Contexts in Dynamic Ontology Mapping

Paolo Besana and **Dave Robertson**
Centre for Intelligent System and their Applications
School of Informatics
University of Edinburgh

Abstract

Agents in open systems interact continuously, each possibly having a different ontology. Mapping in advance all the ontologies that an agent can encounter is not feasible, as all the possible combinations cannot be foreseen. Mapping complete ontologies at run time is a computationally expensive task. This paper proposes a framework in which mappings between terms may be hypothesised dynamically as the terms are encountered during interaction. In this way, the interaction itself defines the context in which small, relevant portions of ontologies are mapped. We use this way of scoping the ontology mapping problem in order to apply mapping heuristics in a more focused way.

Problem description

In order to act properly after receiving a message from an external entity, an agent must understand the content of the message.

A message is created by mapping concepts in the sender's representation of the domain into the terms that compose the message, conforming to the syntax of the language it uses. The receiver maps the terms in the message to the concepts in his own representation, helped by the syntax rules that structure the message. If a term is mapped to a different concept by the receiver agents, or cannot be mapped, then a misunderstanding arises.

The problem would not exist if both agents shared the same representation of the domain, but this is not the most common case.

For example, in an open B2B market, agents gather to offer and to request services or products, working as proxy for their companies. Agents converge to the market from different backgrounds, and are likely to have different ontologies. The brokers receive advertisements of offers, and must classify them correctly to match them with the requests. As agents continuously arrive to the market and leave, the number of possible combinations of agents is high, .

Common approach

Early attempts to overcome the heterogeneity in the representations were to develop general ontologies that could cover a majority of domains and could be shared by the agents. This approach has been unsuccessful on a large scale. First, it proved difficult to find an agreement on what ontology to use. Second, it is difficult to manage the evolution of the ontology: an old version can be inconsistent with a newer one (Hameed, Preece, & Sleeman 2003).

More recent attempts are instead focused on reconciling different ontologies, allowing their coexistence.

Problems with Ontology Mapping in MAS

Most mapping processes are aimed at statically aligning complete ontologies (Kalfoglou & Schorlemmer 2003; Giunchiglia, Shvaiko, & Yatskevich 2004; Nuno & Rocha 1999): two or more ontologies are reconciled and the result is stored for future use.

Preparing in advance all the possible mappings between the ontologies is not feasible in open multi-agent systems, as it is impossible to foresee all the combinations of agents involved in the interactions. Mapping whole ontologies, often a lengthy process, may not be feasible at real time. Interactions should be quick and many can occur at the same time. Moreover, only portions of ontologies may match, as agents can have ontologies about different domains.

Proposed approach

A complete agreement over the semantics is not required in MAS: agents interact only when they must, and they need to understand each other just enough to perform their task. Once the task is performed, mutual understanding is no longer important: agents in an open system interact continuously with different agents, and the mapping found once might be useless any other time.

To perform a coordinated task, the involved agents need to share only the parts of their knowledge contextual to the interaction. It is possible to exploit this idea and map dynamically and only when needed the portions of the ontologies required for the context of the interaction.

Definitions and assumptions

Agent model

Each agent a_i has its own communication environment e_i, consisting of the ontology O_i that defines the terms used by the agent and of the axioms it can use to reason.

An environment can be seen as the context of an agent, as described in (Giunchiglia 1992), but renamed to avoid name conflicts with the concept of context used in this paper.

Any definition is valid only within an environment. An agent can reason over concepts defined in other environments only if mapped to its own concepts.

An agent, for the sake of this paper, can be model as being composed by two layers: a *communication layer*, and a *reasoning layer*. The communication layer is the interface between an agent and the other agents in the system. In the basic case, it handles the transmission and reception of messages. The reasoning layer contains all the agent's skills and knowledge, and it is accessible from the communication layer through access points.

Communication model

During an interaction k an agent a_i sends to an agent a_j a message m composed of terms. For brevity, terms defined in the agent's environment will be called *internal terms*, and will be referenced as w_i(see figure 1), while terms defined in other environments will be called *external terms*, and will be referenced as t_i.

For an interaction k, every involved agent a_i publishes an ontology subset O_{ki}, valid in the context of the interaction, to explain the terms it has used in the messages.

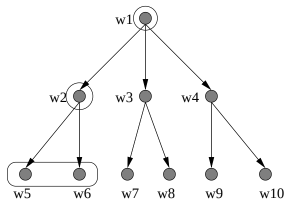

Figure 1: O_i ontology

Semantic bridges

The semantic relations between terms defined in different environments is defined in semantic bridges (Nuno & Rocha 1999). A bridge b is the tuple:

$$b = \langle relation, t, w, c(true), c(false) \rangle$$

where $relation$ can be equivalence or subsumption, while $c(true)$ is the confidence level that the bridge is correct and $c(false)$ that the bridge is wrong.

A bridge b_h is more generic ($\succeq$) than another bridge b_g, if the external term t is the same in both and the internal term w_h of b_h subsumes the internal term w_g in b_g:

$$b_h \succeq b_g \leftrightarrow (w_h \sqsupseteq w_g) \wedge (t_h = t_g)$$

Conversely b_h is more specific than b_g if w_g subsumes w_h:

$$b_h \preceq b_g \leftrightarrow (w_h \sqsubseteq w_g) \wedge (t_h = t_g)$$

During the interaction k the bridges are stored in the set B_k and are used to translate the calls to the reasoning layer.

Framework

An external term t is mapped only when encountered during an interaction. Initially it can only guess at the proper bridges between t and the terms in its own ontology. These hypotheses must be verified, and the most likely is kept.

Without any *a priori* knowledge any possible mapping could be the correct one, and the hypotheses cover all the possible bridges between the external term t and the terms w_i. To make on-the-fly mapping feasible, the number of hypotheses should be drastically reduced.

In the framework, unlikely hypotheses are pruned by the *filter* elements using heuristics based on the experience of past interactions and on the context of the current interaction. The filters aim to minimise, on average, the number of wrong hypotheses to check.

Once the hypotheses are filtered, the *rule* elements generate an arguments in favour or against a hypothesis. The arguments are then combined by the framework to give an overall confidence level for the hypothesis.

Rules can exploit algorithms developed for static mapping, such as S-Match (Giunchiglia, Shvaiko, & Yatskevich 2004), to compute the matching, as the number of useless mappings to verify is reduced by filters.

A generated *argument* is a proposition coupled with two degrees of confidence, one that the proposition is true, and one that the proposition is false. Arguments are organised in a tree: the root is the hypothesis to verify, supported or attacked by the arguments in the nodes. An argument may recursively need other arguments to support it.

Framework explained

The mapping process is iterative. At every iteration i a semantic bridge b_{tki}, more specific than the bridge $b_{tk(i-1)}$ from the previous iteration, is created for the term t:

$$b_n \preceq b_{n-1} \preceq \ldots \preceq b_1$$

At every iteration i the function executes three steps:

- generates hypotheses,
- filters hypotheses and keeps the most probable ones,
- collects evidence for the remaining hypotheses, selects the most reliable hypothesis.

The loop ends when it becomes impossible to generate hypotheses that imply those proved in the previous step, or none of the hypotheses generated can be proved. The bridge created in the last iteration, and therefore the most specific, is returned and added to the set B_k.

Generate the hypotheses

At this step of each iteration i, the system receives the external term t and the mapping $b_{tk(i-1)}$ proved in the previous iteration, and returns a set of hypotheses Ω about the most generic mappings that imply $b_{tk(i-1)}$.

For example, if $b_{tk1} = \langle t \sqsubseteq w_1 \rangle$, $b_{tk2} = \langle t \sqsubseteq w_2 \rangle$, given the ontology in figure 1, then for the iteration #3:

$$\Omega_3 = \{ \langle \{\sqsubseteq, \sqsupseteq, \equiv\}, t, w_5 \rangle, \langle \{\sqsubseteq, \sqsupseteq, \equiv\}, t, w_6 \rangle \}$$

Filter the hypotheses

In the this step, the system combines different filters and produces an argumentation tree for each of the hypotheses selected from the set Ω generated in the previous step:

A filter f_i is characterised by its *breadth* and its *confidence*. The first is the “band-pass” of the filter: the narrower the filter, the fewer hypotheses are left to verify. If none of the filtered hypotheses could be proved, this step is repeated and the narrowest filter used previously is removed. The second indicates how likely is it that the correct hypothesis is in the selected subset. It is used as the first argument added to the argument tree of each filtered hypothesis.

After a term is successfully mapped, the filter receives the new bridge as feedback, and uses it to improve its predictive capability.

Select the best hypothesis

In this step, the system processes the set of hypotheses trees generated by the previous step, and tries to extract the most likely one. If the system fails to select any hypothesis, it goes back to the previous step, relaxes the filter if possible, and tries to obtain a wider set of hypotheses.

This step is composed of three actions.

Collect evidence For each hypothesis the system generates arguments using rules. A rule r_i is characterised by two confidence levels, that measure how strong or weak is the support or the attack of the generated argument: $c(hp|arg)$ is the confidence that the hypothesis is true, given that the argument is true, while $c(\neg hp|\neg arg)$ is confidence that the hypothesis is false, given that the argument is false. The argument is produced by an external function, that receives as input the hypothesis and a set of information specified in the rule. The information is collected by the system, and it is relative to the terms in the hypothesis: it can be the superclass, or the subclasses or the instances of one of the term. Information about internal terms is easily accessible, while the agent may ask the information about external terms to the other agent if it is not contained in the published ontology. External information may trigger further mapping to allow reasoning over the imported terms.

Combine evidence The arguments in the tree are combined to obtain two confidence levels for the hypothesis: one that the hypothesis is true, and one that it is false.

If a hypothesis is supported by one argument, the confidence that the hypothesis is true is computed as:

$$c(hp) = c(hp|arg)c(arg)$$

where $c(hp|arg)$ is given by the rule and $c(arg)$ is computed for the argument. Similar considerations apply for the confidence of the hypothesis being false. When there is more than one argument, the confidences must be combined.

It cannot be assumed that the confidences sum to one: if a rule establishes that a hypothesis is true with 0.4 of confidence, it does not imply that the hypothesis is false with 0.6 of confidence. Therefore, the theory used to combine confidences should be able to express ignorance about the truth value of the hypothesis.

One possible approach is Dempster-Shafer theory (Yager 1994), which computes the probability of a proposition supported by evidences. Following Dempster-Shafer theory, $c(hp)$ is interpreted as the *belief* that the hypothesis is true. While $1 - c(\neg hp)$ is the *plausibility* the hypothesis is true which is the extent to which the available evidence fails to refute the hypothesis. The interval between the two values is the ignorance interval.

The theory provides a formula, called Dempster's rule of combination, to combine evidences for a proposition.

Harvest Hypothesis At the end of an iteration *i* , the hypothesis with the highest confidence is selected. In some cases there might be more than one hypothesis within a narrow band of confidence: in this case the system first tries to apply more rules - if available - to gather more evidence for the conflicting hypotheses. If no rules can be applied, the strongest hypothesis is selected. Then the procedure restarts, until no more hypotheses can be generated.

Possible models of filters

Filters

Filters must operate rapidly, making it difficult to apply complex, symbolic or inductive inference methods. Nevertheless, filters can exploit the large volume of event-based data from the interactions and determine statistical patterns threaded in the dialogues.

Statistical contexts

A possible pattern to recognise is that some terms tend to appear together in interactions: some of these terms are contextual to the topic of the conversation (*buy*, *computer,...*), while other terms are auxiliary to any kind of conversation (*ask*, *inform,...*).

Following this intuition, the terms can be clustered together, and each cluster is a possible context for an interaction. The contexts are created and updated using the feedback from the framework. The contexts are used to classify dialogues as they unfold, and to predict which are the most likely terms that can occour during a conversation. This excludes hypotheses relative to terms that have never appeared in the context.

Formal description More formally, a *context* is a triple:

$$C = \langle id, N, S \rangle$$

where N is the number of dialogues classified by the context and S is the set of internal term elements η that distinguish the context.

Each term element η_i in S is a pair:

$$\eta_i = \langle w, \mu_C \rangle$$

where w is the term in the agent's ontology and μ_C is the grade of membership of the term in the context: terms may appear in different contexts with different frequencies.

Related to the grade of membership of a term there is the function $\mu_C(K)$ that returns the grade of membership to a context C of a set K of terms:

$$\mu_C(K) = \frac{1}{|K|} \sum_{w \in K} \mu_C(w)$$

How they are used Contexts are used to classify dialogues as they are performed. Every time a new term is mapped, the system tries to classify the dialogue finding the contexts that maximise the function $\mu_C(W)$, where W is the set of the internal terms in the bridges contained in B_k.

At the beginning of the mapping process, few terms are mapped and it is difficult to classify the dialogue properly, because more than a context can do it. As the dialogue unfolds, the number of terms in W increases and the number of contexts that can classify them is reduced.

The contexts that classify the dialogue are used to filter the generated hypotheses set: if some terms in the set never appear in the contexts, then it is possible to exclude these hypotheses, adding evidence for the remaining hypotheses.

How they are created When a dialogue is finished, the internal terms W are added to the context that better classifies them. If no context classify them well enough, then a new context is created.

Past mapping experience

Another possible pattern to identify is that some external terms tend to have always the same semantic relations with the same terms in the agent's ontology.

Formal description The set of previous mappings Λ contains a tuple λ_i for each mapping proved in the past, composed of three elements:

$$\lambda_i = \langle b, s_m, n_a \rangle$$

b is the hypothesis proved in the past, s_m is the cumulative confidence of the hypothesis, and n_a is the number of time the term mapped in the hypothesis has appeared in dialogues.

How they are used When the system must select hypotheses for an external term, it can look in past mappings for the term. It then keeps the hypotheses implied by the past mappings, and discards the others.

For the ontology in figure 1, given the generated hypotheses $\Omega = \{\mathtt{t} \sqsubseteq \mathtt{w}_2, \mathtt{t} \sqsubseteq \mathtt{w}_3, \mathtt{t} \sqsubseteq \mathtt{w}_4\}$ and the set of past mappings $\Lambda = \{\langle \mathtt{t} \sqsubseteq \mathtt{w}_5, 4, 5\rangle\}$, the filter should keep only $\mathtt{t} \sqsubseteq \mathtt{w}_2$ as the past mapping $\mathtt{t} \sqsubseteq \mathtt{w}_5$ implies it.

How they are created Mappings established for a particular external ontology and received as feedback from the framework are stored for future use. When mappings are encountered repeatedly, the confidence s_m in the past mapping λ_i is increased by the confidence in the bridge.

There is no issue about inconsistency, as conflicting past mappings are used only as suggestions about the order in which the hypotheses should be checked: conflicting hypotheses are tolerated by collecting evidence in favour or against them.

Related work

The COMA project (Do & Rahm 2002) is focused on combining different matchers to obtain a plausibility level for the computed correspondences. It introduces the reuse of past mappings, although for a different purpose. The abstraction of the argumentation tree in this paper subsumes the distinction between simple matchers and hybrid matchers. The QOM project (Ehrig & Staab 2004) addresses the trade off between efficiency and quality, introducing the concept of filtering the mapping candidates before verifying them with similarity comparators. However, the filtering is based only on properties of the ontologies (labels of nodes or hierarchy): it is does not exploit gained experience and it is not concerned about the purpose of the mapping as a mean to prune useless candidates.

In fact, both projects are oriented toward mapping whole ontologies, without any reference to interactions and their contexts.

Conclusion

In this paper we proposed a framework that allows agents with different ontologies to interact in order to perform a task. The mutual understanding during the interaction is reached dynamically mapping the ontologies. The framework exploits both the structure of the ontologies and statistical patterns threaded in the dialogues to produce heuristics used to improve the work of standard mapping algorithms.

The first advantage is that there is no need to foresee and map in advance all possible combinations of ontologies, because mappings take place only when needed. The second is that part of the algorithms developed and tested for this task can still be used.

This research is at a very early stage, and many details still need to be studied in depth.

References

Do, H. H., and Rahm, E. 2002. Coma - a system for flexible combination of schema matching approaches. In *VLDB*, 610–621.

Ehrig, M., and Staab, S. 2004. Qom - quick ontology mapping. In *International Semantic Web Conference*, 683–697.

Giunchiglia, F.; Shvaiko, P.; and Yatskevich, M. 2004. S-match: an algorithm and an implementation of semantic match. In *In Proceeding of the European Semantic Web Symposium*, 61–75.

Giunchiglia, F. 1992. Contextual reasoning. Technical report, IRST, Istituto per la Ricerca Scientifica e Tecnologica.

Hameed, A.; Preece, A.; and Sleeman, D. 2003. *Ontology Reconciliation*. Germany: Springer Verlag. 231– 250.

Kalfoglou, Y., and Schorlemmer, M. 2003. Ontology mapping: the state of the art. *Knowledge Engineering Review*.

Nuno, S., and Rocha, J. 1999. Mafra - an ontology mapping framework for the semantic web. In *Proc. of the 13th European Conf. on Knowledge*.

Yager. 1994. *Advances in the Dempster-Shafer Theory of Evidence*. John Wiley, New York.