Forgetting Fragments from Evolving Ontologies

Heather S. Packer, Nicholas Gibbins, and Nicholas R. Jennings

Intelligence, Agents, Multimedia Group,
School of Electronics and Computer Science,
University of Southampton,
Southampton SO17 1BJ, UK
{hp07r,nmg,nrj}@ecs.soton.ac.uk

Abstract. Ontologies underpin the semantic web; they define the concepts and their relationships contained in a data source. An increasing number of ontologies are available on-line, but an ontology that combines information from many different sources can grow extremely large. As an ontology grows larger, more resources are required to use it, and its response time becomes slower. Thus, we present and evaluate an on-line approach that forgets fragments from an OWL ontology that are infrequently or no longer used, or are cheap to relearn, in terms of time and resources. In order to evaluate our approach, we situate it in a controlled simulation environment, RoboCup OWLRescue, which is an extension of the widely used RoboCup Rescue platform, which enables agents to build ontologies automatically based on the tasks they are required to perform. We benchmark our approach against other comparable techniques and show that agents using our approach spend less time forgetting concepts from their ontology, allowing them to spend more time deliberating their actions, to achieve a higher average score in the simulation environment.

1 Introduction

Evolving ontologies enable the completion of tasks and queries that were unforeseen during the design phase. Ontologies may evolve in use, by incorporating information from other ontologies. Due to the abundance of available ontologies it is possible that the uncontrolled evolution of an ontology may lead to an ontology that is large in size. Large ontologies require increasing amounts of resources to host, manage, and use. In time-critical environments where a fast response time is required, large ontologies can critically degrade response times. By forgetting concepts from an ontology in order to reduce its size the resources required to host, manage, and use the ontology can be reduced, therefore improving response times. However, forgetting concepts is a trade-off, because if too many concepts are forgotten, the quality of the answers will degrade, while forgetting too few concepts degrades the response time. For example, consider a fire brigade that uses an ontology to describe fire vehicle capability information on a portable device. A large fire requires immediate use of additional vehicles with the capability to move rubble. A nearby building site has suitable

vehicles, although information about their capabilities and operational requirements is not present in the fire brigade ontology. As such, this information must be reused from the construction vehicles manufacturers' ontologies by the fire brigade portable device, using a low-bandwidth mobile internet connection. Response time is critical so that damage to surrounding areas can be minimised, while inferring which construction vehicles are appropriate for the specific nature of the fire and will best protect the vehicles' operators.

In this paper, we focus on reducing the size of evolving ontologies in order to improve their response time. We propose to achieve this by removing fragments, sets of axioms that represent concepts [6]. Our approach removes fragments according to the frequency and recency with which they are used, and the cost of their acquisition, in terms of time and resources. We reduce the size of an ontology when it becomes too large to complete a task within a given period of time. By removing a fragment we hypothesise that the associated costs of using the ontology will be reduced.

We illustrate our work within a controlled environment so that, for now, we can regulate the information available to software agents, and use a standard success metric to compare state of the art approaches. We use an extension of RoboCup Rescue (RCR), a widely used multi-agent platform for agent research that simulates an emergency response scenario. RCR agents learn about concepts that enable them to rescue targets that have been victim to an earthquake in a virtual city. A team of agents has five second time limit in which to determine their actions, or else automatically forfeit their turn, and thus their ontology must enable them to use the information it contains and perform actions given the timeframe. The agents' ontologies evolve as they encounter tasks that require additional information to complete. By reusing the additional information in their ontologies, they do not need to incur the cost of re-learning it in future. However, with each additional fragment they learn, the performance of using their ontology degrades. Thus to ensure that agents can use their ontology efficiently, our approach forgets concepts from their ontologies. While we situate our approach using a specific multi-agent system exemplar, our algorithm is a general approach to selecting concepts to forget from an ontology, and could therefore be applied outside of our framework. For example, agents could learn and forget concepts from ontologies on the Semantic Web, although doing so might bring about challenges in managing inconsistencies and issues regarding trust. Likewise, our approach is not specific to the search and rescue domain; it can be applied to any domain. It should be noted that our forgetting approach does not guarantee that reasoning from an evolving 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, as discussed in the example above. Our approach is agnostic to a specific ontology language, however for simplicity we describe our approach in the context of OWL-Lite ontologies.

Against this background, we advance the state of the art in automatic ontology evolution, with our main contribution, a technique that selects a fragment which represents the least useful concept in the ontology to remove. In order to

achieve this, we contribute a technique that rates all concepts in an ontology and weights according to their use and acquisition cost. Then, in our empirical evaluation, we compare our forgetting approach to other state of the art approaches and evaluate the outcome using RCR's scoring system. Agents using our approach save 99.3% more civilians and 12.4% more of the city, compared with the next best approach.

We begin by introducing related work in Section 2. In Sections 3 and 4, we introduce RCR and highlight the benefits of forgetting. Section 5 describes our approach and Section 6 discusses the empirical evaluation by describing our benchmark techniques. Section 7 presents the results from our empirical evaluation and Section 8 concludes.

2 Related Work

This section presents the current state of the art in agent learning, evolving an ontology, pruning an ontology, and ontology complexity measures.

Currently, the agent community focuses primarily on augmenting an agent's ontology, instead of pruning it. In particular, Bailin and Truszkowski [7], Afsharchi et al. [8], Wiesman and Roos [9], and Soh [10] enable their agents to augment their ontologies with new knowledge, when agents have different domain models representing the same domain. Specifically, Bailin and Truszkowski's approach considers semantically equivalent representations, and Afsharchi et al. and Soh focus on the validation of the knowledge to be incorporated into the agent's ontology. These approaches augment an agent's ontology with one concept at a time, which increases the overhead cost of retrieving the information. In addition to this work, we presented an approach that reduces the cost associated with learning by augmenting a fragment into an ontology [11]. While the above discussed approaches allow agents to augment their ontologies, they do not prune them.

However, the Semantic Web community has produced methods that prune ontologies. In order for an agent to prune its ontology, it could apply the approaches of Eiter et al. [12] or Wang et al. [13] and [14], who provide algorithms to remove one concept from an ontology at a time. In contrast to the approach of [13], Eiter el al.'s approach requires axioms to first be translated from Description Logic (DL) syntax to rule representations, and translated back to DL syntax after the expansion of the rules and the removal of a concept has been performed. In contrast, Wang et al.'s approach can be applied to axioms without need of translation. Although [12]'s definition of forgetting is semantically meaningful, it has restrictions which limit the use of this technique to OWL-Lite and subspecies of OWL-Lite. These approaches enable an agent to evaluate the knowledge and remove concepts that are not required.

In this work, we chose to use the technique presented by Wang et al. to remove concepts from our agent's ontologies because it can ensure the consistency of an ontology after the removal of a concept. However, while this work focuses on removing a concept from the ontology, our approach focuses on the selection of concepts to remove and removes more than one concept at a time. Removing

more than one concept at a time results in an overall smaller ontology and reduces the number of times that the forgetting approach needs to be used, resulting in an increase in performance.

3 RoboCup OWLRescue Framework

The RoboCup OWLRescue (RCOR) framework extends the RoboCup Rescue (RCR) platform, which models the effects of an earthquake on a virtual city's buildings, civilians, and roads [15]. In RCR, 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.

Our RoboCup OWLRescue (RCOR) framework extends buildings to contain (possibly hazardous) chemicals, and extends civilians to have symptoms. The RCOR agents require different knowledge for each run because variables such as chemicals in buildings and civilians' symptoms are stochastic, and differ with each run. 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. In order for an agent to retain its core knowledge, two ontologies are used, a Domain Ontology (DO) from which an agent cannot forget, and an Evolving Ontology (EO) which an agent learns in and forgets from. Both ontologies are used when deciding on the action to take. 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 example, 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 motivate and describe our forgetting approach, which enables agents to reduce the size of their ontologies.

4 The Forgetting Approach

When an ontology becomes too large to use given a specific timeframe, our approach: first evaluates the concepts in its ontology to select which concept to remove; second selects a fragment of the concept that is deemed to be the most irrelevant; and third removes the concept so that the ontology remains consistent.

In order to motivate the need for forgetting concepts from an ontology, we first consider costs associated with a large ontology. Using an ontology incurs costs with hosting, maintaining, and using it, and the larger the ontology, the greater the need for physical memory and time to access required information. It is therefore beneficial to reduce the size of an ontology. We categorise three situations when forgetting concepts would be beneficial:

1. **Performance**: If the performance of querying an ontology falls below required parameters, for example after new information has been learned, removing older less used information can result in performance gains.
2. **Specialisation**: In order to retain specialisation in an ontology, information that is unrelated to the domain can be removed. This can occur because the specialist domain of an ontology is predetermined, or because the specialist domain of an ontology changes over time.
3. **Relevance**: Concepts and relationships in an ontology can become outdated when superseded by information. Forgetting out of date concepts therefore keeps an ontology up to date. Depending on the scenario it might also be beneficial to utilise OWL's deprecation semantics to mark out of date concepts as obsolete.

In our RoboCup Rescue example, the agents decided to forget when they cannot complete their actions within a single timestep. In our scenario, we only consider removing concepts that have been learnt through participating in tasks because we do not want to change an agent's core knowledge. This is because fire brigade agents require a different core set of concepts than an ambulance team because of their specialisation, thus, we only remove concepts from an agent's EO. The following three sections describe how we enable an agent to automatically evaluate the concepts within its ontology, select a fragment to prune from the ontology, and remove the fragment. We use the following example to explain our forgetting technique.

A fire brigade is tasked with extinguishing a building. The chemicals in the building are particularly toxic and human exposure results in severe damage to airways. The fire brigade wants to be able to increase the amount of equipment it carries in its first aid kit so that it can treat affected civilians, and augments an ontology fragment with such equipment. During this augmentation the agent learns about intubation equipment, but is unable to use the equipment because it is not specialised to do so. Therefore, this knowledge is not used during the agent's lifetime. The agent's response time is waning and it decides to reduce the size of its ontology in order to reduce the cost of using it.

4.1 Evaluate Concepts

Once a task agent determines that it needs to contract its ontology, it decides which concepts it wants to remove. Our approach enables an agent to evaluate the concepts in its EO using two influential factors:

1. How recently and frequently the concept is used to answer queries: This approach aims to reduce the cost of acquiring regularly required concepts so we therefore adopt the Least Recently, Frequently Used value (LRFU) used in [17] and is used in caching scenarios to select concepts to remove from a query agent's ontology. Each time a concept is used the LRFU increases as does LRFUs of the concepts which are used to define it. The LRFU is normalised into a ranking so that it can be summed with the concept acquisition below.
2. The cost of the original acquisition of the concept: this cost is recorded in milliseconds and is recorded by our learning approach, in order to be used here. The acquisition value depends on the availability of the concept, and the network bandwidth available to transfer the fragment from another agent. The acquisition cost is normalised into a ranking, so that its value can be summed with the LRFU.

In order to indicate the usefulness of a concept, we sum these two factors to calculate a concept forgetting value (CFV) for each concept in an agent's EO (as shown in Equation 1).

$$CFV = LRFU + AC \tag{1}$$

where CFV is the concept forgetting value of a concept in an agent's EO, the $LRFU$ is the LRFU value for a concept, and AC is the acquisition value of the concept. A low CFV weighting indicates that the concept has not been used recently, often, and was inexpensive time wise to acquire, and a high CFV weighting indicates that the concept is used recently, frequently and was expensive to acquire. A medium CFV weighting can indicate that LRFU is high and AC is low, or that AC is high and LRFU is low, and as such the likelihood of the concept being forgotten is lower than those with a low CFV weighting. In more detail, the LRFU is calculated for each concept in the agent's EO using Algorithm 1. This algorithm shows how an agent calculates the $LFRU$ value for each of its ontologies concepts, where $concept(EO)$ is a function that holds

the set of concepts in an agent's EO, $T = \{< t_1, c_u 1 >, \ldots, < t_n, c_u n >\}$ is a set of tasks where each task is a tuple representing the task (t) and the set of concepts required to complete the task (c_u), all concepts in c_u are a subset of $concept(EO)$, and all concepts in the EO have a $LFRU$ weighting which is represented using a tuple $< c, LRFU >$. The LRFU weighting for each concept is calculated over time. After each time period each concept's LRFU is calculated, by increasing the value by 1 if it is used and decaying it exponentially when it is not, so that concepts not used recently have a lower value. In our RoboCup Rescue example, concepts' LRFU weightings are calculated each timestep and are represented by tasks in the Algorithm because an agent has to complete a task per timestep. Depending on the scenario, it may be appropriate to weight the AC or LRFU differently. For example if network bandwidth fluctuates, the acquisition cost is time-sensitive, and therefore it would be appropriate to weight it lower than LRFU. In our environment available bandwidth does not change, and therefore we do not apply weightings when calculating the CFV.

Algorithm 1 Algorithm calculating the LRFU for each concept in an agent's ontology.

Ensure: $concepts(EO) \neq \emptyset$
Ensure: $T = \{< t_1, c_u 1 >, \ldots, < t_n, c_u n >\}$ /* T is the set of tasks, where tasks require a set of concepts to complete them. */
Ensure: $c_u \neq \emptyset$ /* the set of concepts used for current task */
Ensure: $c_u \subset concepts(EO)$
Ensure: $\forall c \in concepts(EO) = < c, lrfu >$
1: **for all** T **do**
2: **for all** $c \in concepts(EO)$ **do**
3: **if** $c \in c_u$ **then**
4: $c = < c, lrfu + 1 >$
5: **else**
6: $c = < c, e^{-lrfu} >$
7: **end if**
8: **end for**
9: **end for**

Once a concept's LRFU factor has decayed so that the acquisition cost becomes more influential in the weighting, an agent can determine which concept from a set of concepts that have the same LRFU to forget. It is more likely that concepts will have different acquisition costs due to different agent's network location and bandwidth, than different a LRFU because concepts decay exponentially. Performance wise, it is better for the agent to forget concepts that are inexpensive to acquire because the cost of re-acquiring them is less, compared to concepts that are expensive to acquire. To summarise, the agent selects the concept with the lowest rating in its EO to remove. In our example (see Figure 1), the agent selects the concept labelled *endotrachealTubes*, which is a piece of intubation equipment, because it has the lowest weighting. In the next section, we describe how the agent removes a fragment representing the selected concept.

4.2 Select Concepts

Once the agent has selected a concept it desires to forget, it creates a fragment (made up of multiple concepts) representing that selected concept so that the

agent can benefit performance wise from a smaller ontology. We also hypothesise that an agent can benefit from removing more than one concept at a time so that it can perform forgetting less often than forgetting methods that forget less concepts (as proposed by other state of the art approaches, see Section 2).

In order to select concepts to prune, the agent generates a fragment representing the selected concept and selects concepts with a similar CFV weighting to prune. The fragment is generated using the basic segmentation technique presented in [6], where the technique selects the target concept first, in our example (see Figure 1) the target concept is *endotrachealTubes*, and selects concepts by traversing the ontology's concept hierarchy upwards all the way to the root class. It then traverses downwards to the target's leaf classes. Additionally, any links across the hierarchy from any of the traversed classes are followed upwards but not downwards. Once there are no concepts to traverse, the traversed concepts form a fragment representing the target concept (described in more detail in [6]).

In order to detail how our agent selects the concepts to remove from the fragment, we formally introduce the components described above. Let: l be the capacity limit at which the agent is required to prune concepts from its EO; W be the set of weightings for the concepts contained in the EO, where $W = \{w : C \in concepts(EO) \wedge w = weight(c)\}$ and $c_1 \dots c_n \in concepts(EO)$; f_{o_q,c_t} be the fragment representing the concept to be forgotten, where o_q is the query agent's ontology (where $o_q = DO \cup EO$) and c_t is the concept to be forgotten; $W_{f_{o_q,c_t}} = \{W : C \in f_{o_q,c_t} \wedge w = weight(c)\}$ be the set of concept weightings associated with the concepts contained in f_{o_q,c_t}, where $concepts(f_{o_q,c_t}) = \{c_1, \dots c_n\}$. Using this formal notation we describe how we select the concepts to forget in Algorithm 2, which is run over all concepts in the EO.

Algorithm 2 Lowest Weighted Concept Selection Technique: This algorithm is used to select the concepts to be pruned from an agent's EO.

Initialise: $c_t \leftarrow null$
Initialise: $conceptsToRemove \leftarrow \emptyset$
Initialise: $w_{c_t} \leftarrow +\infty$ `/* the weight of the concept; calculated from the LRFU value and Acquisition Cost of` c_t `*/`
{finds the concept with the lowest concept weighting}
for $\forall c \in concepts(EO)$ **do**
  if $w_c < w_{c_t}$ **then**
    $c_t \leftarrow c$
    $w_{c_t} \leftarrow w_c$
  end if
end for
$conceptsToRemove \leftarrow conceptsToRemove \cup \{c_t\}$
{finds all the concepts in the fragment with a similar concept weighting to c_t}
for $\forall c \in concepts(f_{o_q,c_t})$ **do**
  if $|w_c - w_{c_t}| \leq t$ **then**
    $conceptsToRemove \leftarrow conceptsToRemove \cup \{c\}$
  end if
end for
return $conceptsToRemove$

In our example, Figure 1 shows the ontology fragment representing concept *endotrachealTubes* which has weight $w_{c_t} = 0.02$, thus our selection algorithm selects the concepts *endotrachealTubes*, *laryngoscopes*, *connellAnotomicMask*, and

intubationEquipment to forget (these concepts have a shaded background). These concepts have not been used by the agent so they have a low CFV because the agent has not had specialist knowledge to use the equipment, because it specialises in extinguishing fires and only supports first response first aid and triage.

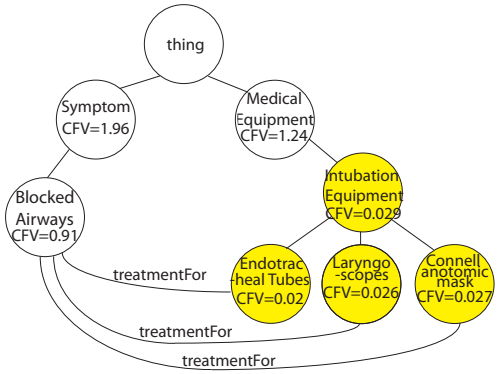

Fig. 1. Concepts selected to be forgotten, the curved lines represent relationships (domain and range restrictions) between concepts.

Once our agents have selected the concepts they desire to remove they then remove these concepts from their ontology, this is described in the next section.

4.3 Remove Concepts

After the agent has selected the concepts that it desires to remove, the agent then prunes these from its ontology. In order to prune these concepts from an ontology we use the technique presented in [13] so that the ontology remains consistent. We illustrate two basic examples in Figure 2, where the concept B is removed from each example. In the first example, the axioms $A \sqsubseteq B$, $B \sqsubseteq C$ are replaced with $A \sqsubseteq C$; and in the second example $A \sqsubseteq B$, $B \sqsubseteq C$ and $B \sqsubseteq D$ is replaced with $A \sqsubseteq C$ and $A \sqsubseteq D$.

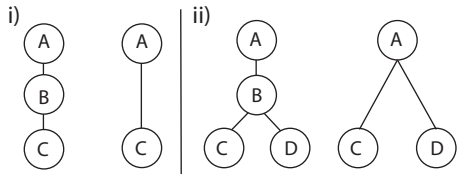

Fig. 2. Two examples of removing concept B from an ontology, when the removed concept has a single, and multiple children, respectively.

Algorithm 3 Pseudocode of the RoboCup simulator

```
Require: function: contains(set, element) returns true if set contains element
Require: simulator ← RoboCup rescue simulator
Require: agent ← RoboCup rescue agent
 1: simulator.generateFires()
 2: simulator.generateBlockades()
 3: simulator.generateCivilians()
 4: for timestep ∈ timesteps do
 5:     target = getFirstTarget()
 6:     targetInfo = agent.getInformation(target)
 7:     if ¬ contains(agent.ontology, targetInfo) then
 8:         fragments ← requestFragements(targetInfo)
 9:         axioms ← selectAxioms(fragments)
10:         agent.ontology.learn(axioms)
11:     end if
12:     if ¬ contains(agent.vehicle, requiredEquipment) then
13:         travelToCentre()
14:         changeVehicle()
15:     end if
16:     rescue(target)
17:     simulator.update()
18: end for
```

5 Evaluation

In order to evaluate our approach, *forget-fragment*, we compare the effectiveness of agents using different forgetting techniques (discussed below) to: rescue civilians; put out burning buildings; to evaluate the requirements of rescuing a target against its RoboCup score (see Section 2); and investigate the amount of time used to forget. Similar to the RCR Competition, our simulation allows a team of agents five seconds to complete an action in a timestep, and are given two thousand 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 there are ten of each of the ambulance, fire brigade and police agents and they use the learning technique presented in [11] when they encounter unknown concepts or do not have the right equipment to rescue their target. This learning technique selects fragments from ontologies about a requested concept, and demonstrated the most efficient learning algorithm (in terms of resulting performance) compared to benchmark approaches. We compare our technique to the following approaches:

Forget Concept This approach removes all concepts and relationships related to the selected concept, where eo^{-c_p} and c_p is the concept to be pruned from the ontology. This technique removes one concept at a time, and is comparable to the techniques presented by [12] and [13].

Forget Tree This approach extends the above approach by selecting a subtree from the hierarchy of concepts in the agent's EO. The subtree is selected by comparing the weight used for each concept (see Algorithm 4), where $eo^{c_{tree}}$ and c_{tree} is a concept represented by the fragment (which is a subtree) being pruned from the ontology. Removing a connected subtree can result in removing a subtree, branch, or extraction of a subtree (shown in Figure 3).

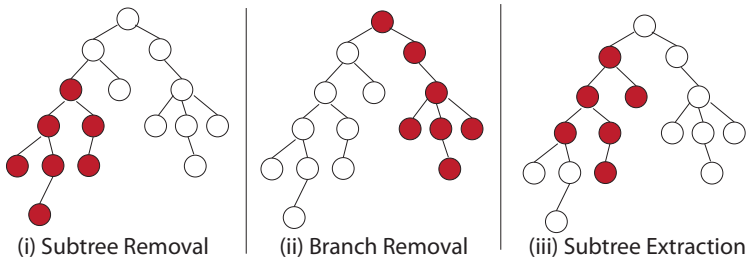

Fig. 3. Subtree Removal, Branch Removal and Subtree Extraction, where the highlighted nodes are removed from the graph.

Forget Redundant This approach removes all concepts that are not used in future queries. We provide a list of the future queries to the agent at the start of the simulation, which have been recorded on a dummy run using the same random seed. This is the only agent that requires a complete list of future queries. This agent is not limited by a capacity.

Forget Nothing This approach does not remove any concepts from the agent's ontology. Hence this agent is not limited by a capacity.

We put forward these four approaches: **forget-concept**, **forget-tree**, **forget-redundant**, **forget-nothing**, as benchmarks for the performance of our forgetting approach, **forget-fragment**. 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, using the standard Kobe map provided with RCR. 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.

Algorithm 4 This algorithm is used to select the concepts which are connected by their concept weighting, to form a subtree, to be pruned from an agent's EO.

Require: **function:** $getConceptWithMinimalWeight(set)$, returns tuple $< concept, weight >$ with the lowest weight in the set.
Require: $c_t \leftarrow null$, $w_{c_t} \leftarrow null$
Require: $conceptsToRemove \leftarrow \emptyset$
Require: $CH \leftarrow \{\emptyset\}$
1: $< c_t, w_{c_t} > = getConceptWithMinimalWeight(concepts(EO))$ {w_{c_t} is the lowest weight in EO }
2: {traverses the children of c_t}
3: $CH \leftarrow children(c_t)$
4: **for** CH **do**
5: **for** $ch \in CH$ **do**
6: **if** $|w_ch - w_{c_t}| \leq t$ **then**
7: $conceptsToRemove \leftarrow conceptsToRemove \cup \{ch\}$
8: $CH \leftarrow CH \cup \{children(ch)\}$
9: **end if**
10: **end for**
11: **end for**
12: $P \leftarrow parents(c_t)$
13: **for** $\forall$ P **do**
14: **for** $\forall$ $p \in P$ **do**
15: **if** $|w_p - w_{c_t}| \leq t$ **then**
16: $conceptsToRemove \leftarrow conceptsToRemove \cup \{p\}$
17: $P \leftarrow P \cup \{parents(p)\}$
18: **end if**
19: **end for**
20: **end for**
21: **return** $conceptsToRemove$

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]

[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

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 1. 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 1. The next section presents the results and our analysis of our evaluation.

6 Results

We compare our results using standard measures of the RoboCup Rescue *scorevector* [18] scoring system: saved civilians, and buildings unburned. The agents using the *forget-fragment* approach outperform the other agents using benchmark approaches, by having the highest average number of civilians alive (99.3% more civilians saved compared to the next highest approach, *forget-tree*) and percentage of the city that is unburned at the end of the simulation (12% more than the next highest approach, *forget-tree*), see Table 2 and Figures 4 and 5.

[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

Table 2. Comparison of average results for each forgetting technique.

Technique	Percentage of City Unburned	Percentage of Civilians in Refuge	Number of Times Forgot	Number of Concepts Forgotten per Forget
Forget-Concept	31.5	29.1	11.0	50.4
Forget-Tree	33.0	29.0	10.3	50.3
Forget-Redundant	31.5	19.8	50.3	215.1
Forget-Nothing	31.5	1.4	0.0	0.0
Forget-Fragment	37.1	58.0	3.1	44.1

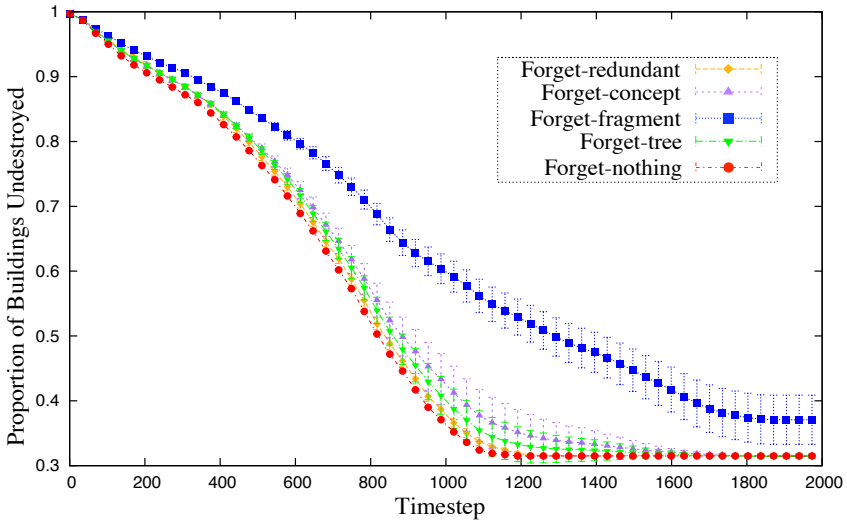

Fig. 4. Chart showing the percentage of buildings unburned for each forgetting approach.

The agents using our approach reduce the size of their ontologies by removing a fragment of a concept, instead of removing trees that contain a smaller number of concepts or removing a single concept. By removing a higher number of concepts from an ontology the agents forget less frequently, and ultimately spend less time deciding what to forget and forgetting than the other approaches, with the exception of the *forget-nothing* approach (*forget-tree* forgets 332% more times than *forget-fragment*, and 14% more concepts when it forgets), see Table 2 and Figure 6. We also note that despite our approach forgetting less concepts than other approaches, it still outperforms them, because it spent less time forgetting, thus demonstrating the trade-off between spending time managing an ontology so that it performs well, and spending time querying a large ontology. Our approach helps agents save civilians at a faster rate than the other approaches, because the fire brigade agents put out more fires thus reducing the injuries to the civilians in the simulation. The *forget-tree* and *forget-concept* approaches' performances are similar; this is because they have a similar forgetting frequency, and thus spend a similar amount of time performing actions derived from their

ontology. The *forget-nothing* approach is unable to submit any commands because it took too long to deliberate over its actions. Despite the *forget-redundant* approach having perfect foresight it spent too long forgetting because it forgot every unnecessary concept increasing the forgetting frequency compared to the *forget-fragment* approach.

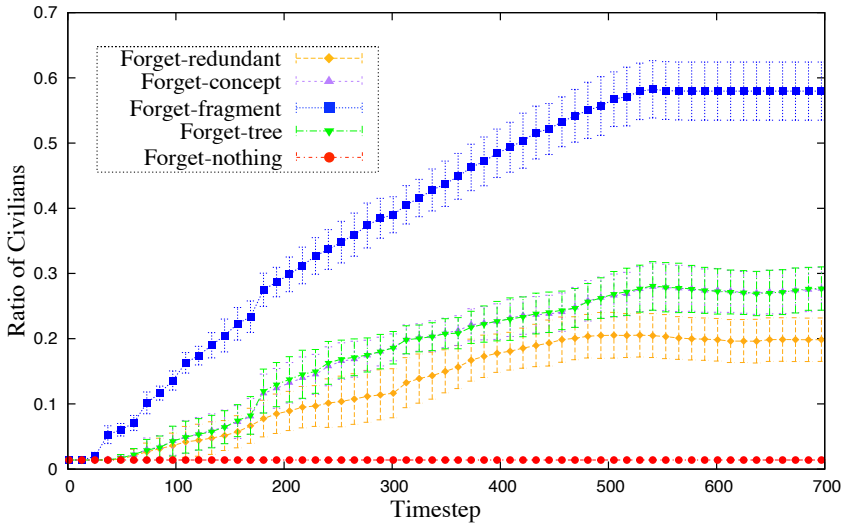

Fig. 5. Chart showing the percentage of civilians rescued for each forgetting approach, our results range from 0 - 700 timesteps; there are no changes to the result after 700 timesteps.

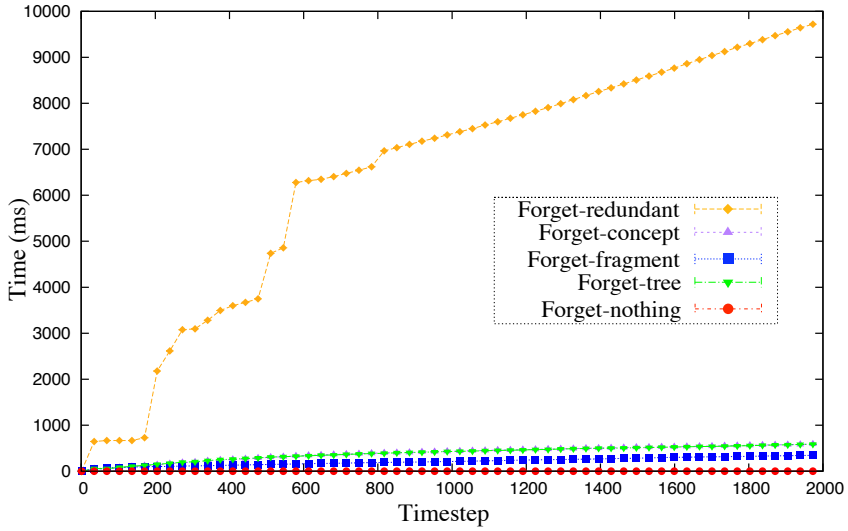

Fig. 6. Chart showing the time spent forgetting for each forgetting approach.

7 Conclusions

In this paper we present a novel technique that can be used to remove a fragment from an ontology. Our technique is tested using an agent-based search and rescue domain, but is generalised and applicable to any scenario where an ontology evolves limitlessly over time. In order to support this contribution we have also developed a semantic extension to the RoboCup Rescue framework which enables its agents to evolve their ontologies, and a technique that rates all concepts in an ontology with a weighting. We have also implemented benchmark approaches which were used to compare the forgetting approaches. Our evaluation shows that our method saves 99.3% more civilians and 12.4% more city area, compared with the next best approach. For the future, we plan to investigate the benefits of using our technique in scenarios that require random and regularly used concepts. This investigation aims to explore the hypothesis that our approach will remove the concepts acquired from the random queries, while generally keeping the fragments required for the regularly repeated queries. We will also explore other motivations for an agent to forget concepts, for example agents that can predict future queries so that they can prioritise forgetting concepts which are unlikely to reoccur.

References

1. Berners-Lee, T. and Hendler, J. and Lassila, O.: The Semantic Web. Scientific American **284**(5) 28–37, 2001
2. Ashburner, M. and Ball, C.A. and Blake, J.A. and Botstein, D. and Butler, H. and Cherry, J.M. and Davis, A.P. and Dolinski, K. and Dwight, S.S. and Eppig, J.T. et al.: Gene Ontology: tool for the unification of biology Nature genetics **25**(1) 25–29, 2000
3. Sidhu, A.S. and Dillon, T.S. and Chang, E. and Sidhu, B.S.: Protein ontology: vocabulary for protein data IEEE Computer Society, 2005
4. Rogers, J. and Rector, A.: The GALEN ontology Medical Informatics Europe 174–178, 1996
5. Stuckenschmidt, H. and Klein, M.: Structure-based partitioning of large concept hierarchies ISWC, 289–303, 2004 (Springer)
6. Seidenberg, J., Rector, A.: Web Ontology Segmentation: Analysis, Classification and Use. Proceedings of the 15th International Conference on WWW, NY, USA, ACM, 13–22, 2006.
7. Bailin, S., Truszkowski, W.: Ontology Negotiation between Intelligent Information Agents. The Knowledge Engineering Review **17**(01)7–19, 2002.
8. Afsharchi, M., Far, B., Denzinger, J.: Ontology-guided Learning to Improve Communication between Groups of Agents. Proceedings of the 5th international joint conference on Autonomous agents and multiagent systems, Hakodate, Japan, 923–930, 2006.
9. Wiesman, F., Roos, N.: Domain Independent Learning of Ontology Mappings. International Conference on Autonomous Agents: Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems, New York City, New York, USA **2**:846–853, 2004.
10. Soh, L.: Multiagent, Distributed Ontology Learning. Working Notes of the 2nd AAMAS OAS Workshop, Bologna, Italy, 2002.
11. Packer, H.S., Gibbins, N., Jennings, N.R.: Collaborative Learning of Ontology Fragments by Cooperating Agents.. In: Web Intelligence-Intelligent Agent Technology International Conference, Toronto (2010).
12. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H., Wang, K.: Forgetting in managing rules and ontologies. Proc. of the International Conference on Web Intelligence, 411–419, 2006.
13. Wang, Z., Wang, K., Topor, R., Pan, J.: Forgetting Concepts in DL-Lite. Lecture Notes in Computer Science **5021**:245, 2008.
14. Wang, K. and Wang, Z. and Topor, R. and Pan, J.Z. and Antoniou, G. Concept and Role Forgetting in ALC Ontologies. The 8th International Semantic Web Conference (2009).
15. Kitano, H. and Tadokoro, S. Robocup rescue: A grand challenge for multiagent and intelligent systems AI Magazine volume 22:1-39 (2001).
16. Wang, K., Sattar, A., Su, K.: A Theory of Forgetting in Logic Programming. Proc. of the National Conference on Artificial Intelligence, **20**(2):682, 2005.
17. Lee, D., Choi, J., Choe, H., Noh, S.H., Min, S.L., Cho, Y.: Implementation and Performance Evaluation of the LRFU Replacement Policy. 106–111, 1997.
18. Siddhartha, H. and Sarika, R. and Karlapalem, K.: Retrospective analysis of RoboCup rescue simulation agent teams Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems **(2)** 1365–1366 (2009)