Collaborative Learning of Ontology Fragments by Co-operating Agents

Heather S. Packer, Nicholas Gibbins and Nicholas R. Jennings
Intelligence, Agents, Multimedia Group,
School of Electronics and Computer Science,
Southampton, UK
{*hp07r,nmg,nrj*}*@ecs.soton.ac.uk*

***Abstract*—Collaborating agents require either prior agreement on the shared vocabularies that they use for communication, or some means of translating between their private ontologies. Thus, techniques that enable agents to build shared vocabularies allow them to share and learn new concepts, and are therefore beneficial when these concepts are required on multiple occasions. However, if this is not carried out in an effective manner then the performance of an agent may be adversely affected by the time required to infer over large augmented ontologies, so causing problems in time-critical scenarios such as search and rescue. In this paper, we present a new technique that enables agents to augment their ontology with carefully selected concepts into their ontology. We contextualise this generic approach in the domain of RoboCup Rescue. Specifically, we show, through empirical evaluation, that our approach saves more civilians, reduces the percentage of the city burnt, and spends the least amount of time accessing its ontology compared with other state of the art benchmark approaches.**

***Keywords*-agent learning; ontology; RoboCup Rescue;**

I. Introduction

Collaborating agents require a shared vocabulary in order to communicate, or in the event that no prior shared vocabulary exists they require a technique for building one. Using a vocabulary, in our case an ontology, that encompasses all concepts and their relationships is resource intensive in terms of hosting, managing and using. Therefore, it is desirable to build a relatively small ontology on-line where an agent can learn about the specific resources already described in its environment. The simplest approach for an agent is to exchange the definition of a single concept when required, however this does not take into account the overhead costs of acquiring new information, such as the composing and sending of messages. We propose a learning approach that automatically augments an OWL ontology with selected information incrementally with consideration to acquiring new information. This is especially important in time-critical scenarios where building a vocabulary should enable rather than hinder prompt decision making.

We situate our approach in the RoboCup Rescue (RCR) simulation [4], a standard search and rescue environment for the development of strategies to enable agents to allocate tasks, co-ordinate agents and plan their paths to rescue allocated targets. Our extension, RoboCup OWLRescue (RCOR), uses ontologies that describe available resources and which are used to inform the agent's decision making process. We consider the costs incurred by the acquisition of knowledge (e.g. composing and sending messages) and the evolution of large ontologies (e.g. hosting, managing and using). Thus, we aim to augment the ontologies used by the RCR agents in an as-needed fashion so that they can perform new tasks and to reduce the cost of acquiring (or reacquiring) concepts from other agents.

In summary, we contribute to the state of the art of ontology evolution with:

1) The first algorithm that selectively allows agent to augment their ontologies on-line with a fragment representing a desired concept, by learning domain-related concepts from other agents in their environment. In related work (see Section II) agents augment their ontologies with one concept at a time and are not selective with the concepts that they learn.
2) A technique that outperforms the state of the art in terms of balancing the trade-off between learning concepts through collaboration and reducing the cost of collaboration. Our technique selectively learns concepts which reduces the reacquisition of knowledge, and the overhead acquisition message cost.

Our results show that our method when compared to other existing approaches (as described in Section II), reduces the time required to deliberate and access knowledge, thus enabling it more time to save targets compared with the current state of the art. Our approach saved more of the city from fire (increasing the average from 88.3% to 92.8%) and saved more civilians by 20.8% compared to other approaches with the exception of an approach that had perfect foresight (see Section VII).

It should be noted that our learning approach does not guarantee that reasoning from an augmented ontology is sound or complete. However, this is not always a requirement, and a 'good enough' answer is appropriate in many cases where a response is required quickly. For example, an agent that rescues casualties requires that it can respond before the patient's condition deteriorates, thus the optimal answer (where the entire knowledge base is consulted) may prevent the casualties being rescued in time, while the 'good enough' answer provides a quick enough response so that

casualties can be rescued. There is a trade-off between these two cases, where the 'good enough' response does not enable the rescue of the casualties, and the optimal answer does not provide a timely response. Our approach enables the agent to respond by selectively augmenting its ontology with domain specific information, and is suited for light-weight ontologies that support services.

We now proceed with related work in Section II. Sections III and IV introduce RCOR and our learning technique. Then we describe in more detail our concept selection process in Section V. Sections VI and VII detail our evaluation and results. Finally, we conclude in Section VIII.

II. Related Work

The approaches presented by Bailin and Truszkowski [5], Afsharchi et al. [6], and Soh [7] enable agents to augment their ontologies with one new concept at a time. In particular, Bailin and Truszkowski's approach considers semantically equivalent representations by using WordNet [8] to translate concepts to aid communication between agents. Afsharchi et al. and Soh focus on the validation of the knowledge to be incorporated into the agent's ontology. In order to validate the knowledge, Afsharchi et al. use positive and negative examples to train agents' new vocabulary. These approaches aim to augment an ontology with one concept at a time, whereas our approach considers the overhead cost of acquiring new concepts and thus augments more than one at a time.

In addition to agent-based research, the Semantic Web community has produced work on evolving and merging ontologies [9] and [10]. Research that focuses on merging two ontologies focuses on handling inconsistencies so that the ontology can give the same answers as it could before. In particular, the techniques presented by Flouris et al. [11] enable the evaluation of coherence and consistency of an evolving ontology by defining a set of postulates that evaluate a revision of an ontology. The work presented by Hasse and Stojanovic [12] further explores the issue of consistency, by proposing techniques to resolve three types of inconsistency; structural, logical, and user defined inconsistencies. These techniques can be used with our approach in order to evaluate, locate and resolve inconsistent knowledge to be incorporated into an ontology. In contrast to these approaches, which focus on guaranteeing soundness and completeness, our approach focuses on providing prompt answers which are used for time critical scenarios.

The Semantic Web community has also produced work on extracting modules, a set of axioms that represent a concept, from an ontology. Automated techniques such as [13] and [14] focus on separating a single ontology into modules. In contrast to our technique, these techniques focus on creating a module from one ontology, whereas our technique selects a fragment from an ontology based on the concepts already contained in another ontology. We note that our scenario can use modularisation to provide agents with fragments of ontologies, so that they can select which concepts they want to augment.

Also, Maedche and Staab [15], provide a method to compare the similarity of two ontologies. This method consists of comparing concept names lexically, and conceptual comparison using structural analysis. In contrast, our approach implements a selection process so that the agent can augment its vocabulary with a small amount of axioms that relate to its ontology, where as Maedche and Staab's approach aims to measure the similarity of two ontologies.

In addition to considering agent learning techniques, we build upon RCR which models the effects of an earthquake on a virtual city's buildings, civilians, and roads. At the beginning of a simulation, buildings may have: collapsed, possibly with civilians buried inside; caused road blockages; and, ignited. There are three types of RCR agents with specific capabilities: ambulance teams recover buried civilians, and transfer them to refuges; fire teams extinguish fires, and police force teams clear blocked roads. The challenge for a RCR team is to save the lives of as many civilians as possible, and to minimise the area of the city which is burnt. The performance of a team is evaluated using a formula which factors in the percentage of live civilians, the state of live civilians, and the average building damage. While this scenario provides a testbed for developing the co-ordination of agents, our extension aims to extend the variables associated with each target (civilians, buildings, and blockages) resulting in a set of possible actions an agent can take. Each action affects the outcome of the scenario. To this end, we follow by describing our extension in the following section.

III. RoboCup OWLRescue Framework

The RCOR framework extends buildings to contain (possibly hazardous) chemicals, and extends civilians to have symptoms. The RCOR agents require different knowledge for each simulation because the variables are dynamic. All agents have their own ontologies so that an agent can augment its ontology with information about its tasks from ontologies in the environment. These *environment ontologies* describe the available resources which can be used in the agents' decision making processes, and describe vehicles and their ability to deal with fires, building collapses and casualties. The RCOR agents access the environment ontologies by requesting information about concepts and receive fragments representing a desired concept. The agent can then augment its ontology with all the concepts or a selection of concepts depending on the agent's strategy. Each command centre is assigned a set of vehicles which it can allocate on a first-come, first-served basis to agents. Each agent is allocated a vehicle; if its vehicle does not have the necessary equipment for a task, it can then exchange it at a command centre.

The RCOR agents can learn about variables encountered while rescuing a target and alternative resources. For ex-

ample, a police rescue agent can discover an appropriate construction vehicle which can remove a blockage from a road. It is beneficial for agents to augment their ontology so that they can successfully perform tasks that they could not complete before. In the RCR, a team of agents must complete these tasks within five seconds which represents one timestep in the simulation. Specifically, a timestep is the amount of time that each agent has to decide on its next action before the targets in the world are updated either with new targets or changes to existing targets. Thus an agent must spend its time efficiently performing actions. In order to do this, our approach aims for agents to maintain a small ontology and send a minimal number of requests for information from the environment ontologies. In the following section we describe our learning approach which enables agents to augment their ontology with information that enables agents to save targets.

IV. Our Learning Approach

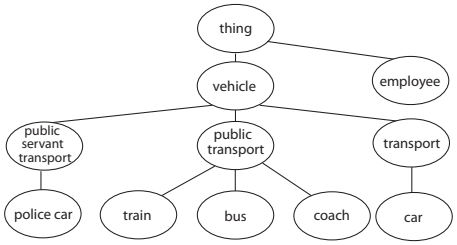
A rescue agent aims to save a target and requires specific knowledge to do so. This knowledge is contained in a private ontology, which is composed of two distinct ontologies: a Domain Ontology (DO) and an Evolving Ontology (EO). The DO contains the axioms with which the agent is instantiated and does not change. The EO contains acquired axioms and allows the agent to augment its knowledge base with concepts, without affecting its core expertise. The DO imports the EO, so that it can make logical entailments from both the DO and EO.

In order to explain our approach, we use the following example: a rescue agent a_q has a private ontology o_q that contains the concepts shown in Figure 1(a). It desires to learn the concept *firefighting motorcycle* as used in Japan, because it currently does not use fire fighting motorcycles but would like to be able to rush to the scene of a fire disaster, without concern for such problems as traffic jams and narrow residential roads. Thus, it has received two fragments from the environment ontologies, which represent *firefighting motorcycle*, as shown in Figures 1(b) and 1(c). We proceed to describe our merging process.

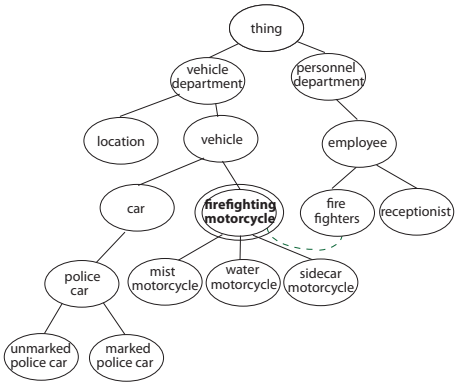
An agent a_q receives a set of fragments F from the environment ontologies, which represent the queried concept c. The agent a_q merges the fragments contained in F to form one merged fragment f_m. As the fragments are loaded, they are consistency checked against the other ontologies to ensure that the resulting merged fragment is consistent. In the next section we describe our learning approach for selecting the concepts to augment into an agent's ontology.

V. Concept Selection

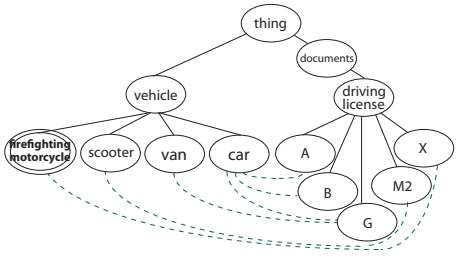
Our concept selection approach selects concepts to augment an agent's ontology. We adopt this approach because we want to balance the trade-off between learning everything which can be costly in terms of accessing the knowledge



(a) An example rescue agent's ontology



(b) First fragment, f_1, which represents the concept *firefighting motorcycle*.



(c) Second fragment, f_2, which represents the concept *firefighting motorcycle*.

Figure 1. The rescue agent's ontology, and three fragments, received from specialist agents in response to a query of *firefighting motorcycle*. The dashed lines represent relationships between two concepts.

from the ontology, against sending requests for fragments which describe concepts required to complete a task which can also incur costs of time and resources. In order to balance this trade-off, we select concepts closely related to the domain of the agent's EO by first selecting a set of concepts with a similar depth, through **Hierarchical Selection**, followed by reducing the number of relations to the agent's ontology, through **Relational Selection**. Both of these techniques are described in the following two sections.

A. The Hierarchical Selection Technique

The hierarchical selection technique returns a list of concepts C^h. The hierarchical selection technique aims to reduce the number of superclasses that are used to represent the target concept c, when f_m has a larger hierarchical depth than o_q, to reduce the amount of non-domain-related information the agent learns. The hierarchical selection

technique rates concept in f_m a *concept rating* using the following equation, this rates concepts according how they relate to the agent's domain of interest:

$$concept\ rating_c = w * (nDO + nDOR) + nEO + nEOR \quad (1)$$

where nDO and nEO are the number of axioms which refer to the concept c in the DO and EO, respectively, and $nDOR$ and $nEOR$ are the number of axioms which contain a relationship between the concept c in f_m and a concept in the agent's DO and EO, respectively. A weighting w is used to increase the rating of concepts that refer to items in the DO, which are domain related. Currently, we use the weighting $w = 2$ where common concepts in the DO are twice as important as common concepts in the EO. This weighting can be adjusted to specific requirements: the lower the weighting, the more likely an agent can evolve its ontology to represent a different domain to its intended domain; the higher the weighting is the less likely it is to deviate from its intended domain.

The hierarchical selection then performs one of two different actions, depending on the depth of the fragment, specifically, the following cases:

1) $depth(concepts(f_m)) > depth(concepts(o_q))$. In this case, the agent reduces the depth of the fragment. The agents calculates the average depth d of f_m and o_q, in our example 4, and selects d levels. It does this by selecting $d - 1$ levels with the highest mean average concept value shown in Figure 2. In our example, the levels selected are: level 3, because it contains c (the level containing c is always selected first); followed by level 2, level 5 and level 4, based on the order of the level's average concept rating.

2) $depth(concepts(f_m)) \leq depth(concepts(o_q))$. In this case, the agent selects all of the levels in f_m.

Both of these cases create a set of concepts, C^h, that represent the target concept, c. After the hierarchical selection process, the resulting set of concepts is then used by the relational selection technique to select the concepts to augment into the rescue agent's ontology. This selection process is described in Algorithm 1.

B. The Relational Selection Technique

The relational selection technique returns a set of concepts C^r that will be incorporated into an agent's EO. The first stage of our relational selection technique selects concepts by traversing relationships, such as the subsumption relationship *subClassOf*, and object properties, for example *requiresLicense*, from our target concept. The distance of traversal is determined by a threshold t. To calculate t, we determine if an agent has an ontology that contains no properties, in which case $t = 1$, otherwise $t = round(\frac{\sum r_r}{r_c})$, where r_r is the number of concepts that have more than one relationship, and r_c is the number of concepts that are linked together by relationships. The second stage of

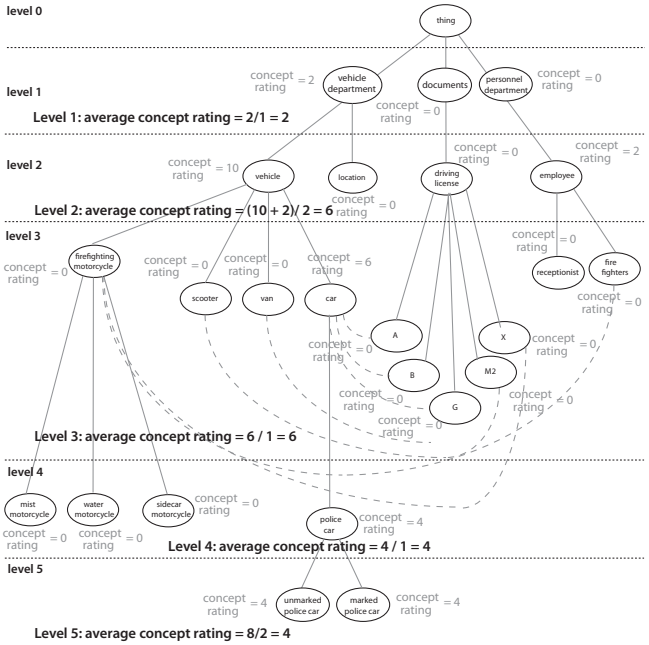


Figure 2. The merged fragment with average concept ratings per level.

Algorithm 1 Hierarchical Selection technique.

Function: $depth(x)$ returns the depth of an object x that contains classes
Function: $rating(c)$ returns a concepts c's *concept rating*
Function: $depthLevel(c)$ returns a concepts c's depth level from the merged fragment.
Input: $F_d \leftarrow$ fragment received from merging process
Input: $o_i \leftarrow$ agent's ontology
Input: $I \leftarrow$ number of depths of classes to incorporate
Input: $c \leftarrow$ concept to incorporate
Require: $Fragment \leftarrow \emptyset$
1: $N \leftarrow depth(F_d)$
2: **if** $N > depth(o_i)$ **then**
3: **for each** $Class \in F_d$ **do**
4: $Level \leftarrow depthLevel(Class)$
5: $R \leftarrow rating(Class)$
6: $Ratings_{Level} \leftarrow Ratings_{Level} \cup \{R\}$
7: $Classes_{Level} \leftarrow Classes_{Level} \cup \{Class\}$
8: **end for**
9: **for each** $N \in \{Ratings_{1...N}\}$ **do**
10: $AvgRating_N \leftarrow average(Ratings_N)$ 1
11: **end for**
12: **for each** $N \in \{Ratings_{1...N}\}$ **do**
13: `/* order sets by average rating, descending`
14: `* subsort sets by number of classes in each depth level, descending*/`
15: $Ratings_N = sort(AvgRating_N, |Classes_N|)$
16: **end for**
17: `/* select highest` I `number of sets */`
18: **for each** select highest I in $Class_I \in Classes_{Level}$ where $c \in Class$ **do**
19: `/* incorporate each class and their properties in the highest` I `into a fragment */`
20: $Fragment \leftarrow Fragment \cup \{Class_I\}$
21: **end for**
22: **else**
23: **for each** select $Class \in F_d$ **do**
24: **if** $rating(Class) > 0$ **then**
25: $Fragment \leftarrow Fragment \cup \{Class\}$
26: **end if**
27: **end for**
28: **end if**
29: **return** $Fragment$

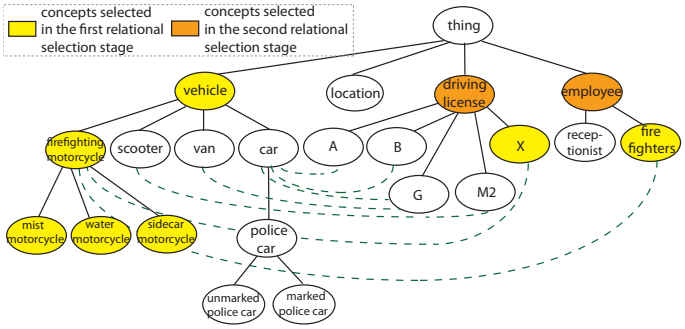


Figure 3. A selection of concepts which are related to c by subsumption, where $c = fire\_engine$.

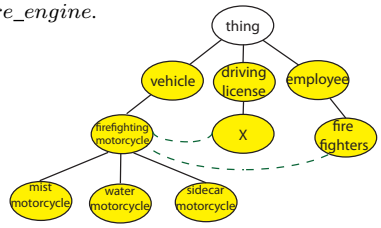


Figure 4. The chosen set of concepts which are augmented into a_q's EO.

our relational selection technique recursively selects the parents of the concepts selected in the first stage, up to the root node. In our example, $t = 1$, and eight concepts are related at this distance from the target concept *firefighting motorcycle*. This process is described in Algorithm 2.

Algorithm 2 Relational Selection technique, where c is the concept to incorporate, t is the distance threshold of axioms to incorporate and C^r is a set of concepts to be returned.

Function: $path(c)$ returns a set of concepts from the root node to the parameter concept c to the leaf nodes, from C^h
Function: $relations(c)$ returns a set of concepts that are relate by relationships to c_t where the relationships are defined in the merged fragment and the concepts are from C^h.
Input: C^h
Input: c_t
Require: $C^r \leftarrow \emptyset$
Require: $related \leftarrow \emptyset$
1: $C^r \leftarrow C^r \cup path(c_t)$
2: **for** $n = 1 \ldots t$ **do**
3: $tmp = \{c_t\}$
4: **for each** $class \in tmp$ **do**
5: $related \leftarrow related \cup relations(class)$
6: `/* select relationships */`
7: **for each** $c \in related$ **do**
8: $C^r \leftarrow C^r \cup path(c)$
9: `/* traverse related classes */`
10: $tmp \leftarrow tmp \cup c$
11: **end for**
12: **end for**
13: **end for**
14: **return** C^r

Figure 3 depicts the concepts selected in the first stage in yellow, and the concepts selected in the second stage in orange. This selection results in a set of concepts to be augmented into the agent's ontology. In our example, the selected concepts (shown in Figure 4) are augmented into the agents ontology which result in the ontology represented in Figure 5.

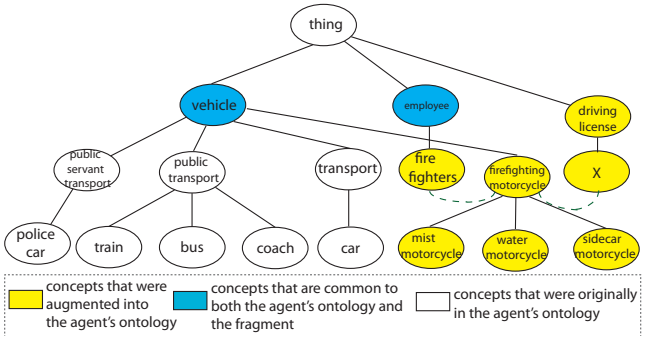


Figure 5. Our example ontology augmented with the selected concepts

Algorithm 3 Pseudocode of the RoboCup simulator

1: RCR Simulator generates fires, blockades and civilians
2: `/* timesteps are limited to 5 seconds per all agents, to model time critical scenario (see RCR simulation) */`
3: **for** each timestep **do**
4: Agent receives a list of targets from the simulator
5: Agent selects a target to rescue, and get the information about the target. For example, the target is a building and it contains the chemicals { `helium`, `kerosene` }
6: **if** Agent Ontology does not contain information about target **then**
7: Request fragments from environment ontologies about unknown concepts
8: Select a set of axioms to learn
9: **end if**
10: **if** Agent's vehicle does not have equipment required to rescue target, based on agent ontology **then**
11: Travel to centre agent to get equipment
12: **end if**
13: Agent rescues target
14: Simulator updates environment based on agent actions
15: **end for**

VI. Evaluation

In order to evaluate our approach, *learn-fragment*, we compare the effectiveness of agents using different learning techniques (discussed below) to: rescue civilians; put out burning buildings; to evaluate the requirements of rescuing a target against its RoboCup score (described in Section II); and investigate the ratio of time taken to deliberate, communicate, and act. Similar to the RCR Competition, our simulation allows a team of agents five seconds to complete an action in a timestep, and are given three hundred timesteps to save as many of the civilians and burning buildings as possible. The pseudo-code in Algorithm 3 provides the basic scenario of the agents in the RCOR.

In our experiments, we initialise a RCOR scenario where the ambulance, fire brigade and police agents use a learning technique when they encounter unknown concepts or do not have the right equipment to rescue their target. In more detail, the comparison learning approaches are:

1) *Learn-repeated* approach: learns all concepts and their relationships which are required more than once. This aims to offset the cost of learning a concept compared with its use. It demonstrates how an agent would learn if it had perfect foresight, and is used in this evaluation as in [16].

2) *Learn-concept* approach: learns all axioms that are directly connected to the concept being queried. This is a comparable technique with the agent approaches of [5], [6], and [7], which learn a single concept at a time (see Section II).
3) *Learn-everything* approach: learns all axioms in all of the fragments it locates. This technique is designed to show an agent's potential ontology complexity and is a comparable technique with the agent approach of [17].
4) *No-collaboration* approach: this is a control approach that does not collaborate with the environment ontologies and highlights the need for learning more about the environment.

These agents adopt their behaviour defined by the sample package in RCR, which determines the agents' behaviour such as planning a path through the virtual city and which target to rescue first. We note this adopted behaviour is basic, whereby there are no algorithms used to co-ordinate agents' targets or to minimise path traversal. Our investigation consists of comparing how five different learning techniques perform given the same set of 200 scenarios. Each scenario is randomly generated by the RCR simulators. For our evaluation our framework includes the following environment ontologies:

1) **EAC Ontology**: This ontology describes the Emergency Action Code (EAC), which is a three character code displayed on all dangerous goods classed carriers. This ontology provides fire agents with information about the required equipment for attending burning targets, and is derived from the National Chemical Emergency Centre (NCEC) code list.
2) **HazChem Ontology**: This Hazardous Chemical (HazChem) ontology classifies chemicals using Hazardous Identification (ID) Numbers (HIN). Similar to the EAC Ontology, this ontology provides fire agents with information about the required equipment for attending burning targets, and is derived from the The National Institute for Occupational Safety and Health (NIOSH) HIN system.
3) **Chemical Ontology**: This ontology contains chemicals and their EAC and HIN classification. This ontology allows agents to use either standard provided by the EAC and HIN, and enables the agent to translate chemicals between both standards.
4) **Vehicle Ontology**: This ontology describes vehicles, their attributes, purpose, and manufacturer. In particular, this ontology provides information about the track type of a vehicle, and its capabilities, and is derived from vehicle categorisations from the Driver and Vehicle Licensing Agency (DVLA).
5) **HantsFireEngineFleet Ontology**: This ontology contains information about the fleet of fire engines in the county of Hampshire (UK). This information is derived from the Hampshire fire service website[1], which details vehicle types, their model, manufacturer, and registration numbers.
6) **Ambulance Ontology**: This ontology contains information about different types of ambulance, their attributes, and equipment and is derived from the standards of the Ontario Ministry of Health and Long-Term Care[2].
7) **ConstructionVehicles Ontology**: This ontology contains information about construction vehicles and their capacity, and is derived from information from the book "Fundamentals of Technical Rescue."[3]
8) **Triage Ontology**: This ontology describes the 5-Category Triage System and identifies symptoms for each category, and is derived from the Australian Ministry of Health guidelines[4].
9) **CSI Ontology**: This ontology contains information from the Chemical Sampling Information (CSI) of the US Department of Labor Occupational Safety and Health Administration. The CSI contains details about chemicals and their health effects on humans, and the organs affected.
10) **Treatment Ontology**: This ontology contains information about burns and broken bones, their symptoms, and their treatment. This information has been taken from the NHS website[5].

Table I
THE NUMBER OF CONCEPTS IN EACH OF THE ENVIRONMENT ONTOLOGIES.

Ontology	No. of Concepts
EAC Ontology	1906
Chemical Ontology	1800
HantsFireEngineFleet Ontology	745
ConstructionVehicles Ontology	114
CSI Ontology	3841
EAC Ontology	1906
Chemical Ontology	1800
HantsFireEngineFleet Ontology	745
ConstructionVehicles Ontology	114
CSI Ontology	3841

These ten ontologies have been chosen because they are representative of standard industry vocabularies for the domains of interest of RCR agents. This combination of ontologies covers the areas required by the RCOR extension and represent a realistic set of information that rescue agents would need to consult in real conditions. The number of concepts in each of the ontologies is given in Table I. The next section presents the results and our analysis of our evaluation.

[1]Hampshire Fire and Rescue Service: http://www.hantsfire.gov.uk/theservice/sp-and-sr/fleetmanagement

[2]Ontario Ambulance Standards: http://www.health.gov.on.ca/english/providers/pub/ambul/equipment/standard.pdf

[3]Fundamentals of Technical Rescue, International Association of Fire Chiefs: http://books.google.com/books?id=mLyYsT8YEWkC&pg=PT33

[4]Ministry of Health Triage Guidelines: http://www.moh.govt.nz/moh.nsf/indexmh/ed-about-triage

[5]NHS Health Information: http://www.nhs.uk/chq/pages/Category.aspx?CategoryID=72

VII. Results

Agents using the *learn-fragment* approach outperformed the other approaches (with the exception of *learn-repeated* which has perfect foresight), by having the highest average number of civilians alive and percentage of the city that is unburned (see Table II) at the end of a simulation. The statistics in this table show that our approach on average saves 20.8% more civilians than the next best approach (*learn-fragment* and *learn-concept* save an average of 51.7% and 42.8% of the civilians, respectively). Also, our approach saves an average of 5.0% more of the city from fires, where our approach and the next best approach, *learn-concept*, saves 92.8% and 88.3%, respectively (shown in Figure II).

Table II
Percentage of living civilians and unburned buildings after 200 runs.

Approach	civilians alive	unburned buildings
learn repeated	52.1	93.7
no collaboration	29.1	83.3
learn everything	42.0	88.2
learn concept	42.8	88.3
learn fragment	51.7	92.8

Our approach balances the trade off between deliberating which equipment to use to rescue targets, and moving to rescue targets. Spending too much time on either deliberating or acting, results in either not enough time allocated to rescuing targets, or the degradation of efficiency of the actions taken. Both of these cases result in lower scores compared with an approach that balances these two actions (see Table II, and Figures 6 and 7). To see this, the amount of time spent on each task is represented in the pixel plots in Figure 8, where each row shows an agent's actions for one simulation and there are 200 agents on each figure. This figure shows that our approach balances the deliberating, and acting (indicated in black and white, respectively), while the other approaches spend a disproportionate amount of time on one single type of task. This is because the other approaches either spend the majority of their time acquiring and loading information (such as the *learn-concept*, *learn-repeated*, and *learn-everything* approaches), or acting without using the optimal equipment to rescue targets (such as the *no-collaboration* approach). We now proceed to discuss each learning approach in more depth:

The *learn-everything* approach augments all concepts received from the environment ontologies into the agent's private ontology. This enables the agent to communicate with the environment ontologies the fewest number of times compared with the other approaches (as shown in Figure 8(d), where this approach has the greatest number of dark pixels). However, in contrast to the other approaches, its performance is hindered by the time taken to access the concepts in the agent's ontologies (as shown in Figure 8(d)). Thus, these agents did not save as many targets compared with the other approaches (as shown in Table II and Figures 6 and 7).

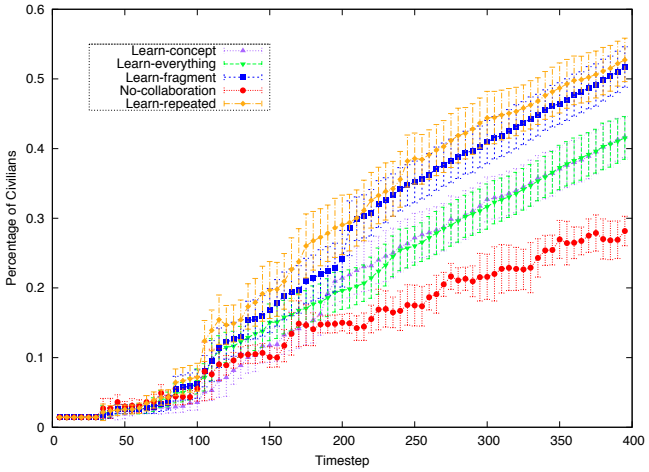


Figure 6. Percentage of civilians in refuge over 200 simulations.

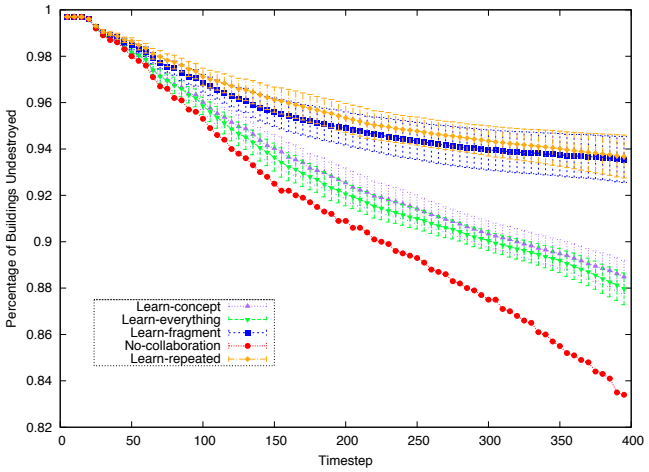
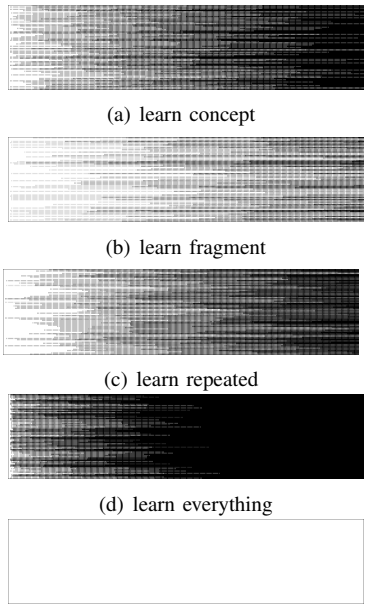


Figure 7. Percentage of buildings that are destroyed over 200 simulations.

The *learn-fragment* approach enables the agents to learn more than one concept at a time, therefore this approach enabled the agent to communicate with the environment ontologies less than the *learn-concept* approach (as shown on Figure 8(b), which has more light coloured pixels than Figure 8(a)), but more than the *no-collaboration*, *learn-everything*, and *learn-repeated* approaches. This approach enables agents to spend less time deliberating their actions than the other approaches, except for the *no-collaboration* approach (as shown on Figure 8(b), which has more light coloured pixels than Figure 8(a)). Thus, saving more targets (as shown in Table II and Figures 6 and 7).

The *learn-repeated* approach enables agents to only learn concepts that are used more than once in a simulation. This optimises the time spent augmenting its ontology and time accessing knowledge for rescuing targets because the agent's ontology only holds information that is useful for the scenario. This approach requires perfect foresight so that it knows which concepts will be required more than once in a scenario, this is unrealistic in rescue scenarios.

The *learn-concept* approach enables the agents to learn



(a) learn concept

(b) learn fragment

(c) learn repeated

(d) learn everything

(e) no collaboration

Figure 8. Pixel plots representing the time spent deliberating actions and communicating with the environment ontologies indicated by black, and actions which are indicated in white.

one concept at a time from the environment ontologies. Thus, agents using this approach spend longer deliberating their actions compared to agents using the *learn-fragment* and *no-collaboration* approaches (shown in Figure 8).

The *no-collaboration* approach was able to attend targets faster than other approaches because it spent no time deliberating its actions (as shown in Figure 8(e), which has the largest number of light pixels than other approaches). Therefore it can save its targets faster, however the civilians have on average lower health because they cannot select equipment that is optimal to save them (shown in Table II). This ability to select equipment affects the area of the city damaged by fires and is the worst performing approach in this aspect (shown in Figure 7).

In summary, our approach, *learn-fragment*, enables agents to balance the trade off between the time spent decided on which actions to take, and the time spent moving and saving targets. This balance is enabled by allowing the agent to augment a fragment of knowledge to its ontology, thus benefiting from reducing the number of communications with the environment ontologies than the *learn-concept* approach (as shown on Figure 8(b), which has more light coloured pixels than Figure 8(a)) and maintaining an ontology 12% of the size of the *learn-everything* approach's ontology.

VIII. Conclusions

We have presented and evaluated a technique that allows agents to learn concepts from other ontologies in their environment. Our approach focuses on learning concepts related to an agent's domain, enabling a higher cohesion between concepts in an agent's ontology, than comparative approaches presented in Section II. We show in our evaluation that our approach: reduces the time required to deliberate and access knowledge during a simulation, thus enabling it more time to save targets compared with the other approaches (see Section VII). On average our approach saved more of the city and a higher number of civilians compared with the other approaches, with exception to the *learn-repeated* approach which had perfect foresight (see Section VII).

We plan to enable our approach to predict future queries so that it can improve its performance. In order to predict future queries, we plan to enable our agents to construct their own probabilistic model based on the queries that an agent receives so that it can adapt to changing requirements.

References

[1] J. Malin. Preparing for the unexpected: Making remote autonomous agents capable of interdependent teamwork. In *Human Factors and Ergonomics Society Annual Meeting Proceedings*, (44), pages 254–257, 2000.

[2] N. Schurr, J. Marecki, J. Lewis, M. Tambe, and P. Scerri. The defacto system: Coordinating human-agent teams for the future of disaster response. *Multiagent Systems Artificial Societies and Simulated Organizations*, 15:197, 2005.

[3] R. Washington, K. Golden, J. Bresina, D. Smith, C. Anderson, and T. Smith. Autonomous rovers for Mars exploration. In *IEEE Aerospace Conf.*, 1999.

[4] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada. RoboCup Rescue: search and rescue in large-scale disasters as a domain for autonomous agents research. In *IEEE Int. Conf. on Systems, Man and Cybernetics, 1999.*

[5] Bailin, S., Truszkowski, W.: Ontology Negotiation between Intelligent Information Agents. The Knowledge Engineering Review **17**(01) (2002) 7–19

[6] Afsharchi, M., Far, B., Denzinger, J.: Ontology-Guided Learning to Improve Communication between Groups of Agents. Int. Joint Conference on Autonomous Agents and Multiagent Systems, Hakodate, Japan (2006) 923–930

[7] Soh, L.: Multiagent, Distributed Ontology Learning. Working Notes of the Second AAMAS OAS Workshop, Bologna, Italy, 2002

[8] Miller, G.: WordNet: A Lexical Database for English. CACM **38**(11) (1995) 39

[9] McGuinness, D., Fikes, R., Rice, J., Wilder, S.: An Environment for Merging and Testing Large Ontologies. Int. Conf. on Principles of Knowledge Representation and Reasoning, Breckenridge, Colorado, USA (2000) 483–493

[10] Stumme, G. and Maedche, A. FCA-Merge: Bottom-up merging of ontologies International Joint Conference on Artificial Intelligence 17:1 225–234 (2001).

[11] Flouris, G., Huang, Z., Pan, J., Plexousakis, D., Wache, H.: Inconsistencies, Negations and Changes in Ontologies. In the Proceedings of the Nat. Conf. on Artificial Intelligence, Boston, MA, USA **21** (2006) 1295

[12] Haase, P., Stojanovic, L.: Consistent Evolution of OWL Ontologies. European Semantic Web Conference, Heraklion, Greece (2005) 182–197

[13] Kontchakov, R. and Pulina, L. and Sattler, U. and Schneider, T. and Selmer, P. and Wolter, F. and Zakharyaschev, M. Minimal module extraction from DL-Lite ontologies using QBF solvers. Int. Joint Conf. on AI, 836–840, 2009

[14] Grau, B.C. and Horrocks, I. and Kazakov, Y. and Sattler, U. Just the right amount: Extracting modules from ontologies. Int. Conf. on WWW, p726, 2007

[15] Maedche, A. and Staab, S. Measuring similarity between ontologies Int. Conf. on Knowledge Eng. and Knowledge Mgmt., 251–263, 2002

[16] Winoto, P., McCalla, G., Vassileva, J.: An extended alternating-offers bargaining protocol for automated negotiation in multi-agent systems. Nat. Conf. on AI (2002) 969–970

[17] Yu, B., Singh, M.P.: An agent-based approach to knowledge management. Int. Conf. on Information and Knowledge Mgmt. (2002) 642–644