MaLM: Machine Learning Middleware to Tackle Ontology Heterogeneity

Licia Capra
Dept. of Computer Science
University College London
Gower Street, London, WC1E 6BT, UK
L.Capra@cs.ucl.ac.uk

Abstract

We envisage pervasive computing applications to be predominantly engaged in knowledge-based *interactions, where services and information will be found and exchanged based on some formal knowledge representation. To enable knowledge sharing and reuse, current middleware make the assumption that a single, universally accepted, ontology exists with which queries and assertions are exchanged. We argue that such an assumption is unrealistic. Rather, different communities will speak different 'dialects'; in order to enable cross-community interactions, thus increasing the range of services and information available to users, on-the-fly translations are required. In this paper we introduce MaLM, a middleware for pervasive computing devices that exploits an unsupervised machine learning technique called* Self-Organising Map *to tackle the problem of* ontology heterogeneity. *At any given time, the MaLM instance running on a device operates in one of two possible modes: 'training', that is, MaLM is autonomically learning how to group together semantically closed concepts; and 'expert', that is, given in input a query or assertion expressed in a foreign dialect, MaLM identifies the concept, expressed in the device mother-tongue, that most closely represents it.*

1. Introduction

Pervasive computing is quickly turning from vision to reality. A wide range of devices, with different hardware, software and network capabilities, are available on the market and an always increasing number of people rely on one or more of them (e.g., mobile phones, PDAs, portable music players, etc.) to accomplish their daily task. The range and variety of public resources and services offered to, and being offered by, users of these devices will soon be overwhelming. Coordinating these entities has been recognised as a major challenge by middleware researchers and practitioners, partly due to high degrees of *heterogeneity* both in device capabilities, types of networks and services, and communication paradigms.

Portable devices come in many forms, from resource-poor to fully-fledged notebooks. As a result of mobility, they make and break connections with a degree of spontaneity not found before in other forms of networks; while moving around different areas, they are required to interact with different types of networks, such as WaveLAN, Bluetooth, GSM, and so on. They have to understand and use different communication paradigms (e.g., JEDI [9], Lime [17]); they are expected to talk different service discovery protocols (e.g., Jini [4], UPnP [18]). Various middleware systems have been proposed that tackle heterogeneity at different levels. For example, SATIN [20] is a light-weight, general-purpose component-based middleware that exploits logical mobility techniques to discover, download and deploy the components needed at run-time by the mobile device to reconfigure itself and interact in new settings. ReMMoC [13] and INDISS [6] are two different approaches to the more specific issue of interoperability among different service discovery protocols, where the former is based on reflection, while the latter exploits event-based parsing techniques.

We argue that there exists another form of heterogeneity implicit in pervasive systems that has not been investigated yet, that is, *ontology heterogeneity*. Pervasive systems appear as localised communities where services and resources are continuously offered and looked for. Devices will be predominantly engaged in *knowledge-based* interactions, where they will ask queries and give assertions about some domain of discourse, based on a formal knowledge representation, or ontology. Service discovery is an example of knowledge-based interaction, where a client device sends around a service request (i.e., query) and expects ser-

vice advertisements (i.e., assertions) in answer. Both the request and the advertisements refer to the same domain of discourse and are expected to be encoded using a formal representation (e.g., OWL-S [19]).

In order to enable device collaboration, current middleware make the assumption that a single, universally accepted, ontology exists with which knowledge is exchanged. We argue that this is an unrealistic assumption, in the same way as it is unrealistic to assume that all mobile devices will talk the same communication protocol over the same network technology. Rather, different communities will use distinct *dialects* to formalise their knowledge. Because of the autonomous nature of ad-hoc communities, these dialects are likely to contain homonym and/or synonym discrepancies, they are likely to evolve over time and may be subject to incremental changes [12]. In order to foster pervasive computing collaborations, cross-community interactions must be supported.

In this paper, we introduce MaLM, a middleware for pervasive computing devices that exploits an unsupervised machine learning technique called *Self-Organising Map* [15] to tackle ontology heterogeneity. MaLM assumes each device speaks a certain language, which we sometimes refer to as mother-tongue. In order to understand incoming queries and assertions, MaLM autonomically learns: (1) how to group together related concepts; (2) how to translate concepts from a foreign dialect to the device mother-tongue. It does so by means of *unsupervised* training, that is, from a software developer and end-user perspective, the learning and translation occur in a transparent manner.

The paper is further structured as follows: Section 2 motivates this research by illustrating two scenarios where ontology heterogeneity is a primary concern. In Section 3 we provide a short background to the machine learning technique we have deployed; we then discuss how this technique is used to enable MaLM autonomic training and translations. Section 4 discusses experimental results, Section 5 compares our work with others in the field, and finally Section 6 concludes the paper.

2. Motivating Examples

In this section, we describe two scenarios where the problem of ontology heterogeneity appears prominently: a pervasive service discovery setting, and a distributed trust management setting.

2.1. Ontology Heterogeneity in Service Description and Discovery.

Almost any service discovery protocol for pervasive computing is based on the following pattern: a client device A sends around a query, which contains a description

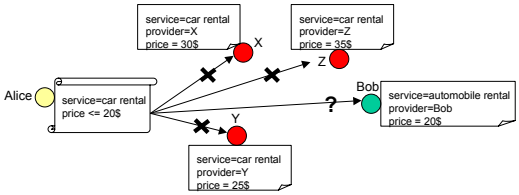

Figure 1: Ontology Heterogeneity in Service Description and Delivery.

of the service S that A is looking for. Such a description usually consists of a list of attribute-value pairs (a_i, v_i), where attributes and values are assumed to belong to a given ontology $\mathcal{O}$. In its simplest formulation, a query contains a single attribute, indicating the service type A is looking for; usually, additional attributes are included for more refined queries, thus constraining, for example, the location of the service, the QoS it offers, the cost, etc. The query is evaluated either by service providers or by intermediate nodes who maintain information about available services in the proximity; available services are described by means of a service descriptor, which is once again nothing but a list of attribute-value pairs (a'_i, v'_i), with attributes and values expressed in a given ontology $\mathcal{O}'$. Until now, one of the following two assumptions has been made by service discovery protocols: either they assume that $\mathcal{O} \equiv \mathcal{O}'$, or that it exists a statically known homomorphism ϕ such that $\phi(\mathcal{O}) = \mathcal{O}'$. In practice, that means that if any device speaks a language other than $\mathcal{O}$ (device mother-tongue) or $\mathcal{O}'$ (statically known dialect), then it will be excluded from potentially fruitful collaborations. Consider for example the scenario depicted in Figure 1, where Alice is looking for a car rental service provider. Among the group of providers speaking Alice's language, no-one exists that can provide the service requested; with available middleware technologies, the discovery would thus fail, even though qualified providers (i.e., Bob) are indeed available, but advertising their services using a different ontology.

2.2. Ontology Heterogeneity in Distributed Trust Management.

Distributed trust management systems are gaining momentum, as a mechanism for devices to make better informed decisions, for example, about who to interact with in a pervasive setting. These systems rely on the following idea: whenever a device A interacts with another device B, A updates its trust $t_{A,B} \in [0, 1]$ in B based on the outcome of the interaction; next time A has to make a decision about B (e.g., whether to cooperate with it), A's trust in B will be used to decide whether to proceed. If B is unknown to A, recommendations from devices that have

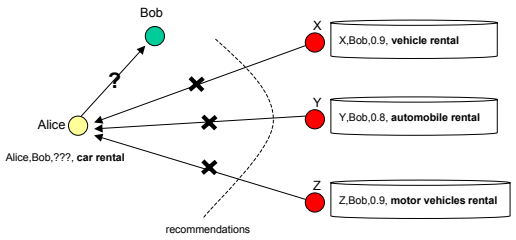

Figure 2: Ontology Heterogeneity in Distributed Trust Management.

interacted with B before are looked for. Behind this idea, a key assumption holds: all devices use the same ontology $\mathcal{O}$ to remember what is called the 'context' of the interaction; remembering the context is vital as B may be a trustworthy 'car rental' company, but might provide untrustworthy 'taxi service'. If only recommendations coming from devices who use the same ontology are taken into considerations, a large amount of information may not be processed, as exemplified in Figure 2.

Assuming the existence of a single, universally accepted ontology is unrealistic in these scenarios. The same could be said for traditional distributed systems; however, with pervasive computing, the problem becomes much more predominant for various reasons: first, the scale of a pervasive system is by orders of magnitude higher (in the number of devices and services), thus the assumption of having a universal ontology becomes even less plausible; second, there are no administrative boundaries within which to enforce an ontology, rather, each device could be considered a bounded domain of its own; finally, services and information are offered in a much more dynamic, open, and serendipitous manner, so that even static cross-ontology translations become inapplicable.

3. MaLM Middleware

We propose MaLM, a middleware that exploits machine learning techniques to learn how to perform on-the-fly translations across ontologies, removing the unrealistic assumption that devices exchange knowledge by means of a shared ontology (or a set of statically known ones). It does so by means of an unsupervised machine learning technique called the Kohonen's Self-Organising Map (SOM) [15]. Informally speaking, a SOM can be visualised as a two-dimensional structure (a rectangular grid), the cells (or nodes) of which represent word categories; words with related meaning are mapped on the same or spatially close nodes. No a priori information about categories is necessary; rather, a self-organizing process, governed by a neural network algorithm, learns how to organise conceptually interrelated words into the map. As a result, a model of the word categories emerges. The learning process is *unsupervised*, meaning that no teacher (i.e., the application developer or the end-user) is needed to define the node in the map (called winner) to which an input word is semantically closest. While more accurate results are expected to be obtained with supervised learning techniques, we chose to investigate unsupervised learning first, in order to avoid the burden that supervised learning imposes on the application developer (or end-user) during training periods.

In the reminder of this section, we first provide some background on the SOM technique; we then describe in details how SOM is used in MaLM.

3.1. Background: The Self-Organising Map

The Self-Organising Map is a very popular technique within the information retrieval community used to perform document classification. The problem could be formalised as follows: given in input a document set, create a two-dimensional map, so that conceptually related documents are mapped onto spatially close nodes. The map self-organises itself by means of an initial training period, during which each document belonging to a training set is processed as follows: the document is described by a real vector $x(t) \in R^n$, where t represents the t^{th} element in the training set; in practice, t could be seen as logical time, as at each step a new document from the set is processed. At bootstrap, each node i in the map is assigned a model vector $m_i \in R^n$, which has the same number of elements as the input vectors x, and whose values are chosen at random. For any input $x(t)$, the following two steps occur:

Step 1 - $x(t)$ is compared with all the model vectors $m_i(t)$. The best-matching node on the map, i.e., the node where the model vector is most similar to the input vector in some metric (e.g. Euclidean) is identified. This best matching unit is called the winner.

Step 2 - the model vectors of the winner and a number of its neighbours are changed toward the input vector according to the following learning principle:

$$m_i(t+1) = m_i(t) + \alpha(t)[x(t) - m_i(t)], \quad \forall i \in N_w(t)$$
$$m_i(t+1) = m_i(t) \quad \text{otherwise},$$

where the factor $\alpha(t) \in [0,1]$ is a scalar that defines the relative size of the learning step, and $N_w(t)$ specifies the neighborhood around the winner in the map array.

At the beginning of the learning process, the radius of the neighborhood is fairly large, but it is made to shrink

during learning. This ensures that the global order is obtained already at the beginning, whereas toward the end, as the radius gets smaller, the local corrections of the model vectors in the map will be more specific. The factor $\alpha(t)$ also decreases during learning. If the number of available input samples is restricted, the samples must be presented reiteratively to the SOM algorithm.

3.2. Self-Organising Ontology Map in MaLM

MaLM uses the SOM algorithm described above in the following way. We describe the algorithm from the perspective of the client device, when trying to understand replies; the same steps apply for the server device, when understanding incoming queries.

Bootstrap - The document set is incrementally built: from the service queries sent out by the device and the service descriptors obtained in answers (in the case of pervasive service discovery), and from the context of recommendation requests' and replies (in the case of distributed trust management). These descriptions (strings) are first broken into words; words are then used to build up two dictionaries: $\mathcal{D}_m$, of cardinality n_m, made up of words used in the requests, and $\mathcal{D}_f$, of cardinality n_f, made up of words used in the replies. In other words, $\mathcal{D}_m$ is the device mother-tongue and $\mathcal{D}_f$ is a collection of 'foreign' ontologies. Each document is then represented as a vector $x(t) \in \{0,1\}^n$, where n is the cardinality of $\mathcal{D} = \mathcal{D}_m \cup \mathcal{D}_f$ and $x[i](t) = 1$ if the i^{th} word in $\mathcal{D}$ is present in document t, and $x[i](t) = 0$ otherwise. In practice, the dictionary is subject to growth over time, while new queries/assertions are processed (i.e., the dictionary is not fully known aforehand), so its cardinality is actually chosen to be bigger than the actual number of different words counted so far.

Training - A SOM is created, with its size chosen one order of magnitude smaller than the size of the mother-tongue dictionary $\mathcal{D}_m$ (a universally good choice does not exist; in the following section, we provide details about our choice of values during experiments). Model vectors are initialised at random within $[0,1]^n$. Training starts with vectors whose words all belong to the mother-tongue (i.e., derived from device's queries). For each of them, the process described in the previous section is applied; the Euclidean distance is used as a metric to locate the winning model vector. At the beginning, both α and N_w are set to high values (α close to 1 and N_w close to the size of the whole map), and are then made to linearly decrease with t. As the sample set is usually not large, the process is repeated over the same input until the map has becomes 'good enough', that is, until the average quantisation error $E(||x - m_w(x)||)$ (where w is the best match for x) is made arbitrarily small.

Translating - Whenever a new service descriptor or recommendation is received, it is processed into a vector $x \in \{0,1\}^n$, with words being added to $\mathcal{D}_f$ (and thus $\mathcal{D}$) if necessary. The best matching node w is identified, and its associated vector m_w (and possibly its close neighbours) returned by SOM. The final translation simply requires to: (1) prune elements of m_w whose corresponding words do not belong to $\mathcal{D}_m$; (2) order the remaining elements by decreasing value of $m_w[i]$ (from most significant to least significant words) and return them.

While the above steps have been presented sequentially, they actually happen in cycle during the device lifetime (i.e., the device goes back to training when new queries/assertions are processed).

4. Evaluation

To evaluate the accuracy of the ontology mapping technique described in the previous section, we have conducted the following experiment. We have extracted information from the ontology used by Amazon Auction [1] and eBay [2]; while these systems are not representative of pervasive computing applications, their ontology could well be used by groups of pervasive devices, for example, in street market fairs to advertise products from specific categories, or to exchange recommendations about trustworthy sellers. While the products on sale by these systems are broadly speaking of the same type (antiques, books, clothes, etc.), the ontology used are rather different; in particular, Amazon Auction uses approximately 350 distinct words (dictionary D_A) to describe product categories, while eBay uses approximately 650 (dictionary D_E). D_A is not strictly contained in D_E as the cardinality of $D_A \cup D_E$ is approximately 800.

We have simulated a scenario where a set of devices uses D_A as mother tongue, while another set of devices uses D_E. The two groups exchange queries/assertions expressed in their own dialect, while learning on-the-fly the foreign one by means of the Self-Organising Map technique implemented by MaLM middleware (we have used an implementation of SOM called SOM_PAK [16]). We have collected 300 product descriptions (set V_A) expressed using D_A (e.g., "Exercise Sports Equipment") and 500 descriptions (set V_E) using D_E (e.g., "Inline Roller Skating Sporting Goods"). These descriptions have been broken into words and transformed into input vectors for MaLM as described in the previous section. From the perspective of

devices speaking D_A, V_A vectors are used for training the map, while V_E vectors are the ones that require translations; for devices speaking D_E, the opposite holds.

We have run a number of experiments, varying the following parameters: size of the map, duration of training, size of the learning step $\alpha(t)$, and size $N_w(t)$ of the neighborhood around the winner in the map. Our aim was to experimentally find the values of these parameters so that training (i.e., overhead) was minimal, while still bringing accurate translations. At bootstrap, a 10x10 map is created with values taken at random. The training is then split in two steps: an initial coarse-grained learning, repeated 30 times, where input vectors in the mother tongue are used to order the map (during this stage, we set $\alpha = 0.05$ and $N_w = 10$, that is, high learning rate and neighbourhood that encompasses the whole map), followed by another 300 steps of fine-tuning, where α starts at 0.02 and N_w at 3, with the former linearly decreasing to 0 and the latter to 1 during training. At this point, the map is ready and, when used to learn the foreign dialect, it gives an average quantisation error (computed over the whole foreign input set) of 1.8 when D_A learns D_E, and 1.65 when D_E learns D_A. The average error is slightly higher in the former case as Amazon Auction ontology is considerably smaller than eBay ontology, that is, it is more difficult to accurately find a mapping from an eBay descriptions to Amazon ones, as some words are completely unknown to D_A. Overall, error is very small and MaLM is capable of quite precisely locating the point in the map representing concepts which are semantically close to the processed input (the average quantisation error at bootstrap, before training starts, is around 10 in all simulated settings). We have run the simulations on a notebook with Intel Pentium processor 1400MHz CPU and 800MB RAM; simulation time was approximately 75" for training, and 60" to map the whole set of foreign descriptions to the device mother tongue (approximately 0.1" to translate each sample). While these numbers are not meaningful as the tested device is ways more powerful than a pervasive computing device, they demonstrate that the approach is indeed worth investigating, and our next step will be to run MaLM on really pervasive devices.

5. Related Work

Many research efforts have been devoted to solve the problem of 'protocol heterogeneity' in pervasive computing, leading to middleware-based solutions, such as SATIN [20], ReMMoC [13] and INDISS [6], that exploit a variety of components, reflection and event-based parsing techniques to achieve interoperability.

To date, the problem of ontology heterogeneity has received little attention instead. The few works that specifically tackle the issue can be classified in one of two categories: *static mapping* and *dynamic mapping*. Approaches belonging to the former category (e.g., [5]) make the assumption that the various ontologies are known a priori, so that static mappings between the various ontologies can be created off-line and injected into the device middleware; no dynamic learning occurs so that, if a device is encountered that speaks a language other than those already known, it is inevitably excluded from interactions. Off-line translations between the new language and each of the other languages must be created and injected into the middleware, inevitably leading to a scalability problem. Approaches belonging to the latter category (e.g., [14]) were mainly developed to tackle the problem of ontology heterogeneity for the Semantic Web. In this domain, performance overhead is not much of an issue; moreover, the degree of dynamicity experienced is by orders of magnitude lower than in pervasive computing. As a result, a variety of supervised and semi-supervised machine learning techniques (e.g., [11, 10]) have been developed and proved to be very useful. We have been inspired by these approaches; however, we argue that such techniques are of limited applicability in the pervasive domain, as they are rather heavyweight both in terms of performance overhead, in the amount of domain knowledge they require from the application developer, and in the amount of user intervention they expect.

In an attempt to find a solution to the ontology heterogeneity problem which could be successfully used in pervasive computing scenarios, we have explored an unsupervised machine learning technique, which enables *autonomic* learning and *on-the-fly* translations between a device 'mother-tongue' and other dialects, thus allowing higher degrees of flexibility and dynamicity.

6. Conclusion

In this paper we have presented MaLM, a machine learning middleware that tackles the problem of ontology heterogeneity. MaLM uses the Self-Organising Map unsupervised learning technique to autonomically map queries/assertions written in a foreign ontology onto the device mother-tongue. A proof-of-concept evaluation based on eBay and Amazon Auction ontology has demonstrated the accuracy of the process.

Our plans for the future spans three main directions. To assess the performance overhead imposed by MaLM, we intend to run it on really pervasive devices, in particular, mobile phones and PDAs; while these devices do not cover the full spectrum of pervasive devices (leaving out the most resource constrained ones), we expect them to be the most widely used ones in the scenarios we have described in the paper. To measure the trade-off between accuracy and usability, we intend to investigate and experiment with supervised and semi-supervised learning techniques, thus being

able to better assess advantages and disadvantages of such techniques in the pervasive computing scenario. Finally, we plan to integrate MaLM with both a service discovery [8] and a distributed trust management [7] platform for pervasive devices we have previously developed, and run experiments in the context of a real market fair, as part of the Utiforo [3] EPSRC project; this will enable proper evaluation in a real life setting where training happens incrementally.

References

[1] Amazon Auctions. http://auctions.amazon.com/, 2006.

[2] eBay. http://www.eBay.com/, 2006.

[3] Utiforo - Pervasive Computing Support for Market Trading. http://www.informatics.sussex.ac.uk/projects/markets/, 2006–2009.

[4] K. Arnold, B. O'Sullivan, R. W. Scheifler, J. Waldo, and A. Wollrath. *The Jini[tm] Specification*. Addison-Wesley, 1999.

[5] Y.-D. Bromberg and V. Issarny. Service Discovery Protocols Interoperability in the Mobile Environment. In *Proc. of the International Workshop Software Engineering and Middleware (SEM)*, volume 3437 of *LNCS*, pages 64–77, Linz, Austria, Sept. 2004.

[6] Y.-D. Bromberg and V. Issarny. INDISS: Interoperable Discovery System for Networked Services. In *Proc. of the* 6^{th} *International Middleware Conference*, Grenoble, France, Nov. 2005. To appear.

[7] L. Capra. Engineering Human Trust in Mobile System Collaborations. In *Proc. of the* 12^{th} *International Symposium on the Foundations of Software Engineering (SIGSOFT 2004/FSE-12)*, pages 107–116, Newport Beach, CA, USA, Nov. 2004. ACM Press.

[8] L. Capra, S. Zachariadis, and C. Mascolo. Q-cad: Qos and context aware discovery frameworkfor adaptive mobile systems. In *International Conference on Pervasive Services (ICPS'05)*, Santorini, Greece, July 2005. IEEE.

[9] G. Cugola, E. D. Nitto, and A. Fuggetta. Exploiting an event-based infrastructure to develop complex distributed systems. In *19th International Conference on Software Engineering (ICSE98)*, 1998.

[10] A. Doan, P. Domingos, and A. Halevy. Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach. In *ACM SIGMOD International Conference on Management of Data*, Santa Barbara, CA, USA, 2001.

[11] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. *Handbook on Ontologies in Information Systems*, chapter Ontology Matching: A Machine Learning Approach. Springer-Velag, 2003.

[12] T. Edgington, B. Choi, K. Henson, T. Raghu, and A. Vinze. Adopting ontology to facilitate knowledge sharing. *Communications of the ACM*, 47(11):85–90, Nov. 2004.

[13] P. Grace, G. Blair, and S. Samuel. Middleware Awareness in Mobile Computing. In *Proc. of the* 1^{st} *International ICDCS Workshop on Mobile Computing Middleware*, page 382, Providence, Rhode Island USA, May 2003.

[14] Y. Kalfoglou and M. Schorlemmer. Ontology Mapping: The State of the Art. *The Knowledge Engineering Review Journal*, 18, 2003.

[15] T. Kohonen. *Self-Organising Maps*. Springer-Verlag, 1995.

[16] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen. SOM_PAK - The Self-Organizing Map Program Package - Version 3.1, 1995.

[17] G. Picco, A. Murphy, and G.-C. Roman. LIME: Linda meets Mobility. In *Proc.* 21^{st} *Int. Conf. on Software Engineering (ICSE-99)*, pages 368–377. ACM Press, May 1999.

[18] UPnP Forum. Universal Plug and Play. http://www.upnp.org/, 1998.

[19] W3C. Web ontology language (owl).

[20] S. Zachariadis, C. Mascolo, and W. Emmerich. Satin: A component model for mobile self-organisation. In *Proceedings of Int. Symposium on Distributed Objects and Applications (DOA)*, number 3291 in Lecture Notes in Computer Science, pages 1303–1321, October 2004.