How Service Choreography Statistics Reduce the Ontology Mapping Problem

Paolo Besana and Dave Robertson

School of Informatics, University of Edinburgh

Abstract. In open and distributed environments ontology mapping provides interoperability between interacting actors. However, conventional mapping systems focus on acquiring static information, and on mapping whole ontologies, which is infeasible in open systems. This paper shows that the interactions themselves between the actors can be used to predict mappings, simplifying dynamic ontology mapping. The intuitive idea is that similar interactions follow similar conventions and patterns, which can be analysed. The computed model can be used to suggest the possible mappings for the exchanged messages in new interactions. The suggestions can be evaluate by any standard ontology matcher: if they are accurate, the matchers avoid evaluating mappings unrelated to the interaction.

The minimal requirement in order to use this system is that it is possible to describe and identify the interaction sequences: the OpenKnowledge project has produced an implementation that demonstrates this is possible in a fully peer-to-peer environment.

1 Introduction

Most ontology mapping systems [9,15] available for the semantic web and for semantic web services focus on acquiring *static*, *a priori* information about mappings. Depending on the approach, matchers compare labels, ontology structures [10], use external dictionaries like WordNet to prove similarity between nodes in hierarchies [7], learn how instances are classified to find similarities between concepts [5] or combine information from different sources [4,6]. In an open and distributed environment ontology mapping systems aim at providing interoperability between interacting actors, each with possibly a different ontology. Mapping in advance, before the interactions, is unfeasible, as the agents may be still unknown. Mapping during the interactions may be computationally difficult, as many interactions with different actors can go on simultaneously.

This paper shows that the interactions between the actors can be used to predict the mappings, making the problems related to dynamic ontology mapping more tractable. The intuitive idea is that interactions follow conventions and patterns, and these patterns are repeated when similar situations arise. The patterns are extracted by analysing the interactions in order to model the relations between the terms that appear in them. If the computed model is representative of a class of interactions, then it can provide the basis for predicting the content

K. Aberer et al. (Eds.): ISWC/ASWC 2007, LNCS 4825, pp. 44–57, 2007.

of exchanged messages in future interactions. A prediction is a list of suggestions for the mapping, that any standard ontology matcher can evaluate. If the predictions are accurate, the matchers can avoid evaluating mappings unrelated to the interaction, improving efficiency and decreasing ambiguity. In fact, the context of the interaction provides additional information, that can help in those cases in which the matchers do not have enough static information to distinguish the correct mapping among many possible ones.

This paper shows that, after a reasonably small number of interactions the predictor consistently provides reliable suggestions. The minimal requirement in order to use this system is that it is possible to describe and identify the interaction sequences. In principle, any system based on workflow language can provide this. Workflow systems normally are centralised, but we have recently shown, in the EU funded OpenKnowledge project [1] that it is possible to achieve peer-to-peer based workflow systems, as a means of web service choreography.

In this paper, we first describe, in Section 2, the intuitive notions of dialogue and interaction behind our work; then, in Section 3, we briefly discuss the alternative approaches for agents communication, introducing the OpenKnowledge peer-to-peer framework for defining and executing interactions. Section 4 defines the concepts and terms used in modelling the context of interactions, while Section 5 describes what needs to be modelled, and how to model it with an example. Section 6 defines what needs to be evaluated, then reports how the testing was structured and next presents and interprets the results.

2 Services' Interactions

Many activities require interaction between different actors: for example, in order to book a room for a conference an inquirer needs to contact a travel agency (or more than one) or directly a number of hotels.

In the simplest version, communication between two agents is a message transmitted from a sender to a receiver. According to the speech act theory, a message is a performative act that changes the state of the world [14]. For example, a message sent from agents i to agent j to *inform* about ϕ will likely change the beliefs of j, adding the belief about ϕ. In our example, the following message, sent from Mr Smith's agent to the agent representing the hotel Y:

```
inform(booking, 11 Nov 2007, 15 Nov 2007, Mr Smith, single)
```

should make the hotel agent believe that a single room must be reserved for Mr Smith from the 11^{th} to the 15^{th} of November. Or at least this is what Mr Smith thinks. But, for example, the hotel agent may not know the meaning of `booking` or `single`, or it may use a different format for dates. To overcome this problem, either all agents that contact the hotel service must share the same ontology, which is not feasible in an open environment where agents from different backgrounds may interact, or the agents must have access to the mappings between different ontologies.

[1] `www.openk.org`

Unfortunately, it is infeasible in an open system to precompute all mappings, as it is impossible to forecast which agents will contact the hotel service, so some (or perhaps all) mappings must be computed dynamically when the interaction takes place. Many different ontology mapping systems have been developed and tested. However, the use of these systems raises two problems, both connected to the fact that they map full ontologies. Firstly, ontologies can be large, and the process can be lengthy, making it difficult to perform at run time, possibly simultaneously with other interactions that require other mappings. Secondly, ontologies often overlap only partially, and evaluating the result of the mapping process can be hard: one mapping process between two ontologies may yield 15% of coverage, while another one between different ontologies may yield 80% coverage, but the mappings in the first case may be the ones needed for an interaction, and those in the second case may be unrelated to it (this only being knowable at interaction time).

Usually interactions are more complex than single messages. Mr Smith's agent may first check the availability of offers, or it may want to first try single and then double rooms. Moreover, the booking may require a deposit or a credit card number. This increased complexity, consisting in exchanges of messages, follows rules and conventions: as the conversation unfolds, the content of new messages is bound by the previously exchanged messages. A message failing to follow these rules would surprise the hearer as being at best off topic or even incomprehensible.

Dialogue norms and conventions appear at syntactic level: a request is normally followed by an answer, an offer by an acceptance or a rejection. They can also be found at semantic level: the topic of a conversation tends to remain consistent over a number of messages, forming a sort of "local" context to the conversation - for example when the agents are discussing about the purchase of a specific product. The intuitions about syntactic norms has prompted researchers in NLP to study the possibility of *dialogue grammars*, that have often been represented as finite state machines, where the speech acts are the transition states between admissible states of the dialogue. The use of models for dialogues has been used, for example, in dialogue translation [12].

The content of interactions (even the simple ones consisting of single messages) are also influenced by the "real world" context - agents still act on the behalf of real actors - which influences the likelihood of some specific topics (tastes and needs change with geography and time). The travel agency context binds already the possible content of the interaction (you may ask for a flight, for a holiday package, but you will unlikely ask for a laptop)

For example, after entering in a travel agency and asking for an accommodation somewhere, the clerk will likely ask the requested period, and then will try to refine the request proposing different types of accommodations (hotels, B&B, hostels, and so on). Once accommodation has been addressed, the content of the possible messages is further constrained: talks about cruises are still possible, but less likely. However, the list of proposed accommodation may change, depending on the deals that the agency has, or more general on the country and

period (for example, B&B are starting to appear in Italy, but were extremely rare until a few years ago, while they are very common in the UK).

Interactions aimed at performing tasks tend to be repeated fairly similarly every time the same task needs to be done: the structure of the interaction in the example will be repeated similarly in different travel agencies. The repetition of interactions offers the possibility of extracting patterns in the interaction, providing the basis for a representation of context. This representation can be used for focusing the mapping process.

3 Dialogues and Protocols

Dialogues between software agents are, at least at the moment, simpler and more restricted than those between humans: they are carried out in order to reach a goal (buying a product, booking a flight, querying a price, ...) and there is no need to care for digressions, unless relevant to the task. Therefore, their grammars can be simpler than those required for human interactions.

Interactions can be hard-coded in the agents involved in them; may be planned dynamically; or defined in workflows that are followed as a script by the agents. These approaches offer different trade-offs between flexibility and efficiency: embedding the interactions in the agents is the most inflexible but possibly very efficient. Planning is based on the the idea that speech acts can be considered as actions that change the world, and have preconditions and postconditions, usually relative to the mental state of the agents involved in the interaction [3]. It offers the maximum flexibility but may require hefty computation at every interaction, and conditions can be difficult to verify. However, as we have argued, interactions tend to be repeated, so planning them every time is a waste of resources: workflows represent a good compromise and are currently the dominant solution.

Workflows can be seen as a sort of explicit, more stringent dialogue grammar, that agents follow. Their speech acts represent the transitions (or moves) between the different state in the dialogue.

Most workflow languages represent interactions between processes, and can be formalised using process calculi (such as π-calculus [11]). The *Lightweight Coordination Calculus (LCC)* [13] is based on π-calculus and can be used as a compact way of representing workflows. It is also executable and it is adapted to peer-to-peer workflows. In the original version, protocols are declarative scripts, circulated with messages. Agents execute the protocols they receive by applying *rewrite rules* to expand the state and find the next move.

It uses roles for agents and constraints on message sending to enforce the social norms. The basic behaviours are to send ($\Rightarrow$) or to receive ($\Leftarrow$) a message. More complex behaviours are expressed using connectives: `then` creates sequences, `or` creates choices.

The protocol in Figure 1 shows the initial part of a protocol describing an interaction between a customer and a supplier of some product. In the fragment, the customer asks for a product and then the supplier verifies if the request must

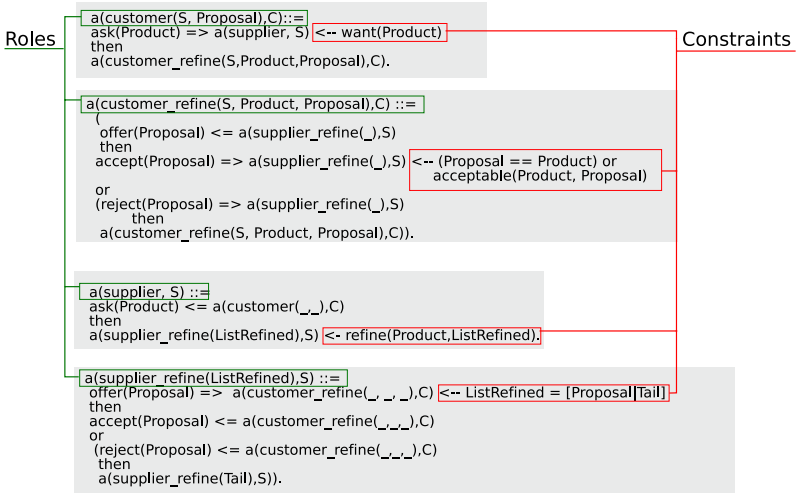

Fig. 1. Request refinement in LCC

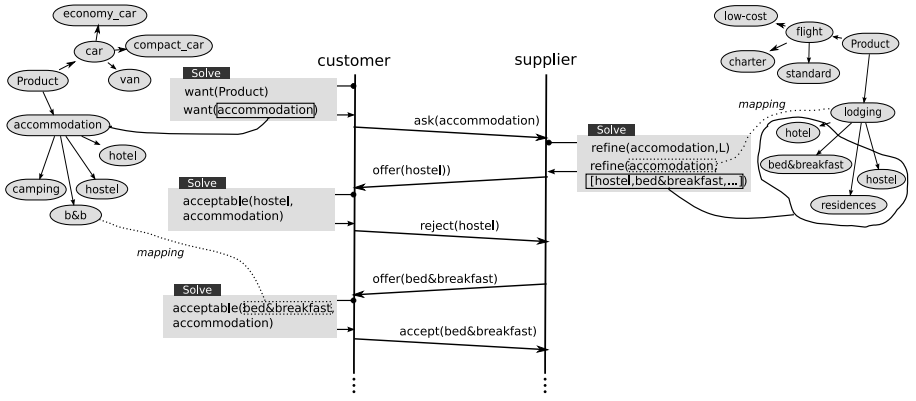

Fig. 2. Run of the protocol for searching an accommodation

be refined. If this is the case, the supplier will propose to the customer another, more specific, product. The customer, in turn, will analyse the proposal and see if it fits its needs. Figure 2 shows a run of the protocol in an interaction between a customer and a travel agency for booking an accommodation.

While the example has been kept simple for explanation purposes, a real interaction could be far more complex, involving many agents: LCC has been used in applications such as business process enactment [8] and e-science service integration [1]. In particular, it has been chosen as the specification language used for defining interaction models in OpenKnowledge, which aims at creating a Peer-to-Peer system that is open in participation, functionality and data and should allow people to easily create, find, invoke, compose and run services in a decentralised and autonomous fashion.

3.1 Open Knowledge Kernel

The core concept in OpenKnowledge are the interactions, defined by *interaction models* written in LCC and published by the authors on the *distributed discovery service* with a keyword-based description. The roles in the interaction models are played by peers. The peers that want to perform some tasks, such as booking a room or provide a booking service, use keyword queries to search for published interaction models for the task, and then advertise their intention of interpreting one of its roles to the discovery service. In the running example, a travel agency subscribes to perform role `supplier`, while a peer searching a room subscribes as `customer`.

When all the roles are filled, the discovery service chooses randomly a peer in the network as coordinator for the interaction, and hands over the interaction model together with the list of involved peers in order to execute it.

The coordinator then asks each peer to commit to the interaction. If they all commit, the coordinator executes the interaction instantiating a local proxy for each peer: the peers are contacted to solve constraints in the role they have subscribed. In the example protocol, the coordinator will ask the peer that has subscribed as `customer` to solve `want(Product)`.

4 Predicting the Content of Messages

A message in an interaction is a tuple, whose elements convey the content of a single communication act:

$m_i = \langle s_1, ..., s_n \rangle$

A term s_i is introduced when a constraint in a role is satisfied by one of the actors playing the role (in the example shown in Figure 2, *"accommodation"* is introduced by the customer peer, satisfying the constraint `want(Product)`). The term s_i is defined in the ontology of that agent, and refers to an entity Q_k. The other agents, if they need to satisfy a constraint that contains s_i, will need to find the term t_m in their ontology that refers to the same, or a similar, entity Q_k (in the example, in order to satisfy the constraint `refine(Product, ListRefined)`, the supplier must map the term *"accommodation"* to *"lodging"* in its ontology). The mapping is performed by a "mapping oracle", whose specific implementation is irrelevant for this work: any existing mapping system, such as S-Match [7], would fit smoothly in the framework.

Let us suppose that a peer, with ontology L_a, needs to satisfy a constraint $\kappa_r(\ldots, w_i, \ldots)$ when in a specific state of an interaction, and that $w_i \notin L_a$ is the foreign term received in some previous message m_j. The task of the oracle is to find what entity Q_k, represented in the agent's ontology by the term $t_m \in L_a$, was encoded in w_i.

Definition 1. *The intended entity Q_k represented in the argument of the constraint by the foreign term w_i is, from the agent's perspective, a random variable, whose domain is the whole ontology.*

As said before, an ontology mapping algorithm can be used to interpret the sign w_i in the message and map it to the corresponding symbol t_m.

Definition 2. *The term t_m is the matching term: it is, in the agent's ontology, the closest to the intended entity Q_k. For the current work, the matching term is assumed to exist in L_a. The assumption is based on the weakness of the relation between t_m and Q_k: it is sufficient that the meanings are close enough to perform the interaction.*

However, conventional ontology mapping algorithms do not take into account the context of the interaction, and consider all the terms in the domain as equiprobable:

$p(Q_k = t_i) = p(Q_k = t_j)\ for \forall t_i, t_j \in L_a$

As introduced earlier, dialogues follows conventions and rules, made explicit by the protocol, and the content of the messages are influenced by the local and the general context: therefore the terms are not equiprobable - some will be more likely than others.

Definition 3. *The random variable Q_k has a conditional probability distribution, where the evidence is the context of the interaction (we discuss context in Section 5):*

$\mathbf{P}(Q_k | context) = \langle \ldots, p(Q_k = t_i | context), \ldots \rangle \quad for \quad t_i \in L_a$

where $p(Q_k = t_i | context)$ is the probability that t_i is the best matching term for Q_k given the current state of the interaction and the history of previous interactions.

The knowledge of the probability distribution of a variable is used to predict the possible values of Q_k selecting a subset of likely terms to be verified by the oracle, improving the efficiency and the results of the ontology mapping systems, and making it more feasible to be performed at runtime.

Definition 4. *A subset $\Lambda \subseteq L_a$ is a set of terms containing the most likely terms for a random variable Q_k. The probability that the correct matching term t_m belongs to Λ is:*

$p(t_m \in \Lambda) = \sum_{t_i \in \Lambda} p(Q_k = t_i | context)$

For $\Lambda = L_a$, this probability is 1. If the distribution is uniform, then a set Λ of size $\gamma |\Lambda|$, with $0 \leq \gamma \leq 1$, will contain t_m with probability $p(t_m \in \Lambda) = \gamma$. If the distribution is non-uniform, then even for smaller resizing factor γ it is possible to obtain high probabilities $p(t_m \in \Lambda)$: it becomes useful to trade off between the size of the set Λ and the probability of finding the correct mapping. To select the terms to insert in Λ, it is necessary to set a threshold $\tau < 1$ for $p(t_m \in \Lambda)$. If the terms are ordered from the most probable to the least one, then this means solving the equation in n:

$\tau \leq \sum_1^n p(t_j)$

That simply means taking the first n most likely terms until their cumulative probability is equal or greater than τ.

5 Modelling the Context

5.1 What to Model

An interaction is an exchange of messages, where the content of the messages comes from satisfying constraints. A peer satisfying a constraint is responsible for the introduction of terms related to the interaction: failure to do so disrupts the communication. If the travel agency peer, after being asked for an accommodation, satisfies the constraint `refine(Product, ListRefined)` with a choice of possible types of coffee, then the communication loses meaning. Therefore, what the predictor should model are constraints. Intuitively, constraints fall into three main categories:

- *Purely functional*: given a set of parameters, they always unify with the same values: for example $multiply(X, Y, Z)$ is supposed to always unify the variables with the same numbers.
- *Purely "preference-based"*: they collect requests from users and their possible values can differ every time. In the example, the constraint `want(Product)` is preference-based: each peer will satisfy it according to its tastes and needs. Overall, the variables in preference-based constraints will have an unknown distribution. These distributions may change with time, depending on general shifts of "tastes" and "needs" (fashions, trends, fads, ...) or the heterogeneity in the peer group composition, and can be more or less biased.
- *Mixed*: they can be mainly functional, but the results may change depending on external factors (availability, new products appearing on the market, etc), or can be mainly preference-based, but constrained by some other parameters. In the example, the constraint `refine(Product,ListRefined)` is mainly functional, as it returns the list of possible subclasses of a term if the query can be refined. The list of terms can however change depending on the specific peer and with time.

A purely functional constraint, when the function is ontological (obtain subclasses, or siblings, or properties), can be guessed and the hypotheses can be verified comparing the guesses with the feedback from the ontology matching process. For the purely preference based, it is possible to count the frequencies of the terms and learn their prior probability distribution. For the mixed, it is possible to use a mix of hypotheses and counting the frequencies. Sometimes the ontology of the peer does not allow him to formulate the correct ontological relation (because the ontology is structured differently from the agent that introduced the term): it is still possible to count the conditional frequencies, modelling the relation from a purely statistical point of view.

5.2 How to Model

Our solution, suggested but not evaluated in [2], is a model of the interaction in which the properties of entities appearing in the random variable Q_k in different runs of the same protocol are counted and stored in *assertions:*

Table 1. Types of assertions

Frequency of terms: $Freq\,(E_i \in \{t_q\})$
Assertions can be about the frequency of the entities in an argument, disregarding the content of other variables in the dialogue, like A_{1-5} in Table 2.

Conditional frequency of terms: $Freq\,(E_i \in \{t_q\} \mid E_k = t_h)$
More precise assertions can be about the frequency of an entity given the content of previously encountered variables, like A_{6-10} in Table 2.

Frequency of relations with terms in other variables: $Freq\,(E_i \in \{X \mid rel\,(X, E_k)\})$
They can regard the relation with an argument of another variable in the protocol, like A_{11-12} in Table 2.

Frequency of relations with terms in ontology: $Freq\,(E_i \in \{X | rel\,(X, t_k)\})$
They can be about an ontological relation between the entity in the argument and an entity t_k in the agent's ontology, like A_{13} in Table 2.

Definition 5. *An assertion about a random variable Q_k keeps track of the frequency with which the entity has been part of a set Ψ in the encountered dialogues:*

$$A_j \doteq Freq\,(Q_k \in \Psi) \tag{1}$$

Assertions can be about frequencies of terms in the variable, or can be about the frequencies of ontological relations between one variable and another, as described in Table 1.

For example, the customer peer, having executed the interaction in Figure 1 a number of times with different types of service providers, will have a table with assertions about the content of the variable $Proposal_k$ in the form shown in Table 2. As protocols can be recursive, the variables are tagged with their appearance in the run (in the example, the variable *Proposal* is used twice, so there will be two random variables named $Proposal_1$ and $Proposal_2$).

When the content of a variable must be predicted, the assertions relative to it are instantiated with the current state of the the interaction. In the interaction shown in Figure 2, in order to predict the content of $\texttt{Proposal}_2$ (received in the second offer sent by the travel agency), given that $\texttt{Product}_1$ was instantiated to *"accommodation"* and $\texttt{Proposal}_1$ was instantiated to *"hostel",* it is necessary:

1. to drop the conditional assertions whose evidence does not correspond to the current state of the interaction; so assertions A_{9-10} are dropped because their evidence $\texttt{Product}_1 =$ "*car*" is inconsistent with the current interaction,
2. to unify the variables in relations with the current state of the interaction ; $\texttt{Product}_1$ in A_{11} is replaced with *"accommodation"* and $\texttt{Proposal}_{k-1}$ in A_{12} is replaced with *"hotel"*, obtaining:
   $A_{11})\ Freq(\texttt{Proposal}_2 \in \{\texttt{Proposal}_2 : \texttt{subClassOf}(\texttt{Proposal}_2, \text{“}accomodation\text{”})\})$
   $A_{12})\ Freq(\texttt{Proposal}_2 \in \{\texttt{Proposal}_2 : \texttt{siblingOf}(\texttt{Proposal}_2, \text{“}hotel\text{”})\})$
3. the relations are computed, obtaining sets of terms; so A_{11-13} becomes:
   $A_{11})Freq(\texttt{Proposal}_2 \in \{\text{“}hostel\text{”}, \text{“}hotel\text{”}, \text{“}b\&b\text{”}, \text{“}camping\text{”}\}) = 24$
   $A_{12})Freq(\texttt{Proposal}_k \in \{\text{“}hotel\text{”}, \text{“}b\&b\text{”}, \text{“}camping\text{”}\}) = 24$
   $A_{13})Freq(\texttt{Proposal}_k \in \{\text{“}accomodation\text{”}, \text{“}hotel\text{”}, ..., \text{“}car\text{”}, ..., \text{“}van\text{”}\}) = 24$

Table 2. Statistical model of the context for the customer peer

$A_1)\ Freq(\texttt{Proposal}_k \in \{\text{“}hotel\text{”}\}) = 6$
$A_2)\ Freq(\texttt{Proposal}_k \in \{\text{“}hostel\text{”}\}) = 6$
$A_3)\ Freq(\texttt{Proposal}_k \in \{\text{“}b\&b\text{”}\}) = 4$
$A_4)\ Freq(\texttt{Proposal}_k \in \{\text{“}compact\_car\text{”}\}) = 3$
$A_5)\ Freq(\texttt{Proposal}_k \in \{\text{“}economy\_car\text{”}\}) = 5$
$A_6)\ Freq(\texttt{Proposal}_k \in \{\text{“}hotel\text{”}\}\,|\texttt{Product}_1 = \text{“}accommodation\text{”}) = 6$
$A_7)\ Freq(\texttt{Proposal}_k \in \{\text{“}hostel\text{”}\}\,|\texttt{Product}_1 = \text{“}accommodation\text{”}) = 6$
$A_8)\ Freq(\texttt{Proposal}_k \in \{\text{“}b\&b\text{”}\}\,|\texttt{Product}_1 = \text{“}accommodation\text{”}) = 4$
$A_9)\ Freq(\texttt{Proposal}_k \in \{\text{“}compact\_car\text{”}\}\,|\texttt{Product}_1 = \text{“}car\text{”}) = 3$
$A_{10})\ Freq(\texttt{Proposal}_k \in \{\text{“}economy\_car\text{”}\}\,|\texttt{Product}_1 = \text{“}car\text{”}) = 5$
$A_{11})\ Freq(\texttt{Proposal}_k \in \{\texttt{Proposal}_k : \texttt{subClassOf}(\texttt{Proposal}_k, \texttt{Product}_1)\}) = 24$
$A_{12})\ Freq(\texttt{Proposal}_k \in \{\texttt{Proposal}_k : \texttt{siblingOf}(\texttt{Proposal}_k, \texttt{Proposal}_{k-1})\}) = 24$
$A_{13})\ Freq(\texttt{Proposal}_k \in \{\texttt{Proposal}_k : \texttt{subClassOf}(\texttt{Proposal}_k, \text{“}product\text{”})\}) = 24$

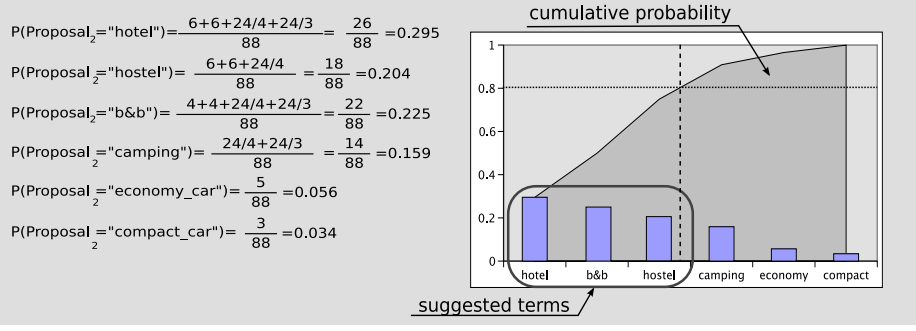

Fig. 3. Probability distribution for variable `Proposal`

The result of the third step is that some of the assertions assign probabilities to possibly large and overlapping sets. The frequencies assigned to sets are uniformly distributed among the members: according to the *principle of indifference* the frequency of mutually exclusive elements in a set should be evenly distributed. However, assertions about ontological relations create two main problems. First, some of the relations can be spurious. Second, some relations may refer to large sets, bringing little information (like assertion A_{13} in the example). To deal with the first issue, only relations found in a significant proportion of the cases are taken into consideration. To deal with the second sets larger than a significant portion of the ontology are discarded. Tests have shown that a threshold for the first issue of 10% and of 20% for the second one minimise the problem.

Finally, the probability that an entity t_i is used for Q_k is computed by summing the frequencies in all the instantiated assertions in which t_i appears, divided by the sum of the frequencies of all the selected assertions:

$$p\,(t_i) = \frac{\sum A_j (t_i \in \Psi)}{\sum A_k} \tag{2}$$

In the example, to compute the probability that the concept in $\texttt{Proposal}_2$ is the term *"hotel"* , the numerator contains the assertions A_1, A_6, A_{11},A_{12}. The assertions A_{11-12} contain more than one element, and therefore the frequency assigned to *"hotel"* is computed dividing the frequency assigned to the set by the size of the set to obtain the following:

$$P(hotel) = \frac{6+6+24/4+24/3}{6+6+4+3+5+6+6+4+24+24} = \frac{26}{88} = 0.295$$

The complete distribution of variable $P(Proposal_2 = \text{"}hotel\text{"}|Context)$ is shown in Figure 3.

6 Evaluation

The predictor is characterised by its average success rate, $E\left[P_Q(t_m \in \Lambda)\right]$, and the average size of the suggested set Λ, $E\left[|\Lambda|\right]$. Let us assume to have the exact probability distribution $\overline{\mathbf{P}}\left(Q_k\,|context\right)$ of the terms for a random variable Q_k given the current context. The correct size n of Λ in order to obtain the desired probability of finding t_m is:

$\tau = \sum_1^n \overline{p}\left(t_j\right)$

If the computed distribution $\mathbf{P}\left(Q_k\,|context\right)$ is a good approximation of $\overline{\mathbf{P}}\left(Q_k\,|context\right)$, then the average of $p(t_m \in \Lambda)$ should converge towards the average of $\overline{\mathbf{P}}\left(Q_k\,|context\right)$ and therefore towards the threshold τ:

$$\lim_{iterations \to \infty} E\left[p(t_m \in \Lambda)\right] = E\left[\overline{p}(t_m \in \Lambda)\right] = \tau \tag{3}$$

If the success rate of the predictor remains lower than the threshold τ, independently of the number of interactions, then the computed distribution is different from the exact $\overline{\mathbf{P}}\left(Q_k\,|context\right)$.

A key issue to evaluate is the number of repeated interactions needed for the predictor to reach a stable behaviour. This number will be different for every type of interaction, but what is necessary is to find its probability distribution: what is the probability that n interactions are enough to have a stable behaviour. Once in the stable region, the predictor will go on updating its representation, but the behaviour should change slowly or remain constant.

The size of the suggested set Λ will depend on the existence relations between variables in the interaction and on the unknown distribution of terms in preference-based constraints, as we have seen in Section 5.1. These unknown distributions can change over time - if the phenomena are non-stationary - obviously decreasing the success rate. The lack of relations or flat distributions will cause large suggestions sets Λ.

6.1 Testing

One way of testing is through real interaction scenarios, using real ontologies and real workflows for the dialogues, but since these are scarce this would cover only

$$
\begin{array}{l}
a(r8a(O), I) ::= \\
\quad m_1(X, P) \Rightarrow a(r8b, O) \leftarrow \kappa_1(P, X) \\
\quad then \left(\begin{array}{l} m_2(Y) \Leftarrow a(r8b, O) \\ or \\ m_3(M) \Leftarrow a(r8b, O) \end{array} \right) \\
a(r8b, O) ::= \\
\quad m_1(X, P) \Leftarrow a(r8a(\_), I) \\
\quad then \left(\begin{array}{l} m_2(Y) \Rightarrow a(r8a, O) \leftarrow \kappa_2(P, X, Y) \\ or \\ m_3(M) \Rightarrow a(r8b, O) \leftarrow \kappa_3(P, X, M) \end{array} \right)
\end{array}
$$

Fig. 4. Protocol template

part of the testing space, without having the possibility of varying parameters to verify the effects.

What is important, however, is to verify the ability of the predictor in statistically modelling the way constraints are satisfied given the state of the interaction. And, as we have seen in Section 5.1, the constraints can be *functional*, *preference-based*, or *mixed*. It is thus possible to simulate different real world scenarios using template protocols executed by dummy peers that can only satisfy constraints according to parametrisable rules and ontologies.

The template protocols must cover the basic patterns in interactions. The functional constraints are ontological rules, the preference-based constraints return terms according to probability distributions that reflect the distribution of "needs" and "tastes" over a community of peers, and mixed constraints are rules with an element of probability.

For example, the protocol in Figure 4 can model many different interactions: m_1 can be a request for information X about P (for example, the price of a X), with m_2 being the reply and m_3 being the apology for failing to know the answer. Alternatively, m_1 can be an offer (the product X at price P), with m_2 being the acceptance and m_3 the rejection. By viewing interaction protocols abstractly we can set up large scale experiments in which we vary the forms of constraints in a controlled way.

Testing has involved a number of different abstract protocols with different possible relations between the terms in variables. The protocols were run in different batches, with each of them consisting of 200 runs of the protocol. Each batch is characterised by a specific ontology (with a hierarchical depth) and a set of preference distributions. Every 10 runs, the average size and the average score are stored in order to obtain a graph representing the improvement of the results over time.

6.2 Results

The results shown in Figure 5 were obtained averaging the results of 12 different batches, generated combining 6 protocols, 3 ontologies (225, 626 and 1850 elements) and different settings for the preference distributions (narrow and wide

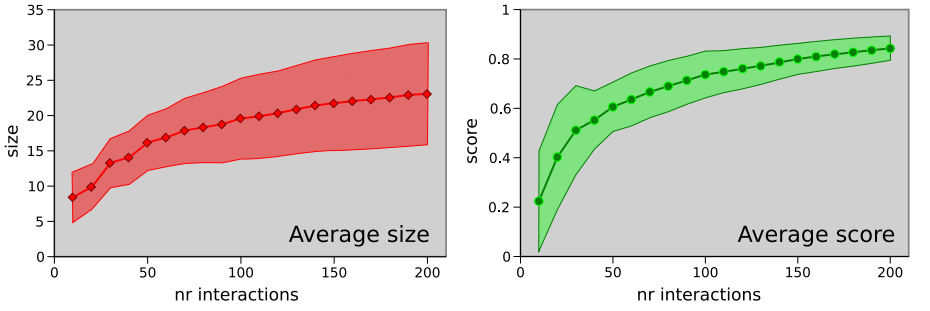

Fig. 5. Average size and average score

distributions for the preference-based constraints). All the batches were run with a threshold $\tau = 0.8$. The figure shows the average value of the size of the suggested set Λ and the average value of $p(t_m \in \Lambda)$, together with a band specifying the standard deviation of the measure. The limit in Formula 3 is verified, as the average score tends to stabilise, logarithmically, around τ (the standard deviation, showing fluctuations in success rate, decreases).

The average size remains small, independently of the size of the ontology, but its deviation tends to increase - albeit only logarithmically and remains well below 15% of the smaller ontology. The relatively large deviation reflects the fact that different batches have different relations between variable, and preference-based constraints have different distributions: therefore to obtain the same success rate the size of Λ may change meaningfully. However, the use of the filters on the assertions, described in Section 5.2, improved the results substantially: previous tests run on the same batches before the introduction of the filters returned the same average score, but a much higher average size (more than 150 elements instead of about 20).

The learning curve is, as stated, logarithmic: on average, most improvement (from 0 to nearly 70%) is obtained in the first 70-80 interactions, which is a small number of interactions in large peer-to-peer communities as those envisioned in the OpenKnowledge project. In the example scenario, the travel agency peer can be contacted by a thousand peers, all making similar requests, while the customer may need to contact several travel agencies before finding an proper accommodation.

7 Conclusion

This paper has shown that it is possible to use the interactions between peers in an open environment to statistically model and then predict the possible content of exchanged messages. The predictions can be forwarded to an ontology matching algorithm that focuses its computational effort on verifying the suggested hypotheses, without wasting time on evaluating mappings not related to the interaction.

The evaluation of the proposed method shows that a relatively small number of interaction is often enough to obtain a good success rate in the suggestions, especially when it is possible to detect ontological relations between terms appearing in the conversation.

The main requirement is to use a framework that allows the description of the interaction sequence: workflow based systems provide the functionality, but are often centralised. With the European project OpenKnowledge we have shown that these results can be obtained in a purely peer-to-peer environment.

References

1. Barker, A., Mann, B.: Agent-based scientific workflow composition. In: Díaz, J., Orejas, F. (eds.) CAAP 1989 and TAPSOFT 1989. LNCS, vol. 351, pp. 485–488. Springer, Heidelberg (1989)
2. Besana, P., Robertson, D.: Probabilistic dialogue models for dynamic ontology mapping. In: Fensel, D., Sycara, K.P., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, Springer, Heidelberg (2003)
3. Cohen, P.R., Levesque, H.J: Rational interaction as the basis for communication. Intentions in Communication, 221–256 (1990)
4. Do, H.H., Rahm, E.: Coma - a system for flexible combination of schema matching approaches. In: VLDB, pp. 610–621 (2002)
5. Doan, A., Madhavan, J., Dhamankarse, R., Domingos, P., Halevy, A.: Learning to match ontologies on the semantic web. The VLDB Journal 12(4), 303–319 (2003)
6. Ehrig, M., Staab, S.: Qom - quick ontology mapping. In: International Semantic Web Conference, pp. 683–697 (2004)
7. Giunchiglia, F., Shvaiko, P., Yatskevich, M.: S-match: an algorithm and an implementation of semantic match. In: Proceeding of the European Semantic Web Symposium, pp. 61–75 (2004)
8. Guo, L., Robertson, D., Chen-Burger, Y.: A novel approach for enacting the distributed business workflows using bpel4ws on the multi-agent platform. In: IEEE Conference on E-Business Engineering, pp. 657–664. IEEE Computer Society Press, Los Alamitos (2005)
9. Kalfoglou, Y., Schorlemmer, M.: Ontology mapping: the state of the art. The Knowledge Engineering Review 18(1), 1–31 (2003)
10. Melnik, S., H,, Garcia-Molina, E R.: Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In: ICDE 2002. Proceedings of the 18th International Conference on Data Engineering, p. 117 (2002)
11. Puhlmann, F., Weske, M.: Using the pi-calculus for formalizing workflow patterns. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 153–168. Springer, Heidelberg (2005)
12. Reithinger, N., Engel, R., Kipp, M., Klesen, M.: Predicting dialogue acts for a speech-to-speech translation system. In: Proc. ICSLP 1996, vol. 2, pp. 654–657 (1996)
13. Robertson, D.: A lightweight coordination calculus for agent systems. In: Declarative Agent Languages and Technologies, pp. 183–197 (2004)
14. Searle, J.R.: Speech acts: an essay in the philosophy of language. Cambridge University Press, Cambridge (1969)
15. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches. Journal on Data Semantics 4, 146–171 (2005)