

# Training a Cognitive Agent to Acquire and Represent Knowledge from RSS feeds onto Conceptual Graphs

Alexandros Gkiokas \* Alexandra I. Cristea †

University of Warwick

Department of Computer Science

Coventry, CV4 7AL, UK

\*e-mail: a.gkiokas@warwick.ac.uk †e-mail: a.i.cristea@warwick.ac.uk

**Abstract**—Imitative processes, such as knowledge transference, have been long pursued goals of Artificial Intelligence (AI). The significance of Knowledge Acquisition (KA) in animals and humans has been studied by scientists from the beginning of the 20th century. Our research focuses on observational imitation through agent-user interaction, for acquisition of symbolic knowledge. The cognitive agent (CA) emulates an imitative learning system, trained for the purpose of learning to represent knowledge, acquired from Rich Site Summary (RSS) feeds. It learns to autonomously represent that knowledge in a manner that is both logically sound, and computationally tractable, through the fusion of Conceptual Graphs (CG) and Reinforcement Learning (RL). The novel algorithm enabling this agent extends Reinforcement Learning, by approximating decisions via exploitation of distributional and relational semantics governing the knowledge domain.

**Keywords**—*Cognitive Agent; Reinforcement Learning; Conceptual Graphs; Expert Systems; Imitation Learning.*

## I. INTRODUCTION

Similar to recent research projects, which try to acquire knowledge from the Internet, such as KnowRob [1] or Never Ending Image Learner [2], this paper focuses on the acquisition of knowledge found on the Internet, a large source of ever growing data. We examine RSS feeds [3] as the source of knowledge, due to their parsimony and quality of information. RSS has been around for more than a decade, and is used to display and update in real time information from various news agencies, websites and social media, and represent an important and stable, widely used means of information distribution of the modern world.

The CA emulates, but does not simulate, a perceptive process and learning system, based upon psychological studies in humans [4] and animals [5]. The CA purpose is to acquire knowledge; for that we train the agent using RL, instead of heuristics, which is the norm. Doing so, we hypothesise makes the agent adaptive, and enables it to, domain-independently, augment its acquisition capabilities (KA) and thus, expand its knowledge in an autonomous or semi-supervised manner. The CA described in the paper, employes a symbolic-based imitation learning process (observational imitation) as defined by previous research

[5][6]. The actual algorithm emulating the imitational process, is performed through the fusion of AI technologies as well as a new approximation algorithm we've developed for RL, in order to enable the CA to keep learning indefinitely, after training. Machine Learning (ML) literature treats an increased state space as a problem rather than an aid [7], something usually dealt with state or action approximation. In contradiction to this belief, we attempt to show that by re-using existing reinforced episodic experience, through semantic approximation, learning can benefit from an expanding state space. CA learning to acquire information, cannot afford to perform random actions (e.g., Monte-Carlo search) as part of the Markov Decision Process (MDP) used by RL, as this leads to poor performance or slow convergence [8]. Modelled after imitation [5] and Programming By Example (PBE) [6], we hypothesise that the rate at which prior experience is exploited can alter the decision making process of RL, through the fusion of approximation methods. By employing semantics and reasoning on underlying semiotics governing the symbolic particles found in the knowledge domain, we attempt to restrict the amount of random decisions, and replace them with known actions, assumed to be correct. The purpose of the RL algorithm is to obtain RSS feeds, and then learn how to isomorphically project the feed onto a CG [9].

Section II visits related research, describing the context in which the current proposal has been created. Section III describes the main problems we are attempting to solve in this paper in greater detail. Section IV describes the learning architecture of the CA. In Section V, we describe the algorithm that implements semantic approximation, and thereby extends RL. This is followed by a discussion in section VI and conclusions in Section VII.

## II. RELATED RESEARCH

Existing research on automated KA has focused on heuristics [10][11] or statistical and probabilistic analysis, either via Natural Language Processing (NLP) or as part of the Knowledge Representation (KR) generation [12] of CG. Other researchers have used CG [13] for mining, yet others used semantics and morphosyntactics [14] in order to extract concepts (not CG). Older research has seen combination of concept maps and neural networks, such as the KBMiner [15] for KA or mining. Direct comparison

with all the aforementioned research won't do justice to either the work described, or ours, as it would partially compare one side of our work (KA) to research that was not focusing on CA or imitation learning. As such, the only comparison possible, would be the performance of our fully trained CA, against the performance of the algorithms described above.

AI research in imitation has historically [5] attracted attention from roboticists, as it stands to be the field that would mostly benefit from knowledge transference. Whilst research on knowledge transference in software agents [6] has focused on inference and pattern prediction for Web interfaces, programming automated tasks, etc., we address bridging CA and RL, for acquiring knowledge (KA), as part of an imitation process. Lieberman set the foundations for PBE [6] on which a great deal of our research relies on. However, most of the research by Lieberman (or his associates) does not deal with agents (much less CA), nor does it deal with KR. The only research by Lieberman on agents [16], was for Web browsing and text mining, which has fundamental differences from KA. Other researchers using PBE have done quite similar research (with respect to intentions and processes) [17], but still the comparison is incomplete as different technologies were used, and for a different purpose.

In comparison to both PBE and KA research, we employ RL, in order to train the agent. The algorithm described in this paper here can be said to belong to the family of stochastic semi-supervised learning algorithms. The agent does not attempt to detect patterns, mine information, or summarize text, but instead focuses only on projecting information into a formal representation (KR structure), such that it can be manipulated for other knowledge management operations (which are outside the scope of this paper). *Knowledge is making sense of information*, and for *learning to make sense of RSS*, we use RL, which has been widely used for a variety of AI tasks, ranging from robotics [18], to games, to intelligent agents [7]. RL research has, however, been conscious about approximating and regressing state-actions since its inception [8][7], and there exists a plethora of methods on how to effectively regress and approximate, in order to infer correct policies. However, our approach using semantically-driven approximation in combination with distributional semantic similarity calculations, should provide grounds to support episodic experience re-usability in CA learning via RL.

### III. PROBLEM SETTING AND OBJECTIVES

The primary objective of the described CA is to obtain feeds and construct the correct Conceptual Graph (CG) from each feed. We assume each feed to contain knowledge that can produce one or more CG [9]. Traditionally, CG creation has been assigned to knowledge engineers, human users acting as experts who construct the correct CG. The second, more practical objective, is to enable the CA to learn how to acquire knowledge by reusing its own experience, thereby aspiring to enable the CA to self-augment its capabilities. By creating a trainable agent to

acquire knowledge, we aim to enable the CA with certain advantages over conventional approaches:

- The agent should be able to process any domain of symbolic information.
- The agent should be adaptive, thus be able to learn training indefinitely and adapt to non-monotonic knowledge (through the use of RL).
- The agent after being trained, should be able to autonomously acquire knowledge without the need of a knowledge engineer.
- The (trained) agent should be able to deal with unknown domain particles (via Semantic approximation) and uncertainty.

### IV. COGNITIVE AGENT ARCHITECTURE

According to the criteria set by Lawniczak and Di Stefano [19], the described agent could be partially called a CA. It satisfies four of the five criteria:

- Perceives Information (RSS) in its environment (Web) and provided by other agents (through textual queries)
- Reasons about that information using existing knowledge (through ML, RL and Semantics)
- Responds to other agents, human (via Web interface) or artificial (via textual responses)
- Learns and changes the existing knowledge if the newly acquired information allows it

In fact, the only criteria not satisfied, is the ability to judge obtained information using existing knowledge [19]. The novelty in the design of the cognitive agent is not merely the fusion of different A.I. technologies, but finding a novel usable expression of the notion that a cognitive agent can be trained to learn how to acquire knowledge, by reusing its own accumulating experience. It is however important to note that this agent is designed to deal only with symbolic knowledge, and not with continuous, metric or numeric information. As the source of information, RSS feeds are used. RSS offer certain distinct advantages over other sources of information. Thelwall, Prabowo and Fairclough [3] mention that RSS feeds are one of the very few sources on the internet, which may be able to offer quality information relevant to specific subjects. The Knowledge Representation (KR) meta-structure used is Sowa's CG [9], which are formal structures, based upon philosophical principles, that offer logically sound reasoning operations [20]. They also offer a few advantages over other KR structures, such as Web Ontology Language (OWL) or the Resource Description Framework (RDF). They do not use a subject-predicate-object format, but are relation centric. Their greatest advantage is that they are visually simple to understand, even by (human) users who are not knowledge engineers or computer scientists. They also offer a computationally affordable [13] and tractable way, in order to perform various logic and reasoning operations [20], which are important and desirable for the future of this project. The agent's main task is to parse feeds,

and construct the correct Conceptual Graph (CG)  $g$  such that  $g$  correctly represents the knowledge encapsulated in the feed  $r$ . What this implies, is that the agent learns how to read text found on the internet and represent it internally. The actual manipulation and management of the acquired knowledge is outside of our scope, we only focus on enabling such a process so as to emulate imitation and observation learning. We model this, as shown in (1), indicating that the imitating function  $f$  is a mapping or projecting function.

$$f : r \mapsto g. \quad (1)$$

The function  $f$  is composed of a parsing  $f_p$  and an isomorphic projection  $f_m$  so that  $r \cong g$ , as shown in (2).

$$f_p(r) = t_r, f_m(t_r) = g_r \therefore (f_m \circ f_p) = f_m(f_p(r)). \quad (2)$$

Those two functions compose the actual operation of the CA: first the feed  $r$  is parsed as a set of tokens  $t_r$ , and then that set is isomorphically projected onto graph  $g$ . If parsing  $f_p$  does not produce the correct output, then projection  $f_m$  may have undefined behaviour. Function  $f_m$  is the subject of RL, and by using  $f_m$  as a policy for projecting  $r \cong g$  (1), we may replace  $f_m$  with  $Q$ , as shown in (3).

$$f_m : r \cong g \Leftrightarrow Q(s_t, a_t). \quad (3)$$

where (3)  $Q$  is the Markov decision, calculated as a policy for a given state  $s_t$  when taking action  $a_t$ , at step  $t$ . Function  $f_p$  parses the feed  $r$ , into the set of tokens  $t_r : \{t_1, \dots, t_n\}$  by tokenizing it. Parsing and tokenizing is a NLP task, which we do not include within the projection, or the learning material for the agent; instead we rely on simple white-space tokenizing, using a grammatical rule-based approach [21]. We chose to do so, in order to simplify the decomposition operation, into a minimal algorithm with small complexity, and thus, focus on the projection algorithm. Using heuristic parsing [21], amongst others, this process concatenates words that are encapsulated by quotes (single or double) into a single token. Using morphosyntactic (part-of-speech) tagging [22], the parsing algorithm also finds tokens, which denote a person, and concatenates them into a single token. Trailing “s” found after a word, are removed, and replaced by the token “owns”. Free particles, such as “the”, “a”, “an” are also removed. All implied tokens, not found in the actual feed  $r$ , should be discoverable by the CA, as it has no other way of inferring those tokens. The tokenizing should capture the ontology of the feed, in the resulting token set  $t_r$ .

Following the tokenizing, the projection starts, and is done in two stages, and represented as a MDP in the form of an episode of RL:

- In function  $f_n$  tokens have to be converted to a vertex (Concept or Relation) [9].
- In function  $f_c$  relations have to connect to concepts with an edge [9].

The state is always represented by the same CG (see figure 1), which is being constructed by the RL algorithm, and is only finished when a terminal state is reached [7]. The construction of the CG, is what that the CA learns to perform, described by the MDP used by RL. In a

temporal sense, each time-step in the MDP is describing the algorithm constructing a piece of the CG. As the graph

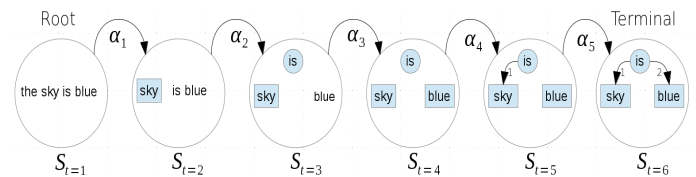


Figure 1. MDP as a RL State-Path

is constructed by the feed tokens, the graph vertices are eventually connected, in order to form the final graph. We employ states and actions as per the RL literature [7], and the human trainer provides a reinforcement at the end of each episode, rewarding that CG. The CA uses that reward to reinforce correct CGs. We chose to use RL over other methodologies, because it deals best with partially observed MDP [7]. The  $Q$ -Policy value is calculated by the SARSA algorithm [7], as shown in (4).

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha [R_{(t+1)} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]. \quad (4)$$

The algorithm back-propagates that reward  $R_{t+1}$  for terminal states, when re-iterating previous states, using a discount factor  $\gamma$  and learning rate  $\alpha$ . We chose SARSA over Q-learning, as it is assumed to be more greedy with respect to immediate rewards [7]. The actual action  $a_t$  in (4), has the binary nature described by (figure 1) and, can logically be represented, as shown in (5).

$$a_t \leftarrow f_n(t_r) \vee f_c(g). \quad (5)$$

The disjunction signifies the two-staged construction as a temporal process: either vertices, or edges, are created. The action search space can vary, and is denoted by  $S_p a_t$ , as shown in (6), where  $n$  is the number of all vertices in  $g$ .

$$S_p a_t(f_n) = 2^n. \quad (6)$$

A correct decision, could be expressed as a search in that action space, where the algorithm has to find the correct vertex types for graph  $g$ , or establish the correct edges between them. The action space is also proportional to the number of tokens in the feed, and is dynamically parametrised by the decided relations  $r$  and concepts  $c$  of  $g$  where  $x_r$  is the available edges for a relation, as shown in (7).

$$S_p a_t(f_c) = \int_1^c (r * x_r). \quad (7)$$

Each relation vertex  $r$  can have minimum one edge to a concept, and maximum an edge for each concept. Only relations can establish edges to concepts [9].

The CA is complemented by a semantic graph, represented internally by a Word-Net instance [23]. Contrary to CG literature [20], we do not employ two distinct maps for relations and concepts, as Word-Net has its own classification and categorization. All semantic operations are delegated to Word-Net, which acts as a semantic authority. Using an action  $a_t$  which has been reinforced, implies that this action has been performed at least once. Normally, this action would be randomly chosen from the space of possible actions  $S_p a_t$  and only after being reinforced, could

it become a policy. Whilst this may not be a problem for other AI-related areas, it is of the utmost importance for the CA to decide the right actions the first time, and not wait for a policy. In reality, the same, identical states would rarely be revisited, in order for a policy to become of much use.

## V. SEMANTIC APPROXIMATION

A vast search space is a problem in RL when computing or searching for policies [7]. Conventionally, RL is a semi-random MDP, often employing a Monte-Carlo search. However, due to the importance of semantics in symbolic domains [23], the described algorithm for selecting an action for a state, assuming  $s_t$  has never been experienced before, attempts to avoid random selections, and instead relies on semantic distance and word similarity, in order to reason, as to why an action should be performed. The premise upon which this notion is based, is that of reusing prior positive experience. Furthermore, the operation is to approximate state similarity, and then approximate action similarity through semantics. Assuming enough reinforced experience exists, the algorithm should have enough sources to reason as to why an approximation *could* or *should* work. The semantic similarity between the particles that describe a state is inherited from the lexicon of the domain, for which knowledge is acquired. Those semantic relations are exploited in a logic fashion, so as to reason about re-using, known to-be-correct (via reinforcement) actions. Each episode in memory, can be positively or negatively rewarded by the trainer. The first operation is to approximate similarity to all positively rewarded graphs in memory, as shown in (8), by using a Vector Space Model (VSM) [24].

$$S_r = \frac{M * V_r}{\|M\| * \|V_r\|}. \quad (8)$$

Sparse matrix  $M$  (8) contains token-frequencies, and is created by indexing each unique CG as a column, and each token used as a vertex in a CG, as a row.  $V_r$  is the (respective to matrix  $M$ ) row vector of tokens representing the feed  $r$ . The VSM discovers similarity based upon distributional semantics, and returns a column vector  $S_r$  of degrees of similarity, as shown in (9), representing how similar each CG in memory, is to the input vector  $V_r$ .

$$S_r = [\cos(\Theta_1), \dots, \cos(\Theta_n)]. \quad (9)$$

For each identical particle in graph  $g'$  (the graph already in memory) that is similar to  $g$  (e.g., the one currently being created), we perform the same action. For example, if  $a_t$  converted  $t$  to concept  $c$ , we assume the same decision to be valid. Thereby by approximating the state  $s_t$  we can obtain policy  $Q(s_t, a_t)$ . However, for non-identical particles, assuming that graph  $g' \in S_r(\cos(g_n))$ , has tokens  $g':\{t_1, \dots, t_n\}$ , we subtract the identical particles of  $g'$  from  $g$ , with result set  $X:\{g_{particle} - g'_{particle}\}$ . Each particle in  $X$  should exist in  $g$  but not in  $g'$ , as shown in (10).

$$(p \in X) \wedge (p \notin g'). \quad (10)$$

The next phase of the action-decision algorithm, iterates for each particle  $p \in X$ , and tries to establish, if  $\exists$

$[n, n']$ , a semantic relation as a graph path, for  $n \in X$  and  $n' \in g'$ . Please note that  $n$  and  $n'$  are the semantic vertices within a semantic graph, with the respective symbolic value of token  $t$ , concept  $c$ , or relation  $r$ . The semantic relation describes an "IS A" relation. Semantic relations in the semantic graph are hierarchically ordered. This semantic graph should in fact be a *Hasse diagram* where super-ordinates are sorted near the root and sub-ordinates are sorted near the bottom, similar to the conceptual graph literature [20]. However, that depends on the semantic authority, in this case, Word-Net [23], which does indeed sort hierarchically the vertices, as hypernyms and hyponyms.

Finding a path in the semantic graph, between  $n$  and  $n'$ , denotes that there exists some form of semantic similarity. In case a semantic relation is discovered as a path, then the action-decision mechanism in RL will use the same action that was used in state  $s_t$  when constructing the graph  $g'$ . The theoretical basis for this decision, is the principle of reusing positively rewarded experience; something known to be correct, will be re-used and assumed correct, until reinforced otherwise. However, the direction of that semantic similarity is important [23]. Super-ordinates denote a super-type, from which sub-ordinates are assumed to inherit all or some of its properties, a form of *inductive* hypothesis. The exact opposite, a path connecting super-ordinate  $n$  to subordinate  $n'$ , implies a *deductive* hypothesis (e.g., a specialization of  $n$  to  $n'$ ). As such,  $n$  may not inherit properties or attributes of  $n'$ . The actual search which tries to establish if there exists a path between  $n$  and  $n'$  is simply a *breadth-first search* (BFS).

Semantic distance  $d_{[n, n']}$  also plays an important role [23]; the further apart the semantic relation discovered, the weaker it is assumed to be. The algorithm will sort the resulting set of semantic paths discovered between  $n$  and  $n'$ , based upon the semantic distance, biased towards smaller, over longer distances (measured in steps  $t$ ). Semantic direction  $d_t$ , is described by the upwards movement towards a super-ordinate or towards the downwards movement towards a sub-ordinate. Direction towards super-ordinates is preferred over sub-ordinate direction, and is perceived to be more important [9]. Direction is accumulated, as  $+1$  for sub-ordinates and  $-1$  for super-ordinates for each step  $t$  until the search terminates. The accumulated sum, as shown in (11), is discounted by factor  $\gamma$ .

$$v[n, n'] = w_s(d_{[n, n']} + \gamma * \sum d_t). \quad (11)$$

However, Word-Net also uses a sense classification [23], related to the frequency at which a semantically hierarchical tree appears. We chose to weight the sense by using  $w_s$ , thereby further biasing the computed value  $v_{[n, n']}$ , to prefer the most frequent sense. The sense weight  $w_s$  is *min-max* normalised in the event that many senses are discovered. Therefore, the resulting path values are described by three attributes: frequency of path, distance and direction. By using those attributes, and computing a  $v_{[n, n']}$  value, we approximate the best action  $a_t$ , for the most similar state obtained by the VSM earlier. The algorithm will use an action  $a_t$  with the smallest  $v_{[n, n']}$ , as shown in (12).

$$\exists : ([n, n']) : a_t \leftarrow a_t(\min(v_{[n, n]})). \quad (12)$$

Below is the pseudo-code describing Semantic approximation. Selecting a vertex of *Concept* or *Relation* type, as shown in (5), is algorithmically implementable, as shown in (Algorithm 1). More specifically, operation *Sim* (Algo-

---

**Algorithm 1.** Token Action Decision
 

---

```

1: function DECIDE( $M, V_r, t$ ) ▷ t is a token
2:    $S_r = \text{SIM}(M, V_r)$  ▷ set of similar graphs
3:   if  $S_r = \emptyset$  then
4:      $S_r = \text{SORT}(S_r)$ 
5:     for  $g' \in S_r(\text{cos}(g_n))$  do
6:       if  $(x = s_g - g') = \emptyset$  then
7:         for  $t' \in x$  do
8:           if  $t \equiv t'$  then return  $g' : (a_t)$ 
9:           else if  $\exists v_{[n, n']}$  then
10:             $a_t : \min(v_{[n, n']})$  return  $g' : (a_t)$ 
11:           else return Null
12:         end if
13:       end for
14:     end if
15:   end for
16: end if
17: end function
    
```

---

gorithm 1, line 2) is the VSM vector output from formula (8), whereas *Sort* (Algorithm 1, line 4) is simply a descending ordering based upon *cosine*, as shown in equation (9). Please note that the BFS is not shown in (Algorithm 1). The algorithm may terminate when:

- A path between  $n$  and  $n'$  is found.
- The search is exhausted (e.g., there exists no path)

The CA commences (Algorithm 1) whilst there exist tokens not converted to vertices in the CG. As such, this is a linear operation, proportional to the tokens found in the feed  $r$ . Complexity is  $O(n^2)$ , and the algorithm describes the function  $f_n(t_r)$ , as shown in (5).

The second part of the action space, function  $f_c(g)$ , as shown in (5), is depicted in (Algorithm 2) which has to decide what *edges* to create for *Relations*. In (Algorithm 2), the complexity is at least  $O(n^3)$ , and depending on the output of the similarity function, there is no guarantee that a decision may be possible to be approximated. The minimum  $\min(v_{[n, n']})$  calculation has been omitted, albeit it should be at (Algorithm 2, line 15). Deciding for relations  $r$ , is a bit different: the algorithm not only has to establish if there exists a similarity between relations  $r$  and  $r'$ , but also discover if  $r'$  is usable by establishing if the concepts, to which  $r'$  is known to connect to, exist in our graph  $g$ . Failing to establish such a condition, the algorithm will try to find if for any of the concepts  $c'$  connected by  $r'$ , there exists a semantic relation (Algorithm 2, line 14). Therefore, the action approximation will not be based solely on the similarity between relations but is further based upon the similarity of concepts. Only in the event that the algorithm cannot provide an appropriate action  $a_t$ , a random action will be selected. In such an event, the algorithm falls back to being a semi-random MDP, where  $f_n(t_r)$  randomly selects a vertex type, and  $f_c(g)$  randomly connects relations to concepts.

---

**Algorithm 2.** Edge Action Decision
 

---

```

1: function DECIDE( $M, V_r, r$ ) ▷ r is a relation
2:    $S_r = \text{SIM}(M, V_r)$ 
3:   if  $S_r = \emptyset$  then
4:      $S_r = \text{SORT}(S_r)$ 
5:     for  $g' \in S_r(\text{cos}(g_n))$  do
6:       if  $(x = s_g - g') = \emptyset$  then
7:         for  $r' \in x$  do
8:           if  $(r' \equiv r) \vee (\exists v_{[r, r']})$  then
9:              $\text{edges}_{r'} : \{[r', c'_1], \dots, [r', c'_n]\}$ 
10:            for  $c' \in \text{edges}_{r'}$  do
11:              for  $c \in g$  do
12:                if  $c' \equiv g_c$  then
13:                  return  $g' : (a_t)$ 
14:                else if  $\exists [c, c']$  then
15:                  return  $g' : (a_t)$ 
16:                end if
17:              end for
18:            end for
19:          else return Null
20:        end if
21:      end for
22:    end if
23:  end for
24: end if
25: end function
    
```

---

## VI. DISCUSSION AND FURTHER WORK

Reusing known positively rewarded experience is a characteristic of the human psyche known as crystallized intelligence [4]. By extending RL in such a way, which is also modelled upon psychological processes [7], we hope to enable a form of RL algorithm, which can avoid random selections as often as possible, whilst emulating a cognitive function in a somewhat realistic manner. Furthermore, an expansion in experience should actually aid the action-decision, as more experience implies that there will exist a plethora of episodes and their respective graphs, upon which the algorithm may reason why an action should be taken. The proposed action-decision algorithm is an on-policy approximation, based upon logic and semantics, as shown in (13).

$$Q(s_t, a_t) \leftarrow Q(s_t(\max(\text{cos}(\Theta_g))), a_t(\min(v_{[n, n']}))). \quad (13)$$

Entirely avoiding random selections is not possible, as a cold-start issue may always ensure that at some point, a random choice will have to be made. Furthermore, semantic relations are obtained by a semantic authority, such as Word-Net [23]. Semantic relations may not be discoverable, due to being non-existent in Word-Net, as may be the case with highly specialized terms or words, in which case, the algorithm will fall back to random selections. This relates to whether the MDP is fully observable; it is assumed not to be due to the limitations of discovering semantic relations. In addition to the above, the actual size of reinforced experiences (as episodes or conceptual graphs) would not benefit from a high ratio of negative experiences, albeit those could be used to avoid negative actions when randomly deciding. We have empirically optimised the RL

algorithm in our preliminary experiments, by adjusting for a low  $\alpha$  learning rate and high  $\gamma$  discount factor. During non-RSS trial runs, convergence occurred with a thousand iterations whilst using an exploration-greedy policy. Early tests of the algorithm indicate that it appears to approximate *states* and *actions*, with an accuracy that could be related to the VSM model and Word-Net accuracy, but that remains to be evaluated. Future work will see the comparison of a plain RL (Markov e-greedy) algorithm, a probabilistic (Morphosyntactic) RL algorithm, and the Semantic-driven RL algorithm described in this paper. A double-blind experiment where human users have to project RSS feeds to CG, may also provide further psychological insight and information on the modelled process and algorithm.

## VII. CONCLUSIONS

The need for such an algorithm was discovered after early anecdotal experiments indicated that frequent random actions occurred due to infrequent Q policy re-visitation, and thus had a dramatic effect on the learning ability of the agent. The novelty is in implementing approximation methods on a purely symbolic basis. As most approximation and regression methods are based on numeric foundations [8][7], semeiotic relations governed by distributional and relational semantics, were the most important factor in dictating the design of this algorithm.

Whereas a lot of effort has been put into approximation, regression, knowledge mining, acquisition and management, it is our belief that trying to tackle such a task using a trainable and adaptive cognitive agent, presents some very interesting dilemmas and hence, can provoke interesting solutions. However, the algorithm is still rudimentary, as it does not rely on exploiting knowledge acquired by conceptual graphs, but only episodic experience. The absence of domain knowledge re-usability should not be a limiting factor for learning, but it could be a limiting factor for discovering new knowledge, or other logic operations based upon that domain knowledge. Will prior knowledge and experience aid learning? Could the agent successfully reason after the solution? It is widely accepted that an increase in the number of propositions has a dramatic impact on learning algorithms [8]. More importantly, as the state space grows, will the propositional logic deteriorate or augment? The algorithm and framework is based upon the principality that relations discovered will not drastically change over time, but even if they do, the agent should be able to adapt to those changes through learning.

## REFERENCES

- [1] M. Tenorth and M. Beetz, "KnowRob - A Knowledge Processing Infrastructure for Cognition-enabled Robots. Part 1: The KnowRob System", *International Journal of Robotics Research (IJRR)*, vol. 32, no. 5, April 2013, pp. 566-590.
- [2] X. Chen, A. Shrivastava, and A. Gupta, (retrieved:May,2014), "NEIL: Extracting Visual Knowledge from Web Data", Available (URL): <http://www.neil-kb.com>
- [3] M. Thelwall, R. Prabowo, and R. Fairclough, "Are Raw RSS Feeds Suitable for Broad Issue Scanning? A Science Concern Case Study", *Journal of the American Society for Information Sciences and Technology*, vol. 57, Issue 12, 2006, pp. 1644-1654.
- [4] R. B. Cattell, "Abilities: Their structure, growth, and action", New York, Houghton Mifflin, 1971, ISBN-10: 0395042755
- [5] C. L. Nehaniv and K. Dautenhahn, "Imitation in Animals and Artifacts", Bradford, Complex Adaptive Systems, 2002, ISBN: 9780262042031
- [6] H. Lieberman, "Your wish is my command: Programming by example", Interactive Technologies, Kaufmann, 2001, ISBN-10: 1558606882
- [7] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction", MIT press, 1998, ISBN-10: 9780262193986
- [8] P. Tadapalli, R. Givan, and K. Driessens, "Relational Reinforcement Learning: An Overview", *Proceedings of the ICML'04, Workshop on Relational Reinforcement Learning*, Ban, Canada, July 2004, pp. 1-9.
- [9] J. F. Sowa, "Knowledge Representation: Logical, Philosophical and Computational Foundations", Brooks Press, 1999, ISBN-10: 0534949657
- [10] A. Karalopoulos, M. Kokla, and M. Kavouras, "Comparing Representations of Geographic Knowledge Expressed as Conceptual Graphs", *First International Conference, GeoS 2005*, Mexico City, Mexico, Nov. 2005, pp. 1-14.
- [11] A. Strupchanska, M. Yankova, and S. Boytcheva, "Conceptual Graphs Self-tutoring System", *Proceedings of the 11th International Conference on Conceptual Structures (ICCS)*, Dresden, Germany, July 2003, pp. 323-336.
- [12] D. T. Wijaya, P. P. Talukdar, and T. M. Mitchell, "PIDGIN: Ontology Alignment using Web Text as Interlingua", *Proceedings of the Conference on Information and Knowledge Management (CIKM)*, ACM, 2013, pp. 589-598.
- [13] M. Monstes-y-Gomez, A. Gelbukh, and A. Lopez-Lopez, "Text Mining at Detail Level Using Conceptual Graphs", *10th International Conference on Conceptual Structures (ICCS)*, Borovets, Bulgaria, July 2002, pp. 122-136.
- [14] D. Rajagopal, E. Cambria, and D. Olsher, "A Graph-Based Approach to Commonsense Concept Extraction and Semantic Similarity Detection", *WWW 2013 Companion*, Rio de Janeiro, Brazil, May 2013, pp. 565-570.
- [15] T. Hong and I. Han, "Knowledge-based data mining of news information on the Internet using cognitive maps and neural networks", *Expert Systems with Applications*, vol. 23, Issue 1, July 2002, pp. 1-8.
- [16] H. Lieberman, B. A. Nardi, and D. J. Wright, "Training Agents to Recognize Text by Example", *Autonomous Agents and Multi-Agent Systems*, vol. 4, Issue 1-2, March 2001, pp. 79-92.
- [17] M. S. Pandit and S. Kalbag, "The Selection Recognition Agent: instant access to relevant information and operations", *Knowledge-Based Systems*, vol. 10, Issue 5, March 1998, pp. 305-310.
- [18] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey", *The international journal of Robotics Research*, vol. 32, no. 11, Sep. 2013, pp. 1238-1274.
- [19] A. T. Lawniczaka and B. N. Di Stefano, "Computational intelligence based architecture for cognitive agents", *International Conference on Computational Science (ICCS)*, 2010, pp. 2227-2235.
- [20] M. Chein and M. Mugnier, "Graph-based Knowledge Representation", Springer, 2009, ISBN: 978-1-84800-285-2
- [21] D. Jurafsky and J. H. Martin, "Speech and Language Processing", 2nd Edition, Prentice Hall, 2008, ISBN-10: 0131873210
- [22] H. Van Halteren, J. Zavrel, and W. Daelemans, "Improving Accuracy in Word Class Tagging through the combination of Machine Learning Systems", *Association for computational Linguistics*, 2001, pp. 200-229.
- [23] C. Fellbaum, "WordNet: An Electronic Lexical Database (Language, Speech, and Communication)", MIT press, 1998, ISBN-10: 026206197X
- [24] P. Turney and P. Pantel, "From Frequency to Meaning: Vector Space Models of Semantics", *Journal of Artificial Intelligence Research*, vol. 37, 2010, pp. 141-188.