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

Alexandros Gkiokas
University of Warwick
Department of Computer Science
Coventry, CV4 7AL
e-mail: A.Gkiokas@warwick.ac.uk

Alexandra I. Cristea
University of Warwick
Department of Computer Science
Coventry, CV4 7AL
e-mail: A.I.Cristea@warwick.ac.uk

***Abstract*—Imitative processes such as knowledge transference, have been long pursued goals of Artificial Intelligence. The significance of knowledge acquisition in animals and humans has been studied by scientists from the beginning of the 20th century. Our research focuses on acquiring information via observational imitation and agent-user interaction. The cognitive agent described here emulates a perceptive and learning system, trained for the purpose of self-augmenting its learning capabilities, in order to achieve knowledge acquisition. The agent's purpose is to learn the semiotics of Rich Site Summary feeds through empirical observation. It is also trained to autonomously represent that knowledge in a manner that is both logically sound, and computationally tractable, through the use of conceptual graphs. The main novel algorithm enabling this agent is based upon Reinforcement Learning, by approximating decisions through distributional and relational semantics.**

Keywords–Cognitive Agent; Reinforcement Learning; Conceptual Graphs; Expert Systems;

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 Rich Site Summary (RSS) feeds [3] as the source of knowledge, due to their parsimony and quality of information (as discussed in section IV). 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 storage and transfer of the modern world. In this paper, we propose the algorithmic fusion of various AI technologies as well as a new regression and approximation algorithm for Reinforcement Learning (RL), in order to enable a cognitive agent (as will be shown in section IV) to autonomously learn how to acquire knowledge. The cognitive agent emulates, but does not simulate, a perceptive process and learning system, based upon psychological studies in humans [4] and animals [5]. The cognitive agent's purpose is to acquire knowledge; for that we employed training the agent (via machine learning), rather than using conventional heuristics. Doing so, we hypothesize it makes the agent adaptive, and enables it to constantly and domain-independently augment its acquisition capabilities and expand its knowledge base.

Machine Learning (ML) literature treats an increased state space as a problem rather than an aid [6] and is usually dealt via state or action approximation. In contradiction with this belief, we attempt to show that by re-using existing reinforced experience, via a semantically driven approximation, learning can benefit from an expanding state space. Cognitive agents, learning to acquire information cannot often afford to perform random actions (e.g. Monte-Carlo search), as this leads to poor performance or slow convergence [7]. Modelled after a cognitive ability, that of imitation through observation [5], we hypothesize that the rate at which prior experience is exploited in comparison to random actions, is an accurate indicator of the performance of the proposed algorithm. By employing semantics and reasoning on semantic underlying relations between the symbolic particles of the domain, we attempt to restrict the amount of random decisions, and replace them with known actions, assumed to be correct. The purpose of this agent is to obtain RSS feeds, and then learn how to isomorphically project the feed onto a Conceptual Graph (CG) [8]. In this paper we will describe a novel algorithm that implement this process, and thereby extends RL. The remainder of the paper is organised as follows. 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 describe the architecture of the cognitive agent. Section V tackles the issue of semantic approximation, and describes the main algorithm proposed. This is followed by a discussion in section VI and conclusions (section VII).

II. Related Research

Existing research on automated Knowledge Acquisition (KA) has focused on heuristics [9], morphosyntactics [10] or statistical and probabilistic analysis, either via natural language processing or as part of the knowledge representation meta-structure creation [11]. Other researchers have used conceptual graphs [12] for mining, or semantics and morphosyntactics [13] in order to extract concepts, thus removing the limitations of heuristic approaches; and yet others have combined concept maps and neural networks such as the KBMiner [14].

In direct comparison to the existing research, we employ a machine learning algorithm, in order to train the agent for KA. The algorithm described in this paper here can be said to belong to the family of stochastic supervised learning methods. The agent does not attempt to detect patterns, regress information, or summarize text, but instead focuses only on projecting mined information, into a formal representation, 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 this purpose, we use RL, which has been widely used for a variety of Artificial Intelligence (AI) tasks, ranging from robotics [15], to games, to intelligent agents [6]. Reinforcement learning research has, however, been conscious about approximating and regressing state-actions since its inception [7], [6], 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 experience re-usability in cognitive agents learning via RL.

Research in imitation for artificial artefacts has historically [5] focused in robotics, as it stands to be the field that would mostly benefit from knowledge transference. Whilst research on knowledge transference in software agents [16] has focused on inference and prediction, our approach addresses bridging Cognitive Agents (CA) and ML, for acquiring knowledge (KA).

III. Problem Setting and Objectives

The primary objective of the proposed cognitive agent is to obtain RSS feeds and construct the correct CG from each feed. We assume each RSS to contain knowledge that can produce one or more CG [8]. Traditionally, CG creation has been assigned to knowledge engineers, human users acting as experts who construct the correct conceptual graphs. The second, more practical objective, is to enable the cognitive agent 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 believe the approach has several advantages over conventional approaches:

- The agent should be able to process any domain of symbolic information.
- The agent has an adaptive and online operation, thus it can learn even after it has been trained, it should in fact be able to keep learning indefinitely.
- The agent, after being trained, should be able to autonomously acquire knowledge without the need of a knowledge engineer or human user.
- A trained agent should be able to deal with unknown knowledge, uncertainty and adapt to nonmonotonic knowledge (through the use of RL).

IV. Cognitive Agent Architecture

According to the criteria set by Lawniczak and Di Stefano [17], the proposed agent could be partially called a cognitive agent. It satisfies four of the five criteria:

- Perceives Information (RSS) in its environment (Web) and provided by other agents (through implementation protocol)
- Reasons about that information using existing knowledge (through ML, RL and Semantics)
- Responds to other agents, human (via Web interface) or artificial (via implementation protocol)
- 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 [17]. 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 [8], which are formal structures, based upon philosophical principles, that offer logically sound and reasoning operations [18]. We 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 [12] and tractable way, in order to perform various logic and reasoning operations [18], which we do not examine, but are important and desirable nonetheless.

The agent's main task is to parse RSS feeds, and construct the correct conceptual graph g such that g correctly represents the knowledge encapsulated in the RSS 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 learning function f is a mapping or projecting function.

$$f : r \mapsto g. \tag{1}$$

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

(2) and (3).

$$f_p(r) = t_r, f_m(t_r) = g_r. \tag{2}$$

$$(2) \therefore (f_m \circ f_p) = f_m(f_p(r)). \tag{3}$$

Those two functions compose the actual operation of the CA: first the feed r is parsed as a set of tokens t_r (2), and then that set is isomorphically projected onto graph g (3). 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 (4).

$$f_m : r \cong g \Leftrightarrow Q(s_t, a_t). \tag{4}$$

Where (4) Q is the Markov decision, calculated as a policy for a given state s_t when taking action a_t, at time-step t. Function f_p parses the feed r, into the set of tokens t_r:$\{\mathrm{t}_1,..,\mathrm{t}_n\}$ by tokenizing it. Parsing and tokenizing, is a Natural Language Processing (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 [19]. 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 [19], amongst others, this process concatenates words that are encapsulated by quotes (single or double) into a single token. Using morphosyntactic (part-of-speech) tagging [20], 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 cognitive agent, as it has no other way of inferring those tokens. The tokenizing should capture both the ontology and relations encapsulated in 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 Markov Decision Process (MDP) in the form of an episode of RL:

- In function f_n tokens have to be converted to a vertex (Concept or Relation) [8].
- In function f_c relations have to connect to concepts with an edge [8].

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 [6]. 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 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 [6], 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

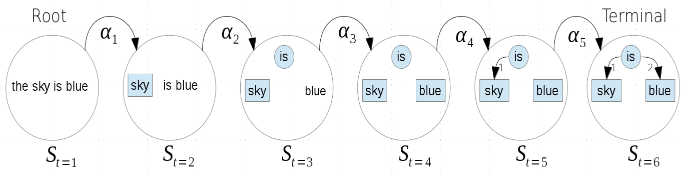

Figure 1: MDP as a RL State-Path

partially observed MDP [6]. The *Q-Policy* value function is calculated by the S.A.R.S.A algorithm [6], as shown in (5).

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]. \tag{5}$$

The algorithm back-propagates that reward R_{t+1} for terminal states, when re-iterating previous states, using a constant discount factor gamma. We chose SARSA over Q-learning, as it is assumed to be more greedy with respect to immediate rewards [6]. The actual action a_t in (5), has the binary nature describe by (figure 1) and as such can logically be represented as shown in (6).

$$a_t \leftarrow f_n(t_r) \vee f_c(g). \tag{6}$$

The logical 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 $\mathrm{S}_p\mathrm{a}_t$, as shown in (7), where n is the number of all vertices in g.

$$S_p a_t(f_n) = 2^n. \tag{7}$$

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 (8).

$$S_p a_t(f_c) = \int_1^c (r * x_r). \tag{8}$$

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 [8].

The CA is complemented by a semantic graph S, represented internally by a Word-Net instance [21]. Contrary to CG literature [18] 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 at 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 $\mathrm{S}_p\mathrm{a}_t$ 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 [6]. Conventionally, RL is a semi-random MDP, often employing a Monte-Carlo search. However, due to the importance of semantics in symbolic domains [21] 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 action similarity inherent between the particles that describe a state is obtained from the lexicon of the domain for which knowledge is acquired. Those semantic relations are exploited similar to a logic programming fashion, so as to infer known correct action-decisions. 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 (9), by using a Vector Space Model (VSM) [22].

$$s = \frac{M * V_r}{\|M\| * \|V_r\|}. \tag{9}$$

M (9) is the *sparse* matrix of token-frequencies, and is created by indexing each unique CG as a column, and each token used as a vertex in the CG as a row. V_r is the (respective to matrix M) vector of tokens representing the feed r. The VSM discovers similarity based upon distributional semantics, and returns a vector s of degrees of similarity, as shown in (10), representing how similar is each CG in memory, to the input vector V_r.

$$s = [cos(\Theta_1), ..., cos(\Theta_n)]. \tag{10}$$

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, has tokens g':{t_1,..,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 (11).

$$(p \in X) \wedge (p \notin g'). \tag{11}$$

The next phase of the action-decision algorithm, iterates for each particle p ∈ X, and tries to establish, if ∃ *[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 *S*, 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 [18]. However, that depends on the semantic authority, in this case, Word-Net [21], which does indeed sort hierarchically the vertices, as hypernyms and hyponyms.

Finding a path in the semantic graph *S*, 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 [21]. 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.*

Semantic distance $d_{[n,n']}$ also plays an important role [21], 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 [8]. 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 (12), is discounted by a constant *gamma.*

$$v[n,n'] = w_s(d_{[n,n']} + \gamma * \sum d_t). \tag{12}$$

However, Word-Net also uses a sense classification [21], which is 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-man* 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 use 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 before. The algorithm will use an action a_t with the smallest $v_{[n,n']}$, as shown in (13).

$$\exists : ([n,n']) \therefore a_t \leftarrow a_t(min(v_{[n,n']})). \tag{13}$$

The actual test framework is developed using the C++11 standard as approved by ISO on 12 August 2011.

However, for simplicity, we display the core algorithm implementation using pseudo-code. Selecting a vertex of *Concept* or *Relation* type, as shown in (6), is algorithmically implementable as shown in (Algorithm 1).

Algorithm 1 Token Action Decision

```
function DECIDE($M$,$V_r$, $t$)                         ▷ t is a token
    $g$ = SIM( $M$, $V_r$)
    if $g\neg\emptyset$ then
        $s$ = SORT( $s$)
        for $g' \in s$ do
            if $(x = s_g - g')\neg\emptyset$ then
                for $t' \in x$ do
                    if $t \equiv t'$ then return $g' : (a_t)$
                    else if $\exists v_{[n,n']}$ then
                        $a_t : min(v_{[n,n']})$ return $g' : (a_t)$
                    else return Null
                    end if
                end for
            end if
        end for
    end if
end function
```

More specifically, operation *Sim* (line 2) is the VSM vector output from formula (9), whereas *Sort* (line 4) is simply a descending ordering based upon *cosine* as shown in formula (10). The algorithm may terminate when:

- A path between *n* and *n'* is found (please note that the search is not shown in Algorithm 1).
- The search is exhausted (e.g., there exists no path)
- A pre-set distance has been traversed and therefore the algorithm stops.

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 (6).

The second part of the action space, function $f_c(g)$, as shown in (6), 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 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 (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

```
function DECIDE($M$,$V_r$, $r$)                         ▷ r is a relation
    $g$ = SIM( $M$, $V_r$)
    if $g\neg\emptyset$ then
        $s$ = SORT( $s$)
        for $g' \in s$ do
            if $(x = s_g - g')\neg\emptyset$ then
                for $r' \in x$ do
                    if $(r' \equiv r) \vee (\exists v_{[r,r']})$ then
                        $edges_{r'} : \{[r', c'_1], .., [r', c'_n]\}$
                        for $c' \in edges_{r'}$ do
                            for $c \in g$ do
                                if $c' \equiv g_c$ then
                                    return $g' : (a_t)$
                                else if $\exists [c, c']$ then
                                    return $g' : (a_t)$
                                end if
                            end for
                        end for
                    else return Null
                    end if
                end for
            end if
        end for
    end if
end function
```

VI. Discussion & Further Work

Reusing known positively rewarded experience is a characteristic of the human psyche known as crystallized intelligence [4]. By extending reinforcement learning in such a way, which is also modelled upon psychological processes [6], we hope to enable a form of reinforcement learning 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 (14).

$$Q(s_t, a_t) \leftarrow Q(s_t(max(cos(\Theta_g))), a_t(min(v_{[n,n']}))). \quad (14)$$

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 [21]. 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 few hundred iterations whilst using an exploration-greedy policy. Early trials of the algorithm indicate that it does indeed approximate *states* and *actions* using the aforementioned models and formulas, with an accuracy that is directly related to the VSM model accuracy and influenced by Word-Net results. 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.

VII. Conclusions

The need for such an algorithm was discovered after early anecdotal experiments indicated that frequent random action-decision due to infrequent Q policy revisitation, had an 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 [7] [6], 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. Although, the algorithm is still rudimentary, as it does not rely on exploiting knowledge acquired by conceptual graphs, but only experience, it could still provide an interesting reference point. 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 [7]. 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, "M. Beetz, KnowRob - A Knowledge Processing Infrastructure for Cognition-enabled Robots. Part 1: The KnowRob System", International Journal of Robotics Research (IJRR), April 2013, vol 32 number 5, pp: 566-590

[2] X. Chen, A. Shrivastava, A. Gupta, "NEIL: Extracting Visual Knowledge from Web Data" 2013, unpublished, Carnegie Mellon University, Available: `http://www.neil-kb.com`.(URL)

[3] M. Thelwall, R. Prabowo, 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, 2006, vol 57, Issue 12 pp : 1644 – 1654

[4] R. B. Cattell, "Abilities: Their structure, growth, and action", Houghton Mifflin, ISBN-10: 0395042755, New York, 1971

[5] C. L. Nehaniv, K. Dautenhahn, "Imitation in Animals and Artifacts", Complex Adaptive Systems, Bradford, ISBN: 9780262042031, 2002

[6] R. S. Sutton, A. G. Barto, "Reinforcement Learning: An Introduction", MIT press, ISBN-10: 9780262193986, 1998

[7] P. Tadapalli, R. Givan, K. Driessens, "Relational Reinforcement Learning: An Overview", Proceedings of the ICML'04 workshop on Relational Reinforcement Learning, Ban, Canada, 2004

[8] J. F. Sowa, "Knowledge Representation: Logical, Philosophical and Computational Foundations", ISBN-10: 0534949657, Brooks Press, 1999

[9] A. Karalopoulos, M. Kokla, M. Kavouras, "Geographic Knowledge Representation, using Conceptual Graphs", 7th AGILE Conference on Geographic Information Science, 2004

[10] A. Strupchanska, M. Yankova, S. Boytcheva, "Conceptual Graphs Self-tutoring System", Proceedings of the 11th International Conference on Conceptual Structures, ICCS 2003, Dresden, Germany, July 21-25, 2003, pp: 323-336

[11] 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), 2013

[12] M.Monstes-y-Gomez, A.Gelbukh, A.Lopez-Lopez, "Text Mining at Detail Level Using Conceptual Graphs", 10th International Conference on Conceptual Structures, ICCS 2002 Borovets, Bulgaria, July 15–19, 2002, pp: 122-136

[13] D. Rajagopal, E. Cambria, D. Olsher, "A Graph-Based Approach to Commonsense Concept Extraction and Semantic Similarity Detection", WWW 2013 Companion, Rio de Janeiro, Brazil, May 13–17, 2013, pp: 565-570

[14] T. Hong, I. Han, "Knowledge-based data mining of news information on the Internet using cognitive maps and neural networks", Expert Systems with Applications, July 2002, vol 23, Issue 1, pp 1–8

[15] J. Kober, J. A. Bagnell, J. Peters, "Reinforcement learning in robotics: A survey", The international journal of Robotics Research, September 2013, vol 32 no 11, pp: 1238-1274

[16] H. Lieberman, "Your wish is my command: Programming by example", Interactive Technologies, Kaufmann, ISBN-10: 1558606882, 2001

[17] A. T. Lawniczaka, B. N. Di Stefanob, "Computational intelligence based architecture for cognitive agents", International Conference on Computational Science, ICCS 2010, pp: 2227–2235

[18] M. Chein, M. Mugnier, "Graph-based Knowledge Representation", Springer, ISBN: 978-1-84800-285-2, 2009

[19] D. Jurafsky, J. H. Martin, "Speech and Language Processing", 2nd Edition, Prentice Hall, ISBN-10: 0131873210, 2008

[20] H. Van Halteren, J. Zavrel, W. Daelemans, "Improving Accuracy in Word Class Tagging through the combination of Machine Learning Systems", Association for computational Linguistics, 2001, pp: 200 -229

[21] C. Fellbaum, "WordNet: An Electronic Lexical Database (Language, Speech, and Communication)", MIT press, ISBN-10: 026206197X, 1998

[22] P. Turney, P. Pantel, "From Frequency to Meaning: Vector Space Models of Semantics", Journal of Artificial Intelligence Research, 2010, vol 37, pp: 141-188