. The semantic distance is an average of all previous experiences. They do not differentiate between recommendations about different agents, which are based on different types of interactions.

We propose to enrich the model of communication by con-

sidering it separate from the actual trust model. By doing this, we can allow for different trust models. We note, however, that while trust is modeled in disparate ways, all definitions do agree on the fact that trust is a social phenomenon. Just as any social phenomenon, it arises from the complex relationships between the agents in the environment and, without losing generality, we say these relationships are based on any number of interactions between the agents. These interactions can have many different forms, such as playing squash with someone, buying a bicycle on eBay or telling Alice that Dave is a trustworthy keynote speaker. Note that not all interactions are perceived equally by all participants. Due to having different goals, agents may observe different things, or even more obviously: by having a different vantage point. Simply by having more (or different) information available, agents may perceive the interaction itself differently. In addition, interactions may be accompanied by some kind of social evaluation of the interaction. These can range from an emotional response, such as outrage at being cheated in a trade, to a rational analysis. Thus, we see that how an agent experiences an interaction is unique and personal. This only adds to the problem we are considering. To be able to align, there needs to be some common ground from which to start the alignment, but any agent’s experience of an interaction is subjective, and thus not shared. We call this personal interpretation of the interaction an *observation*. We say an agent’s observations allow it to evaluate trust.

Now that we have discussed what interactions mean to a single agent, we will return to the focus of communicating about trust. One interaction may be observed by any number of agents, each making different observations, which support different trust evaluations of different targets performing different roles. However, to communicate about trust evaluations, the agents need to have a starting point: some basic building blocks they implicitly agree they share. We note that the interactions provide precisely such a starting point. While all the agents’ observations are different, they do share one specific thing: *the interaction itself*. We therefore argue that to find a reliable alignment between two agents they can align based on these interactions.

Our approach uses these shared interactions as building blocks to align the agents’ trust models, based on the gossip they send each other. The gossip specifies certain interactions, which each agent observes differently. These observations form the support for an agent’s trust evaluation. If another agent communicates this trust evaluation, the interpretation should be based on the underlying interactions. An alignment of the trust models gives a way of doing this by gossiping about the agents’ trust evaluations and the observations (and thus interactions) they base these on.

Semantic alignment based on interactions has been studied in [2]. This approach to semantic alignment is based on the general framework of Channel Theory [3, 19]. We use this same mathematical theory as a framework for aligning trust and introduce it in the next section before discussing the technical details of the algorithm.

2. THE ALGORITHM

Before we consider possible solutions we need a clear definition of the problem we are considering. We follow the formalization we described in [10] and will summarize it briefly in the following sections. Firstly we consider agents with

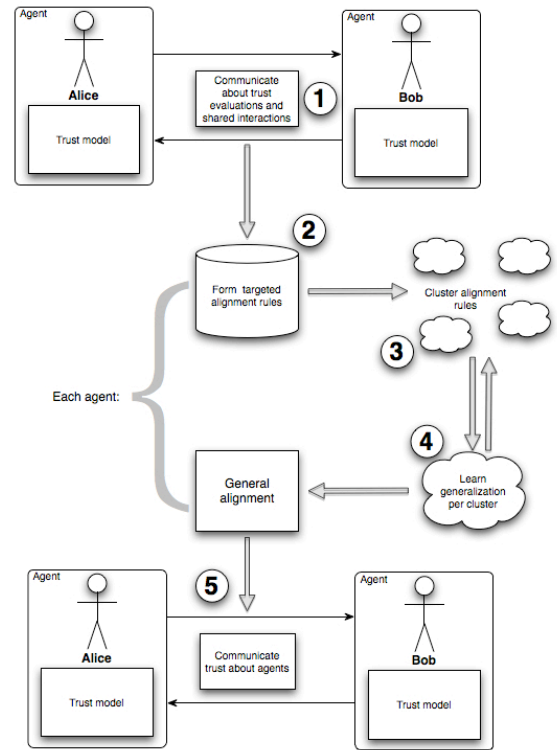


Figure 1: Schematic diagram of the steps in the alignment process

heterogeneous trust models, but we have no clear description of what a trust model is in the first place. We explain this in Section 2.1. Furthermore, to align, the agents need to communicate. For this we will need to define a language in Section 2.1.1. And finally, the agents need to have some method of forming an alignment based on the statements in this language. This we describe in Section 2.1.2. In Section 2.2 we describe the computational approach we take. The whole process is summarized in Figure 1.

2.1 A Formal Representation

As argued in Section 1, interactions form the building blocks for talking about trust. An interaction is observed by different agents and represented internally by them. These observations then lead to trust evaluations of the various agents involved. Any trust model can therefore be described as a binary relation between an agent’s observations and its trust evaluations. In addition, trust always has a target: any form of representing trust will have a trusting agent and a target agent, which is the agent the trust evaluation is about. It is assumed that any agent’s trust evaluations can be represented in some formal language \mathcal{L}_{Trust} . Note that because trust is a subjective phenomenon, the semantics of this language aren’t shared, but by sharing the syntax the agents can communicate about it. A trust model is therefore a binary relation \models , such that $X \models \varphi$ means that there is a set of observations X which support trust evaluation $\varphi \in \mathcal{L}_{Trust}$. The observations X are unknown as they are an internal representation of the agent. However, we know these are based on some set of interactions. If \mathcal{O} is the set of an agent’s possible observations and \mathcal{J} is the

set of all interactions in the environment, then each agent A has a function $observe_A : \mathcal{I} \rightarrow \mathcal{O}_A$ which associates interactions with observations. The observations X in the trust model are therefore generated (with the *observe*-function) from some set of interactions $I \subseteq \mathcal{I}$. These interactions are facts in the environment all agents may know about and can be used as the basis of an alignment.

2.1.1 Formalizing gossip

In addition to \mathcal{L}_{Trust} a second language is needed for effective trust alignment: a language in which to talk about the interactions. Knowing which information about the interactions is relevant depends on the domain. Thus a language for discussing interactions comes from the domain the agents operate in. Usually such a language already exists or is defined together with the MAS. We call this language \mathcal{L}_{Domain} and note that it is a shared language: both the syntax and the semantics are known by all agents in the system, as opposed to the semantics of \mathcal{L}_{Trust} , which is interpreted differently by the agents. With this shared language it is possible to define exactly what it means for two agents to share an interaction. A set of interactions I is shared by agents A and B if there is some $\varphi \in \mathcal{L}_{Domain}$ such that φ is in both A and B 's sets of observations of interaction I , or, in other words, φ is the information shared between the agents about I . Formally neither agent can know that φ is observed by the other agent, however if we limit \mathcal{L}_{Domain} to objective and easily observable properties of the domain, we assume such φ exist.

Messages, containing a trust evaluation of a specific target in \mathcal{L}_{Trust} and pinpointing the specific shared interactions this evaluation is based on in \mathcal{L}_{Domain} , form the basis of the trust alignment. We call such messages ‘‘gossip’’. Formally we say gossip from agent B to agent A is a message $\text{gossip}(T, \beta, \psi)$, with T the target of the trust evaluation $\beta \in \mathcal{L}_{Trust}$ and $\psi \in \mathcal{L}_{Domain}$ describing the set of interactions I which support trust evaluation β for agent B . We cannot simply enumerate the interactions in I because agents may not be willing to do so. \mathcal{L}_{Domain} serves a double purpose: firstly it may be more descriptive, giving more information than simply an enumeration of interactions. Secondly it may allow agents to describe interactions without pinpointing them exactly. This allows agents to align without divulging sensitive information. Sending gossip messages is point 1 in Figure 1.

The receiving agent A can now use its own trust model to find an $\alpha \in \mathcal{L}_{Trust}$, such that α is supported by I and the resulting rule $\alpha \leftarrow \beta, \psi$ will form the basis of our alignment. What this rule means is: the interactions which support ψ , support trust evaluation α for agent A and β for agent B . These rules are at point 2 in Figure 1. The goal is now to find a way of generalizing from such rules to a more general, predictive model, such that, for example, agent A can know what trust evaluation α' it should associate with a certain $\beta' \in \mathcal{L}_{Trust}$, given ψ , despite neither knowing the interactions which support ψ nor being able to conclude an own trust evaluation from the observation of those interactions. This would be the outcome of the algorithm, applied at point 5 in Figure 1.

2.1.2 Generalizations and coverage

Now that we have a way of describing the relationship (alignment) of two agents’ trust models with regards to a

specific target, we wish to expand this idea to a more predictive model: we wish to find the more general alignment between the trust models. This problem is considered as an inductive learning problem [8]. Given a number of targeted alignments with regards to different agents, is there an alignment that describes all (or most) of them?

To use inductive learning, it is necessary to define what the solution should look like. This should be a generalization of the above mentioned rules $\alpha \leftarrow \beta, \psi$. We note that both \mathcal{L}_{Trust} and \mathcal{L}_{Domain} are represented in a standard first-order logic. Thus it is possible to use θ -substitution to generalize these rules. The way to do this is by structuring the search space. The solution should be the least general alignment, which covers all the rules given in the messages. A hypothetical alignment \mathfrak{T} is said to cover a rule $\alpha \leftarrow \beta, \psi$ if there is a rule $\Gamma \leftarrow \Delta, \Psi \in \mathfrak{T}$ such that all sets of interactions I which support $\alpha \leftarrow \beta, \psi$ also support $\Gamma \leftarrow \Delta, \Psi$. One hypothetical alignment \mathfrak{T} is more general than another \mathfrak{T}' if its coverage is greater: $\mathbf{c}(\mathfrak{T}) \supset \mathbf{c}(\mathfrak{T}')$. We write this $\mathfrak{T} \succ \mathfrak{T}'$. The overall trust alignment between two agents can now be found by finding a minimally general generalization, which covers all the communicated rules.

2.2 An Inductive Algorithm

As described in the preceding section, our algorithm must generalize the specific targeted alignments to a predictive ruleset. This is very similar to the problem in concept formation. The approach taken in these problems is by clustering the data together and finding a description of each cluster. However, the fact that we have descriptions in first-order logics invalidates the use of propositional clustering algorithms for this purpose [9]. Some more modern approaches combine clustering and ILP methods [12, 5] to allow for clustering of first-order formulas. This is exactly the problem we are trying to solve and we therefore propose a modification of these algorithms, using the distance function from [17] and a conventional agglomerative clustering algorithm. The found clusters can then be used as the input for an ILP algorithm to learn the generalizations. Furthermore, we have an additional wish: our partitioning may be too strict, which will not allow for enough positive examples and too many negative examples to learn anything useful. In these cases we will want to relax our partitioning criteria to amplify the base of positive examples, in the hope that this will allow for a better generalization. This obviously comes at the cost of accuracy of the predictive ruleset found, but this can be taken into account.

2.2.1 A short overview

The input of the algorithm will be any number of rules \mathcal{R} generated from *gossip* statements. These rules, the same as described in Section 2.1.1, will serve as the initial input and have the form shown below, where T_1, \dots, T_m are target agents, $\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_n \in \mathcal{L}_{Trust}$ and $\psi_1, \dots, \psi_n \in \mathcal{L}_{Domain}$ describe the interactions.

$$\begin{aligned} \alpha_1[T_1] &\leftarrow \beta_1[T_1], \psi_1 \\ &\vdots \\ \alpha_i[T_j] &\leftarrow \beta_i[T_j], \psi_i \\ &\vdots \\ \alpha_n[T_m] &\leftarrow \beta_n[T_m], \psi_n \end{aligned}$$

Algorithm 1 Generalize rules \mathcal{R}

```
1: INPUT: set of SRAs to be generalized  $\mathcal{R}$ 
2: INPUT: distance measure on  $\mathcal{L}_{Trust}$   $D(x, y)$ .
3: INPUT: set of increasing distances for clustering  $S$ 
4: General_rules :=  $\emptyset$ 
5: Clusters :=  $\{\{r\} | r \in \mathcal{R}\}$ 
6: Covered :=  $\emptyset$ 
7: for all Stopcriteria  $s$  in  $S$  do
8:   Clusters := agglomerative_clustering(Clusters,  $s$ ,  $D$ )
9:   if  $|\text{Clusters}| = 1$  then
10:     break
11:   end if
12:   for all  $C \in \text{Clusters}$  do
13:      $H := \text{generalize\_head}(C, \mathcal{R} \setminus C)$ 
14:     if  $H \neq \text{null}$  then
15:        $G := \text{generalize\_body}(C, \mathcal{R} \setminus C)$ 
16:       if  $G \neq \text{null}$  then
17:         General_rules := General_rules  $\cup \{ \langle H \leftarrow G, s \rangle \}$ 
18:         Covered := Covered  $\cup C$ 
19:       end if
20:     end if
21:   end for
22:   if Covered =  $\mathcal{R}$  then
23:     break
24:   end if
25: end for General_rules
```

This says there are n different rules about m different agents. To learn the underlying structure we will use Algorithm 1.

We use three important procedures, which we will explain in more detail: the distance metric D on targeted alignment rules, the clustering algorithm in line 1 and the generalization algorithm we use on the clusters in lines 1 and 1. This last one takes as input the rules in the cluster as positive examples and the rules outside clusters as negative examples and uses an ILP algorithm to learn a generalization. Furthermore we use the flag “terminate” to end the algorithm if at a certain clustering resolution we have rules covering all targeted alignments. In this case there is no reason to continue, because we have a maximum coverage of the examples.

We are interested in finding generalizations which allow us to predict what the receiving agent’s trust evaluation α would be, given that the sending agent’s trust evaluation is β , based on interactions which support ψ . We therefore need to be able to cluster the rules above according to the relative distance between the receiving agent’s trust evaluations. The rest of the information in the rules is used to learn the generalization.

2.2.2 A distance metric

An agent’s trust evaluation is in the \mathcal{L}_{Trust} language, which in general could be any first-order logic. Distances on first-order logic objects have received a lot of attention lately [17]. Such distance measures work on arbitrary clauses, however, they do require them to be rewritten in clausal normal form (CNF). We note that for any closed formula in a first-order logic its CNF can be found in polynomial time [15]. The distance measure is then split up into two different parts:

- A context-dependent part, defining the distance between the disjunctions in the CNF in \mathcal{L}_{Trust}

- A generic part, defining the distance between phrases, based on the distance between the clauses in each phrase.

We stipulate, however, that the distance metric can be agent-specific and may be as complicated as the programmer wishes. To further illustrate this description of a distance metric, we give an example of \mathcal{L}_{Trust} and a metric on it. Our example of \mathcal{L}_{Trust} has the following predicates: $image(A, V)$ and $reputation(A, V)$, where A is an agent and $V \in [1, 10] \subset \mathbb{N}$. For the context-dependent part of the metric we use the closure under symmetry of the following recursive definition:

1. $d(\varphi_1 \vee \varphi_2, \psi_1 \vee \psi_2) = \frac{\min([d(\varphi_1, \psi_1) + d(\varphi_2, \psi_2)], (d(\varphi_1, \psi_2) + d(\varphi_2, \psi_1)))}{2}$
2. $d(\varphi_1 \vee \varphi_2, \psi) = \frac{\min[d(\varphi_1, \psi), d(\varphi_2, \psi)] + 1}{2}$
3. $d(\neg\varphi, \neg\psi) = d(\varphi, \psi)$
4. $d(\neg\varphi, \psi) = 1$
5. $d(image(A_1, V_1), image(A_2, V_2)) = \frac{|V_1 - V_2|}{10}$
6. $d(reputation(A_1, V_1), reputation(A_2, V_2)) = \frac{|V_1 - V_2|}{10}$
7. $d(\varphi, \psi) = 1$ otherwise

As mentioned above, this distance measure is dependent on the language and the agent. All we require in the continuation is that it is defined for all simple clauses in \mathcal{L}_{Trust} and that it is a metric. For that it must satisfy the following properties:

1. *non-negativeness*: $\forall \varphi, \psi : d(\varphi, \psi) \geq 0$
2. *reflexivity*: $\forall \varphi : d(\varphi, \varphi) = 0$
3. *symmetry*: $\forall \varphi, \psi : d(\varphi, \psi) = d(\psi, \varphi)$
4. *strictness*: $\forall \varphi, \psi : d(\varphi, \psi) = 0$ iff $\varphi \equiv \psi$
5. *triangle inequality*: $\forall \varphi, \psi, \theta : d(\varphi, \psi) + d(\psi, \theta) \geq d(\varphi, \theta)$

It is easy to prove that the measure we provided above is a metric, disregarding inequalities between agents.

A generic metric.

Now we can define a generic metric, which uses the context-dependent metric described above. A clausal form can be represented as a set of disjunctions, which allows us to use distance metrics on sets. There are several such metrics available in the literature, but one has been developed for defining distances between first-order logic objects. This metric, designed by Ramon and Bruynooghe [17] uses a matching between two clausal forms to calculate the distance. We use this metric, because it allows a direct syntactic comparison between different formulas. It is once again free to the designer to choose a different metric. All that is really required for the algorithm is for there to be a distance measure on sentences in \mathcal{L}_{Trust} . Clustering algorithms work better with metrics, because the triangle inequality can be used to prune the choices.

2.2.3 Clustering

Because we wish to learn generalizations which predict the receiving agent’s trust evaluations, based on the gossip sent, we want to consider those rules where the receiving agent’s trust evaluations are “near each other”. That means we wish to cluster based on the heads of the rules. It is immediately obvious why an agglomerative hierarchical is the best fit:

- We want to work our way from small precise clusters to large clusters covering a broad spectrum of trust evaluations.
- We want to be able to stop the algorithm when we have found general rules covering all examples.

Bottom-up incremental clustering algorithms fit these criteria best, which leads us to the family of agglomerative clustering algorithms [21]. In this family, complete-link clustering creates more balanced clusters than single-link algorithms, yet has less overhead than average-link algorithms. All other clustering algorithms we explored require the computation of some form of centroid or medioid of the cluster, which speeds up the agglomeration process at the cost of calculating this centroid. Because it is hard to find a centroid for phrases in a first-order logic and we do not expect to have more than a few thousand data points, our choice fell on complete-link clustering. A drawback of complete-link clustering is that it deals badly with outliers. However, we are clustering on the agent’s *own* trust evaluations. If there are outliers, they will not be in these evaluations, but rather the alignment rule itself will be an outlier. We will need to deal with the outliers in the learning of the body, but we should not encounter them when clustering.

Complete-link clustering algorithm.

To start, the complete-link agglomerative clustering algorithm places each element in a separate cluster. It then iteratively merges the two clusters that are nearest together, according to a distance measure between clusters. This distance measure is the maximum distance between two single elements in each cluster, using the distance measure as in Section 2.2.2. This process of agglomeration is continued until there is either only one cluster left, which contains all examples, or some stop criterion has been reached. This stop criterion is defined in line 1 of Algorithm 1. We stop the agglomeration when the distance between two clusters is greater than \mathbf{s} .

A naive implementation of the complete-link agglomerative algorithm would take $O(n^3)$ time, where n is the number of elements to be clustered. The reason is fairly obvious: we start with each element in its own cluster. For each cluster we need to find the distance to each other cluster. This needs to be repeated any time a cluster is merged. Because we start with n clusters, this naive algorithm takes $O(n^3)$ time. This is fairly prohibitive, even for the relatively small datasets we expect to cluster. Luckily there are improvements. Because the distance measure is symmetric, it stands to reason we can skip some calculations. Furthermore, if we merge two clusters then the distance from that cluster to any other cluster is the maximum distance of either of those clusters to the other cluster. This allows us to reduce the algorithm to $O(n^2)$ time in a fairly straightforward manner: for each cluster we need to calculate the distance to each other cluster for which this hasn’t been calculated. There are computational methods, some of which only work for metrics, for optimizing it even further. This makes the computation of clusters quite doable. Clustering is the process at point 3 in Figure 1.

2.2.4 Learning rules

For each distance \mathbf{s} we will have a set of clusters. For each of these clusters we shall attempt to generalize the rules. This is point 4 of Figure 1. Although we clustered on

clausal normal forms of only the heads of the rules, for this part we revert back to the full rule written in the original form. Within the cluster are two or more rules of the form: $\alpha_i[T_j] \leftarrow \beta_i[T_j], \psi_i$.

Learning the head.

All the α_i within a cluster are within distance \mathbf{s} of each other. We therefore start with finding the “centre” of all α_i . Firstly we note that each α_i has a target agent T_j . We will immediately replace all these agents with a variable, because we do not wish to be dependent on the agent. In the future we may not wish to do this, but rather abstract to some subset of all the agents which fulfill a certain role, are within a subgraph of a social network or use other background information about the agents to refine the algorithm. For now, however, we do not distinguish between individual agents and assume trust is global and based only on the interactions. The “centre” of the cluster will be the least general generalization of the α_i under θ -subsumption. It is relatively easy to compute using an algorithm such as Aleph [20]. This is an inductive learning algorithm which uses the “learn from example” setting [8]. We wish to learn some phrase α^* in \mathcal{L}_{Trust} such that if α^* holds then all α_i hold. As parameters for learning we therefore use the definitions of \mathcal{L}_{Trust} and as the set of positive examples the α_i . Because we’re learning the least general generalization (lgg), we can use only positive examples and assume everything that is not a positive example is a negative one. In actual fact this is not quite the case. For example in our example of \mathcal{L}_{Trust} above, if we have the formulas $image(X, 5)$ and $image(X, 7)$ in the same cluster, we will wish to learn that the cluster holds for all phrases such that $image(X, Y) \wedge Y \in [5, 7]$, while this will not be the lgg considering only the given examples as positive: $image(X, 6)$ will necessarily be considered a negative example, leading to the generalization: $image(X, 6) \vee image(X, 7)$. Therefore depending on \mathcal{L}_{Trust} we will want to define some background knowledge in the learner to rectify cases like these.

Learning the body.

The real work comes in when we wish to learn the body. We rewrite our rules with α^* in the head, such that we have a list of rules: $\alpha^*[X] \leftarrow \beta_i[X], \psi_i$, which count as positive examples of the concept α^* . All rules that fall outside the cluster count as negative examples for α^* . Thus giving us the basis required for applying an inductive learning algorithm. Furthermore we note that we have more information available than when we learn the generalization of the head, namely we have a list of situations β_i, ψ_i in which the example holds. This coincides with the “learning from interpretation” setting of ILP [8] and we can use TILDE [4] to learn these generalizations.

3. DISCUSSION AND FUTURE WORK

We are currently in the process of implementing the algorithm as described above. While we do not have any computational results yet, we will discuss our expectations. In [11] we discuss a preliminary proof of concept we implemented using Aleph to learn the rules. This small scenario taught us that the approach is viable, however using that implementation, the computational limitations were inhibitive to scaling the example up. For this reason we have taken great

caution in this approach to keep the computational complexity of each step into account. Firstly we must note that we are dealing with several NP-complete problems: finding the θ -subsumption of a set of clauses has been shown to be NP-complete, as has calculating the coverage of a given clause [8]. It was therefore very important to search for approaches which reduce this complexity. Firstly by clustering our examples and then considering them as positive and negative examples for some concept allows us to use established algorithms for learning. The clustering and learning of the head is a typical example of concept formation, which has an established body of research and is applied in various data mining problems. We feel confident that these approaches, tested in various datamining scenarios will tackle this initial problem well. The second part of the problem uses “learning from interpretations”. While this is still a computationally hard problem, it is easier to learn than the approach using Aleph. TILDE has been tested on some very large data sets and performs efficiently. It is implemented with many optimizations in the ACE package [6].

We are currently implementing the overall system and testing the various components. This is the work for the immediate future. In addition it will be important to assess the quality of the aligned trust models, by comparing the performance of agents using the system to agents using the simpler model of Abdul-Rahman and Hailes [1] as well as agents not aligning at all. We will also extend the algorithm to allow for background knowledge, which can give the system extra information about the agents involved or background knowledge about the interactions and the environment. Furthermore, this model assumes agents always give truthful information. If this is not the case, the learning algorithm will need to be able to cope with “lies”. The mathematical framework we have designed allows for all of this and the combination of different algorithms we use in practice looks promising.

Acknowledgements

This work is supported by the Generalitat de Catalunya under the grant *2009-SGR-1434*, the Agreement Technologies Project *CONSOLIDER CSD2007-0022, INGENIO 2010* and the LiquidPub Project *CIT5-028575-STP*. M. Schorlemmer is supported by a *Ramón y Cajal* research fellowship from Spain’s Ministry of Science and Innovation, which is partially funded by the European Social Fund.

4. REFERENCES

- [1] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 6, 2000.
- [2] M. Atencia and M. Schorlemmer. A formal model for situated semantic alignment. In *Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2007)*, volume 6, pages 1270–1277, Honolulu, Hawaii, USA, 2007.
- [3] J. Barwise and J. Seligman. *Information Flow: The Logic of Distributed Systems*. Cambridge University Press, 1997.
- [4] H. Blockeel and L. De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, 1998.
- [5] H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. In J. Shavlik, editor, *Proceedings of the 15th International Conference on Machine Learning*, pages 55–63. Morgan Kaufmann, 1998.
- [6] H. Blockeel, L. Dehaspe, B. Demoen, G. Janssens, J. Ramon, and H. Vandecasteele. Improving the efficiency of inductive logic programming through the use of query packs. *Journal of Artificial Intelligence Research*, 16:135–166, 2002.
- [7] S. Casare and J. Sichman. Towards a functional ontology of reputation. In *AAMAS ’05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 505–511, New York, NY, USA, 2005. ACM.
- [8] L. De Raedt. *Logical and Relational Learning*. Springer Verlag, 2008.
- [9] D. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.
- [10] A. Koster, J. Sabater-Mir, and M. Schorlemmer. Formalization of the trust and reputation alignment problem. Technical Report TR-2009-03, CSIC-IIIa, 2009. <http://www2.iii.a.csic.es/~andrew/files/techreport.pdf>.
- [11] A. Koster, J. Sabater-Mir, and M. Schorlemmer. An interaction-oriented model of trust alignment. Technical Report TR-2009-05, CSIC-IIIa, 2009. <http://www2.iii.a.csic.es/~andrew/files/techreport2.pdf>.
- [12] F. A. Lisi. Building rules on top of ontologies for the semantic web with inductive logic programming. *Theory and Practice of Logic Programming*, 8(3):271–300, 2008.
- [13] P. Omidyar. Ebay. <http://www.ebay.com>, retrieved September 26, 2008, 1995.
- [14] I. Pinyol and J. Sabater-Mir. Arguing about reputation. the Irep language. In *Proceedings of the 8th Annual International Workshop “Engineering Societies in the Agents World” (ESAW’07)*, volume 4995, pages 284–299. Springer LNCS, 2007.
- [15] D. A. Plaisted and S. Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, 2:293–304, 1986.
- [16] S. D. Ramchurn, D. Huynh, and N. R. Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 19(1):1–25, 2004.
- [17] J. Ramon and M. Bruynooghe. A polynomial time computable metric between point sets. *Acta Informatica*, 37:765–780, 2001.
- [18] J. Sabater-Mir, M. Paolucci, and R. Conte. Repage: REPUTation and imAGE among limited autonomous partners. *JASSS - Journal of Artificial Societies and Social Simulation*, 9(2), 2006.
- [19] M. Schorlemmer, Y. Kalfoglou, and M. Atencia. A formal foundation for ontology-alignment interaction models. *International Journal on Semantic Web and Information Systems*, 3(2):50–68, 2007.
- [20] A. Srinivasan. The aleph manual. <http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/>, retrieved February 9, 2009.
- [21] R. Xu and D. Wunsch II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, May 2005.

Motivation

- In open multi-agent systems, the agents have different models of trust and reputation.
- Agents benefit from receiving trust and reputation information from other agents.
- How can an agent handle trust and reputation information if the other agent's trust model is dissimilar to its own model?
- **We present a framework for aligning trust models. Based on this framework, we present an algorithm which uses clustering and inductive logic programming to form the alignment.**

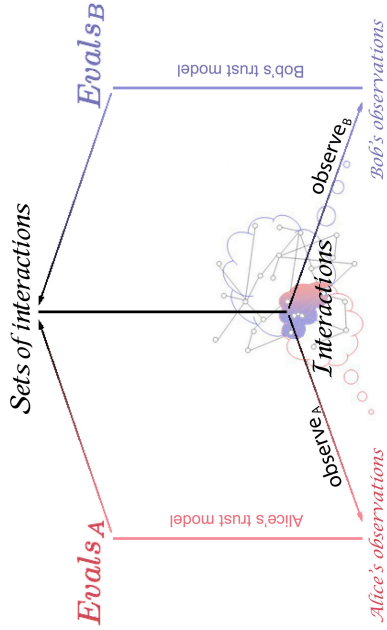


Figure 1. Basing trust on shared interactions

Interactions

- Observed interactions between agents are what all trust models base their evaluations on.
- Every trust evaluation is based on some set of interactions

Alignment

- By talking about the underlying interactions agents can *understand* each others' trust evaluations.
- Align trust models based on this type of gossip.

Algorithm

- 1 **INPUT:** Messages from the other agent, relating trust evaluations to underlying interactions
 - 2 For each message, the agent finds his own trust evaluation of the same interactions
 - 3 Cluster the messages based on the own trust evaluations
 - 4 For each cluster, use an inductive learning algorithm to learn:
 - a. a generalization of the own trust evaluations in the cluster (learning the head)
 - b. a generalization of the other agent's trust evaluation and interactions (learning the body)
 - 5 Interpret new messages from the other agent by finding the corresponding own trust evaluation in the alignment: this is what the other agent's gossip *means* to the receiving agent.
- Implemented using Prolog, Java and OWL.

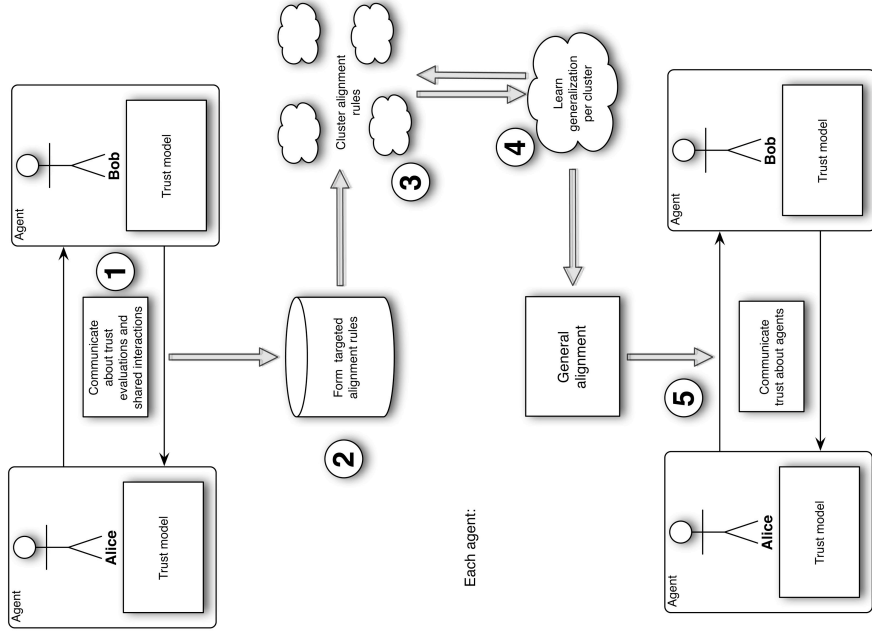


Figure 2. Schematic overview of the alignment process

Conclusions

- Trust alignment based on shared interactions
- We use clustering and inductive learning in an alignment algorithm