Engineering Trust Alignment: a First Approach

Andrew Koster, Jordi Sabater-Mir, and Marco Schorlemmer

IIIA, Artificial Intelligence Research Institute
CSIC, Spanish National Research Council
Bellaterra, Spain
{andrew, jsabater, marco}@iiia.csic.es

Abstract. In open multi-agent systems trust models are an important tool for agents to achieve effective interactions. However, in these kinds of open systems, the agents do not necessarily use the same, or even similar, trust models, leading to semantic differences between trust evaluations in the different agents. Hence, to successfully use communicated trust evaluations, the agents need to align their trust models. We explicate that currently proposed solutions, such as common ontologies or ontology alignment methods, lead to additional problems and propose a novel approach. We show how the trust alignment can be formed by considering the interactions agents share. We describe our implementation of a method, which uses inductive learning algorithms, to accomplish this alignment and test it in an example scenario.

1 Introduction

In open multi-agent systems, trust and reputation models are considered an important feature in managing the social environment agents are immersed in. One of the benefits offered by using trust is that agents can communicate their trust evaluations to each other, thus warning other agents for fraudulent agents or helping each other to select good interaction partners. This communication, however, becomes problematic if the different agents use diverse models of trust, as is a very real possibility in a heterogeneous environment. In this case, trust may mean something different to both agents. These agents need to align their definitions of trust, before the communication becomes meaningful to them.

Trust in computational systems, as in human environments, cannot be seen as independent from the social interactions on which it is based. Each agent computes its trust evaluations of the different target agents in the system, based on some, possibly partial, observation of that agent's interactions. These trust evaluations are computed by an agent's trust model and thus, if the agents use different trust models, then the trust evaluations may be different, despite being based on the same interactions. Additionally we argue that this can be the case even if the agents use the same computational trust model. Some state of the art trust models are based on cognitive principles [1, 2] and take an agent's beliefs and goals into account when computing a trust evaluation. While, in these cases, the computational model is the same, agents with different beliefs and goals will

have different trust evaluations given exactly the same information. We see that also in these cases it is therefore important to align the trust models.

The interactions trust is based on depend largely on the purpose of the multi-agent system and the types of agents in it. They can be as diverse as air traffic agents optimizing a landing schedule, personal agents buying a bicycle on eBay or agents communicating trust evaluations. The developers of the system generally develop an ontology to facilitate communication about these interactions and the domain in general. This should facilitate the communication about the interactions, however the trust evaluations are not based on public information alone. An agent may have its own personal observations of an interaction. In the case of an eBay auction, the seller for instance has information about all bids received, while the bidder only has information about his own bids. Additionally agents may associate different subjective observations with the interaction. For instance, the seller may not be satisfied with the transaction, because he had to sell at a loss. This type of information is private and often just as subjective as the trust evaluation itself. In the eBay example it is no easier to communicate the meaning of satisfaction than of the trustworthiness supported by that satisfaction. This difference in observations complicates the matter of aligning trust models, however we postulate that there is always some amount of shared information. At the very least, there is shared information that an interaction took place. Our approach uses these shared interactions as building blocks for a trust alignment.

So far, communication about trust evaluations has been tackled by defining common ontologies for trust [3, 4], however in practice these ontologies do not have the support of many of the different trust methodologies in development. An ontology alignment service is presented in [5], but it requires a translation of all specific trust model ontologies into a general ontology. In addition, even if support were added for all systems and a common ontology emerged, a cognitive agent will still have its own interpretation of the world on which it bases its trust evaluations: thus trust must always be considered in the light of *why* agents trust each other.

Abdul-Rahman and Hailes' reputation model [6] approaches the problem of alignment 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 cannot be assumed. Secondly, it uses a heuristic based on prior experiences, to "bias" received messages. This bias is an average of all previous experiences. They do not differentiate between recommendations about different agents, which are based on different types of interactions.

Semantic alignment based on interactions has been studied in [7]. This approach to semantic alignment is based on the general framework of Channel Theory [8, 9]. We use this same mathematical theory as a framework for aligning trust.

2 The Algorithm

Channel Theory is a qualitative theory modeling the flow of information in distributed systems. From our point of view we can use this to describe a channel in which information about trust can be transferred from one agent to another. This is described in detail in [10]. The intuition is that both agents can relate each others' subjective trust evaluations to the objective descriptions of interactions, communicating about them using the languages $\mathcal{L}_{Trust}$ and $\mathcal{L}_{Domain}$, respectively. By doing so they are able to find the underlying meaning of the trust evaluations. The computational model based on this approach is described in [11] and we will summarize it here before explaining our implementation.

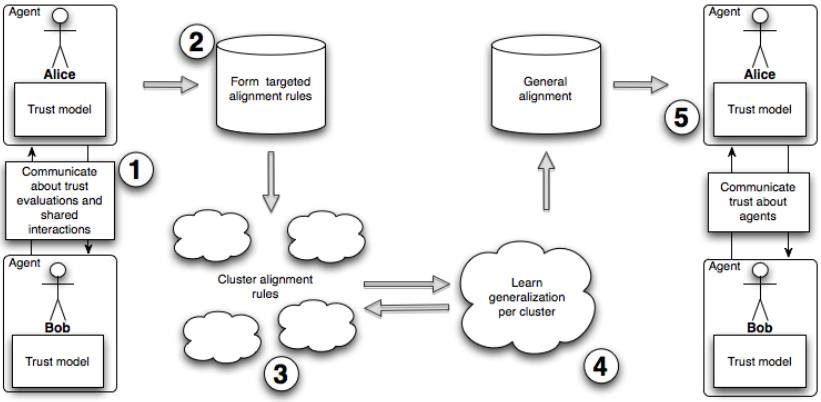


Fig. 1. Schematic diagram of the steps in the alignment process

In Figure 1 we give a graphical overview of the algorithm. First, at ① the agents have to communicate their trust evaluations to each other in the form of `gossip` messages. For each message, the receiving agent computes its own trust evaluation, leading to a set of Specific Rules for Alignment (SRAs) at ②, each of the form $\alpha_i[T_j] \leftarrow \beta_i[T_j], \psi_i$, which would be the ith SRA about the target agent T_j. The heads of the rules α are the own trust evaluations, while β in the bodies are the other agent's. ψ describes the set of interactions which support both evaluations. The agent then has to learn the model underlying these SRAs. This is done, in repeated steps of clustering ③ and inductive learning ④. The output will be a set of General Rules for Alignment (GRAs), which is the generalization of the SRAs we give as input. These can then be used to interpret future messages ⑤. We describe the method used in Algorithm 1.

We use three important procedures, the clustering algorithm in line 5 and the two generalization algorithms we use on the clusters in lines 9 and 11.

– *Clustering* is used to group those SRAs where the *receiving* agent's trust evaluations are "near each other", because we want to learn generalizations that will predict that agent's trust evaluations, based on the gossip sent. That means we cluster based on the heads of the SRAs and we have an additional requirement for $\mathcal{L}_{Trust}$: there must be a distance measure defined on it. An

Algorithm 1: Generalize SRAs

```
   Input: R, the set of SRAs to be generalized
   Input: D(x, y), a distance measure on L_Trust
   Input: S, a set of increasing distances for clustering
 1 GRAs := ∅
 2 Clusters := {{r}|r ∈ R}
 3 Covered := ∅
 4 foreach Stop_criteria s in S do
 5     Clusters := agglomerative_clustering(Clusters, s, D)
 6     if —Clusters— = 1 then
 7         break
 8     foreach C ∈ Clusters do
 9         H := generalize_head(C, R\C)
10         if H ≠ null then
11             G := generalize_body(C, R\C)
12             if G ≠ null then
13                 GRAs := GRAs ∪ {⟨H ← G, s⟩}
14                 Covered := Covered ∪ C
15     if Covered = R then
16         break
17 Output: GRAs
```

example of such a distance measure is described in Section 3.3. The clustering fulfills another role in the algorithm: it allows for the incremental learning of the alignment. This allows us to stop the algorithm when a suitable alignment is found. To this purpose we use a list of stop criteria $\mathtt{S}$. This is a list of maximum distances for the clustering algorithm. The algorithm goes through this list from smallest distance to greatest and continues merging clusters until all the clusters are at a distance greater than the current stop criterion being evaluated. The resulting set of clusters will serve as input for the learning algorithms.

– *Generalizing* the SRAs is the main part of the algorithm. For each stop criterion $\mathtt{s}$ we will have a set of clusters of SRAs and for each of these clusters we shall attempt to generalize a set of GRAs covering it. Our first task is to learn a generalization of the heads of the SRAs. By definition, all the α_i within a cluster are within distance $\mathtt{s}$ of each other and we want to find some defining quality of these α_i which we can use in our final ruleset. We want to learn the generalization α^* which θ-subsumes [12] all α_i. Afterwards, when we generalize the body, we are learning the conditions for which the receiving agent should have trust evaluation α^*. For both these tasks we can use an inductive learning algorithm, however the specific type of learning differs. For generalizing the heads of the SRAs we use an algorithm specialized in the "learn from example" setting [12], whereas for generalizing the bodies, we have richer information available and by using the "learn from interpretation" setting [12] we can use heuristics which take advantage of this, resulting in a faster algorithm for similar problems.

If we can find a generalization for the body it means we have a GRA which covers all of the targeted rules in the cluster. We stop the algorithm when all SRAs are covered, or when the remaining clusters are further apart than the largest stop criterion. When the algorithm ends we have a list of GRAs. This list can be used to translate messages from the other agent. Because each GRA is stored with the stop criteria which allowed it to be generated, we have an internal distance of the cluster it covers. We use this as the measure of accuracy of the alignment. We can use this, together with the actual aligned message, in the trust model.

3 Implementation and results

Now that we have described the problem and given an explanation of the system, we will give a brief description of the tools used to implement it. The implementation must:

- define a language for $\mathcal{L}_{Trust}$ and $\mathcal{L}_{Domain}$.
- implement the incremental clustering algorithm.
- use a "learn from example" ILP algorithm on the heads of SRAs.
- use a "learn from interpretation" ILP algorithm on the bodies of SRAs.

The implementation is predominantly in Java, which allows for flexibility in the tools used. For logical reasoning we use SWI-Prolog, which provides a JNI interface, so it can be accessed from Java.

3.1 $\mathcal{L}_{Trust}$ and $\mathcal{L}_{Domain}$ in OWL

To be able to gossip, the agents need two separate languages. One for the trust evaluations and one domain dependent language to talk about interactions. Often agents will be developed for domains where there already is a fairly extensive domain language available, so it makes sense to adopt this language as our $\mathcal{L}_{Domain}$. If there is no such global language, then agents will need to align their domain languages first. This is a separate problem and its solution is outside the scope of this paper. We assume there is a shared $\mathcal{L}_{Domain}$ language and we follow the W3C recommendation for its specification: we use the Web Ontology Language (OWL) [13]. Statements in $\mathcal{L}_{Trust}$ and $\mathcal{L}_{Domain}$ are expressed in a subset of OWL-DL in which we will not allow quantification at this point, because our learning algorithms are unable to deal with quantified variables in the examples. This is not a very big restriction, because we are always gossiping about one specific target agent, based on certain specific interactions and we can give the specific instance, without using quantification.

A remark about the semantics of $\mathcal{L}_{Trust}$: while we use OWL-DL to specify the language, we only fix the semantics of the connectives. The meaning of the predicates themselves is precisely what we want to align.

3.2 Clustering and Learning

As described in Section 2, agglomerative clustering methods best fit our needs. In this family, complete-link clustering [14] creates balanced clusters without requiring the computation of some form of centroid or medioid of the cluster. The calculation of such a centroid in our example is computationally intensive, as it is equivalent to generalizing the head of the SRAs. We will still need to compute this generalization, but not when using it as a distance measure, but only once the cluster is finalized. 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 SRA 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. Our implementation is based on the algorithm described in [15], sufficiently optimized,

in Java. The distance measure on the clusters depends on the distance measure on the elements, which are the statements in $\mathcal{L}_{Trust}$. Our implementation runs in $O(n^2)$ time, where n is the number of SRAs formed from communication.

Aleph and Tilde For the generalization of the heads of all SRAs in the cluster we use Aleph [16], because it functions well at the "learn from example" setting. For learning the generalization of the bodies we use TILDE [17], which was designed for "learning from interpretation". Because the clusters aren't known beforehand and both these algorithms run with a specific format of input files, we generate the files for each cluster at runtime. Then we call the algorithm on the newly created files. The output is then read and reformatted so the head and body together form a complete Prolog clause. If both steps succeed, they result in a GRA, which covers the SRAs in the cluster. The algorithm terminates when all SRAs are covered in this manner.

3.3 Example environment

So far we have explained the implementation of the algorithm. To test it we designed an experimentation environment with a specific trust and domain language, a way of generating interactions and agents with different trust models. We base this environment on the following example scenario: Alice is looking for a keynote speaker for a conference and wants advice from Bob on who is trustworthy. This forms the basis of our example, with Bob and Alice trusting different agents based on different criteria. We form a random graph of interactions, which may range from an extremely sparse to a complete network, depending on the chance agents interact. The number of interactions between agents is customizable in a configuration file by varying the number of agents in the system and the chance agents interact. So far we haven't added support for varying the topology of the network, because testing the alignment depends mainly on the quantity of interactions, rather than the shape of the network. The ontology for $\mathcal{L}_{Domain}$ is defined in Figure 2. It is kept deliberatively small and describes only objective properties of the interactions.

Based on these interactions, the agents compute trust evaluations, which is communicated using $\mathcal{L}_{Trust}$. In this example $\mathcal{L}_{Trust}$ only has one predicate: image(Target, Value)[1], with $Value \in [1, 10]$. The distance between two elements image(T_1, V_1) and image(T_2, V_2) is defined as $\frac{|V_1 - V_2|}{10}$. This is the normalized distance between the two values. We purposefully disregard which agent is evaluated, because we want the clusters to generalize over all similar trust evaluations, ignoring which agent is being evaluated.

Trust Models We specify the trust models in our environment using Prolog programs. The relevant parts of the two models we use in our example are outlined in pseudocode in Table 1. The interactions are described in $\mathcal{L}_{Domain}$, which allows for 3 types of interactions: writing an article, a lecture and a personal interaction. These are observed and the participants in the interactions

[1] image is the trust evaluation of the target that the agent believes in

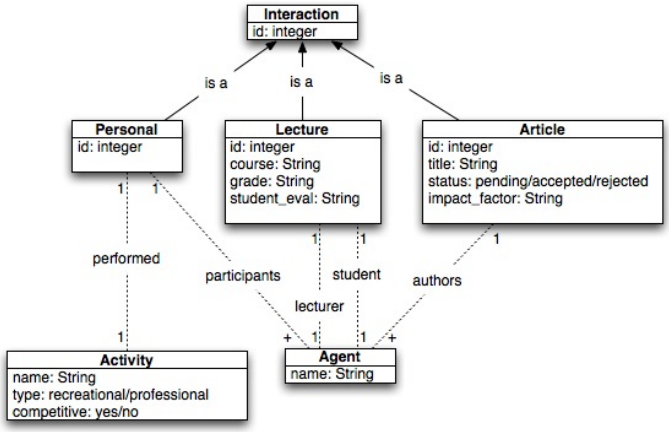


Fig. 2. The ontology for $\mathcal{L}_{Domain}$, in a UML-like representation

are evaluated in the trust models. We see that there are correspondences between the trust models, such as an interaction resulting in a high impact article, supports a high value image in both models. However we also see that Alice's model is less general than Bob's in this aspect: any article with an impact factor higher than 6 results in Alice evaluating the authors thereof with an image with value 10 (line A1). Bob's model splits this into articles with an impact factor higher than 8 (B1) and an impact factor between 6 and 8 (B4). Vice versa, Bob's trust model is quite general regarding lecture interactions where the target to be evaluated is the lecturer (B9). Any such interaction supports a trust evaluation with an image of 5. Alice's trust model considers such interactions in a more fine-grained manner, using various different rules (A2, A3, etc.). Finally we see a large difference between the trust models in that Bob's trust model has quite a few trust evaluations supported by lecture interactions where the target is a student (B2, B3, etc.). These simply do not correspond to any trust evaluation in Alice's model. These characteristics make these trust models useful for analyzing the running of the algorithm, because they cause the problems we typically expect to encounter when aligning.

Alice and Bob's knowledge bases contain the shared interactions between them. Each agent can therefore, based on these interactions, compute all possible trust evaluations for all agents in the system. In our experiments we consider Alice aligning with Bob, so we have Bob compute his trust evaluations and generate all the messages he can. These are sent to Alice, who uses them as input for the alignment algorithm. The reverse problem of Bob aligning with Alice is analogous.

3.4 Experiments and analysis

We run the above environment with stop criteria 0, 0.1, 0.5, 0.8 and 0.9. These give the increasing internal distance of the clusters we will attempt to learn generalizations for. The interactions are a randomly generated set, based on

Alice's trust model

```
A1  image(T, 10) ← article(I), author(I, T) impact_factor(I, Imp), Imp > 6.
A2  image(T, 8) ← lectured(I), lecturer(I, T), student_eval(I, excellent).
A3  image(T, 8) ← lectured(I), lecturer(I, T), grade(I, G), G > 6.
A4  image(T, 7) ← personal(I), participant(I, T), activity_type(I, recreational).
A5  image(T, 7) ← lectured(I), lecturer(I, T), student_eval(I, good).
A6  image(T, 7) ← article(I), authors(I, T), impact_factor(I, Imp), Imp > 4, Imp < 7.
A7  image(T, 6) ← lectured(I), lecturer(I, T), student_eval(I medium).
A8  image(T, 6) ← article(I), author(I, T), evaluated(I, accepted).
A9  image(T, 5) ← article(I), author(I, T), evaluated(I, rejected).
A10 image(T, 4) ← personal(I), participant(I, T), activity_class(I, competitive).
A11 image(T, 3) ← lectured(I), lecturer(I, T), student_eval(I, bad).
A12 image(T, 1) ← lectured(I), lecturer(I, T), student_eval(I, awful), !.
```

Bob's trust model

```
B1  image(T, 10) ← article(I), author(I, T), impact_factor(I, Imp), Imp > 8.
B2  image(T, 10) ← lectured(I), studied(I, T), student_eval(I, excellent).
B3  image(T, 10) ← lectured(I), studied(I, T), grade(I, G), G > 8.
B4  image(T, 8) ← article(I), author(I, T), impact_factor(I, Imp), Imp > 6.
B5  image(T, 8) ← personal(I), participant(I, T), activity_type(I, recreational),
                  activity_class(I, cooperative).
B6  image(T, 7) ← lectured(I), studied(I, T), student_eval(I, good).
B7  image(T, 7) ← personal(I), participant(I, T), activity_class(I, cooperative).
B8  image(T, 7) ← personal(I), participant(I, T), activity_type(I, recreational).
B9  image(T, 5) ← lectured(I), lecturer(I, T).
B10 image(T, 5) ← article(I), author(I, T), evaluated(I, accepted).
B11 image(T, 4) ← lectured(I), studied(I, T), grade(I, G), G < 6.
B12 image(T, 4) ← lectured(I), studied(I, T), student_eval(I, medium).
B13 image(T, 4) ← lectured(I), studied(I, T), student_eval(I, bad).
B14 image(T, 2) ← personal(I), participant(I, T), activity_type(I, professional),
                  activity_class(I, competitive).
B15 image(T, 2) ← lectured(I), studied(I, T), student_eval(I, awful).
B16 image(T, 1) ← article(I), author(I, T), evaluated(I, rejected).
```

Table 1. Two sample trust models in Prolog-like syntax

different configuration settings. We use 60% of these interactions as our training set and 40% as the control set. We run thirty trials at each configuration and Table 2 shows a summary of the results, where each row is the average over the trials run. The first two columns describe the configuration and the third the number of interactions the configuration results in. The other columns we describe below.

Num. agents	Interact chance	Interact number	Coverage training	Coverage control	Accuracy
10	10%	14	75%	45%	0.99
10	30%	40	89%	70%	0.97
10	70%	95	99%	95%	0.92
25	1%	7	69%	41%	0.96
25	5%	44	78%	66%	0.95
25	10%	96	96%	93%	0.95
25	30%	275	94%	94%	0.92
25	50%	451	98%	98%	0.87
50	2%	74	76%	65%	0.96
50	10%	373	97%	97%	0.94
50	30%	1096	100%	100%	0.99

Table 2. Summary of results

Coverage of the Alignment Our algorithm always results in a "catch all" GRA at the highest stop criterion of 0.9. All SRAs are covered by this rule, however it adds no information. We discuss this further in Section 3.4. The coverage in Table 2 is calculated as the percentage of SRAs that are covered by any GRA other than this "catch all" GRA.

We see that even in a fairly small shared network, such as the one with 10 agents and a 30% interaction chance the alignment achieves a fairly high coverage of 70%. This was based on an average of 40 interactions which led to an average of 48 SRAs for aligning. The networks smaller than that do not allow for good alignment. With 10 agents and 10% chance of interacting the coverage drops below 50% with quite a few instances of runs with 0% coverage of the

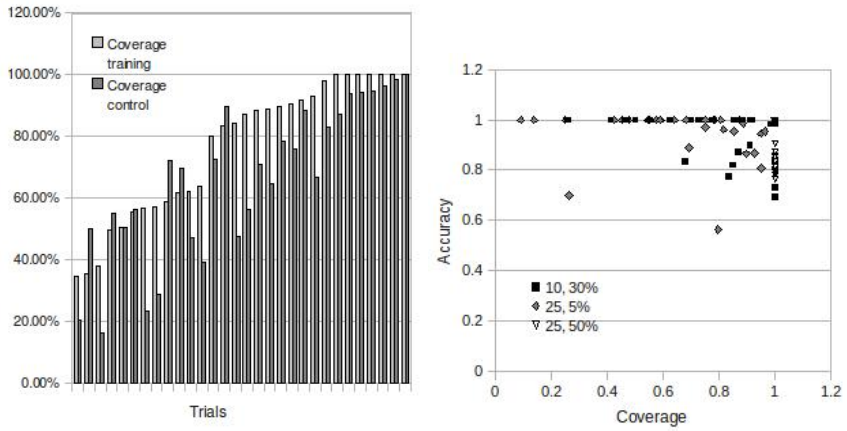
training set. This is based, on average, on 14 interactions. The trials with 25 agents and 1% chance of interaction suffer even more under the network size: with only around 7 interactions to base the alignment on, 5 of the 30 trials did not complete. Of course, by using all interactions, and not just 60%, an agent can improve its learning capacity in small networks, but in this example around 20 interactions is the minimum for a reasonable alignment. This improves fairly rapidly if agents have more shared interactions, because, as is to be expected, the quality of the alignment is mainly dependent on the number of interactions. We see for instance that the trials with 25 agents and a 5% chance of any two agents interacting on average finds an alignment that covers 66%, a vast improvement over the network with 1% interaction chance. Number of interactions is not the only factor influencing the coverage of the alignment. Sometimes the training set does not allow for a good alignment and the training set has bad coverage. Luckily, there is a correlation between the coverage of the training set and the control set: the Pearson's correlation coefficient is 0.51[2] and if we do not take the data from the two smallest sets into account, which do not result in decent alignments, the correlation is stronger, with a coefficient of 0.72[3]. In Figure 3(a) we have graphed the coverage of the training and control set for trials with a small sized network: with 50 agents and a 2% chance of interacting, displaying this correlation. This correlation is useful, because an agent can know the coverage of its training set and thus if the training set has low coverage it can expect bad results when applying the alignment.

The knowledge about the training set is useful. If the training set is small or has low coverage the agent knows that its alignment probably has a low coverage for any future messages from the other agent too. Furthermore, it can attempt to improve the alignment by obtaining information about more interactions and the other agent's trust evaluations based on them. We show the improvement an agent can achieve by ignoring badly covered training sets. In Table 3 we give the average coverage of the control sets if we leave out all alignments that cover less than 70% of the training set, showing a marked improvement in the trials run in smaller networks. In medium networks with over 100 interactions we can achieve a similar improvement by leaving out all alignments with less than 100% coverage of the training set. In the largest networks, with over 500 interactions the coverage of both sets is so near perfect that there is no significant improvement.

Experiment	10, 10%	10, 30%	10, 70%	25, 1%	25, 5%
Coverage	59%	75%	96%	49%	77%
Number ignored	9	5	2	9	12
Experiment	25, 10%	25, 30%	25, 50%	50, 2%	50, 10%
Coverage	97%	98%	100%	81%	100%
Number ignored	3	2	6	12	5

Table 3. Corrected coverage of control set, by removing the trials with bad coverage of the training set

[2] With 95% confidence we know the correlation coefficient is between 0.44 and 0.59
[3] With 95% confidence we know the correlation coefficient is between 0.66 and 0.77



(a) Coverage of training and control sets (b) Coverage vs. accuracy

Fig. 3. Graphs showing various aspects of the coverage

Accuracy of the Alignment As mentioned in Section 2, the accuracy of a GRA is directly proportional to the maximum distance of the SRAs it covers, in fact, the accuracy is $1 - \mathtt{s}$, with $\mathtt{s}$ the stop criterion used to generate the cluster of SRAs. We define the accuracy of the alignment as the average of the accuracy of the GRAs used to translate each message in the control set. Thus any uncovered message is not taken into account in the accuracy calculation. In most cases we find an extremely accurate alignment. Often, the highest accuracy is when coverage is low, while the alignments with higher coverage have a lower overall accuracy. This can be explained by looking at the way the learning algorithm works: when we decrease accuracy by moving to a large clustering distance, any GRAs learned will have a higher coverage. Most alignments in the small and medium networks with a high coverage have one or two GRAs with lower accuracy. We have plotted the relation between accuracy and coverage for a few trials in Figure 3(b), where we see that lower accuracy occurs more often at high coverage.

Additionally, for our example, we can look at the trust models to see that an accuracy of 100% is a theoretical impossibility. In some situations, especially those concerning lecture interactions, Alice's trust model is far preciser than Bob's, leading to messages from Bob necessarily being aligned with a lower accuracy. We expect this to be a general feature of two different trust models.

If we take coverage into consideration, the average accuracy drops to around 40% for the smallest data sets. Larger networks aren't influenced as much by this correction, because coverage tends to be higher and accuracy already lower. The large networks have similar accuracy before and after the correction: around 90%.

Unaligned messages As described in Section 2, there is another measure of the coverage of the alignment, which we have so far ignored: the messages Bob sends

which are based on interactions which support no corresponding trust evaluation for Alice, for instance in our example Alice cannot use any trust evaluation Bob bases on `lecture` interactions in which the student is evaluated. These cases are not taken into account in the data displayed above, because there are two ways of considering them. One possibility is to consider any such message as successfully aligned: the agent knows these messages do not result in a corresponding trust evaluation. We could therefore use the learner to find a generalization of these messages in the same manner as we learn the body of messages that do result in an alignment. In this example these messages can easily be generalized: they are either the `lecture` interactions described above, or `personal` interactions with a cooperative and professional activity. Learning this generalization results in a 100% coverage for both training and control groups and thus Alice can know in the future when a message does not correspond to an own trust evaluation. The other possibility is for an agent to consider such messages unaligned. This results in an overall lower coverage of both the training and control sets. In this example 21% of the messages Bob sends do not result in a trust evaluation for Alice.

We prefer the first way of considering these messages, because knowing in which situations a trust evaluation of the other agent does not correspond to any trust evaluation in the own agent is valuable information in and of itself and can even be considered as a successful alignment: knowing *when* there is no information is also information!

4 Conclusion

The problem we address in this paper is that of aligning trust models. We describe the implementation of an alignment algorithm and have performed a preliminary set of experiments with it. The experiments show that even at low numbers of interactions the coverage and accuracy of the alignment is quite high, although for a reliable 100% alignment it requires large amounts of interactions. Luckily, just by analyzing the coverage of the training set, we can estimate whether the alignment will be effective. This gives the agent an extra tool in the application of the alignment. Furthermore we see that the average accuracy is high for aligned messages.

The trust models we used are simple and in the future we will test the algorithm with actual trust models, used in the community. We will also test it in a scenario with real data, rather than randomly generated interactions. This approach will necessitate more robust accuracy checks and outlier detection, because in these experiments we did not deal with noise or inaccurate trust evaluations. The ILP algorithms used can be configured to deal with noisy data, however we did not do this in these experiments: we only use results from TILDE with a 100% coverage of the examples, by allowing results with less coverage, we would need to combine the accuracy results from TILDE with our own accuracy calculations. Because there are various ways of doing this and it was unnecessary for the example scenario we have not considered this.

References

1. Sabater-Mir, J., Paolucci, M., Conte, R.: Repage: REPutation and imAGE among limited autonomous partners. JASSS - Journal of Artificial Societies and Social Simulation **9**(2) (2006)
2. Hübner, J.F., Lorini, E., Herzig, A., Vercouter, L.: From cognitive trust theories to computational trust. In: Proc. of the Twelfth Workshop "Trust in Agent Societies" at AAMAS '09, Budapest, Hungary (2009) 55–67
3. Pinyol, I., Sabater-Mir, J.: Arguing about reputation. the lrep language. In Artikis, A., O'Hare, G., Stathis, K., Vouros, G., eds.: Engineering Societies in the Agents World VIII: 8th International Workshop, ESAW 2007. Volume 4995 of LNAI., Springer Verlag (2007) 284–299
4. Casare, S., Sichman, J.: Towards a functional ontology of reputation. In: AAMAS '05: Proc. of the fourth international joint conference on Autonomous Agents and Multiagent Systems, Utrecht, The Netherlands, ACM (2005) 505–511
5. Nardin, L.G., Brandão, A.A.F., Muller, G., Sichman, J.S.: Effects of expressiveness and heterogeneity of reputation models in the art-testbed: Some preliminar experiments using the soari architecture. In: Proc. of the Twelfth Workshop "Trust in Agent Societies" at AAMAS '09, Budapest, Hungary (2009)
6. Abdul-Rahman, A., Hailes, S.: Supporting trust in virtual communities. Proceedings of the 33rd Hawaii International Conference on System Sciences **6** (2000) 4–7
7. Atencia, M., Schorlemmer, M.: I-SSA: Interaction-Situated Semantic Alignment. In: OTM 2008, Part I. Volume 5331 of LNCS., Springer (2008) 445–455
8. Barwise, J., Seligman, J.: Information Flow: The Logic of Distributed Systems. Cambridge University Press (1997)
9. Schorlemmer, M., Kalfoglou, Y., Atencia, M.: A formal foundation for ontology-alignment interaction models. International Journal on Semantic Web and Information Systems **3**(2) (2007) 50–68
10. Koster, A., Sabater-Mir, J., Schorlemmer, M.: A formalization of trust alignment. In Sandri, S., Sànchez-Marré, M., Cortes, U., eds.: AI Research and Development. Proc. of the Twelfth International Congress of the Catalan Association of Artificial Intelligence (CCIA 2009). Volume 202 of Frontiers in Aritficial Intelligence and Applications., Cardona, Spain, IOS Press (2009) 169–178
11. Koster, A., Sabater-Mir, J., Schorlemmer, M.: Inductively generated trust alignments based on shared interactions (extended abstract). In: Ninth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2010), Toronto, Canada, IFAAMAS (In Press)
12. De Raedt, L.: Logical and Relational Learning. Springer Verlag (2008)
13. McGuinness, D.L., van Harmelen, F.: Owl web ontology language overview. http://www.w3.org/TR/owl-features/, retrieved July 27, 2009
14. Defays, D.: An efficient algorithm for a complete link method. The Computer Journal **20**(4) (1977) 364–366
15. Johnson, S.C.: Hierarchical clustering schemes. Psychometrika **32**(3) (September 1967) 241–254
16. Srinivasan, A.: The aleph manual. http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/, retrieved 9/2/2009 (June 2004)
17. Blockeel, H., Dehaspe, L., Demoen, B., Janssens, G., Ramon, J., Vandecasteele, H.: Improving the efficiency of inductive logic programming through the use of query packs. Journal of Artificial Intelligence Research **16** (2002) 135–166