Unsupervised Neural Controller for Reinforcement Learning Action-Selection: Learning To Represent Knowledge

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*—Constructing the correct Conceptual Graph representing some textual information, requires a series of decisions, such as vertex and edge creation. The process of creating Conceptual Graphs, involves ontologies, semantic relations and similarities, as well as structural relativity and isomorphic projection, all described as decisions of a learning agent or system. The actual process is known as Semantic parsing, yet the novel fusion of Reinforcement Learning (RL) and Restricted Boltzmann Machines (RBM), learns how to perform this complex task by demonstration from a human user. Herein we showcase the design of such a system in a theoretical manner, in order to establish the background mechanisms which will learn how to parse information and correctly project it onto Conceptual Graphs.**

Keywords–Deep Learning; Restricted Boltzmann Machines; Reinforcement Learning; Conceptual Graphs; Complex Systems; Programming by Example.

I. INTRODUCTION

Parsing a piece of information written in text, and then constructing a Conceptual Graph[2], correctly representing that information, is a hard and tedious task. The norm has been to use statistics, heuristics or other algorithm-based approaches. What we propose hereinafter, is to construct an agent that is able to learn how to do that task, simply by being shown how we humans would construct certain graphs when given a small sentence. In that sense, our approach is a black box, which utilises neural-based models: Reinforcement Learning[1], and Restricted Boltzmann Machines[3]. The black box's input is the sparsely encoded information underlying a Markov State, and the output (obtained by the reservoir) is a decision or action. This binary system, relies on State-Action-Reward-State-Action (SARSA) algorithm[1], whereas the decision making uses an RBM, to approximate correct actions given past training experience.

The system, is taught by being given human-generated tuples of text and corresponding conceptual graphs. It learns by reversely decomposing conceptual graphs into Markov Decision Processes (MDP), and recoding statistical information. When queried, the expected behaviour is to reuse previously learnt operations, and perform similar projections. The training data set consists of a sentence s which is projected onto a conceptual graph g. We use conceptual graphs, as they offer a minimalistic, logically sound and structurally simple way to represent knowledge, visually, mathematically and programmatically [4],[5].

II. RELATED RESEARCH

The interdisciplinary nature of such research makes it impractical to mention all and every research carried out, thereby we focus only on the most relevant recent research. *Deep Learning* has seen an increased amount of research in Restricted Boltzmann Machines, most notably from Yann LeCun[6], Geoffrey Hinton[7], and Jürgen Schmidhuber[8]. A variety of Deep models, Convolutional Neural Networks and auto-encoders has seen a wide interest from researchers from all around the globe. Our choice of using a somewhat simplistic RBM is founded on the work of Geoffrey Hinton, and the demonstrated ease of use and fast performance of RBM.

Conceptual Graphs have been thoroughly researched by their creator Jown Sowa[2], Chein and Mugnier[9], as well as Madaleina Croitoru[10]. Their usage has normally been in common logic, predicate logic and reasoning, but some newer research (mostly from Croitoru and INRIA) has seen their usage for knowledge storing and manipulation[5][10].

The most relevant research, has been in *Semantic Parsing* as it is the field which addresses the task of our research. Whilst a variety of methods and techniques have been used to achieve the same goal, most notable is the work from Standford's NLP group[11], Poon and Domingos's unsupervised parsing, and the work carried out by University of Texas, Machine Learning group[13].

III. REINFORCEMENT LEARNING

The agent's task, is to learn to construct a conceptual graph g from a sentence s. The actual operation of constructing a conceptual graph, is learnt by the agent, by using Reinforcement learning. Reinforcement learning has three added benefits, other than *being a well tested and well proven* learning architecture:

1) It is based on behaviouristic psychology, where an interactive reinforcement by a user (or a self-reward) enables learning.
2) It can learn online, offline, using batches and has seamless training and learning phases.
3) It employs Markov Decision Processes, in combination with "States" which can describe Symbolic information is a very natural way.

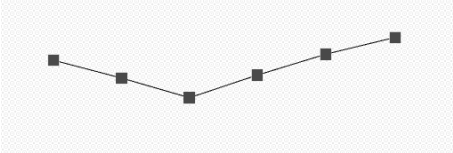
4) It is considered a neuro-computation algorithm, yet it has many clearly defined learning algorithms, from which I can chose or swap at any time.

The basis of Reinforcement Learning, is the State,Action pair, which, when using episodic Reinforcement Learning, a sequence of State,Actions make up an episode, as shown in (1).

$$\vec{e}^{t} : \{(s_1, a_1), (s_2, a_2), \cdots, (s_t, a_t)\} \tag{1}$$

This form of chain, uses a state to describe the temporary memory or condition of the agent, for time-step *t*. States, are joined together by actions, and a state may lead to one or many other states, depending on the actions. Below in (Figure 1), a single episode is shown, which is made up of 6 states and 5 actions.

Figure 1. An episode.



There exists a root node, a terminal node, and states linked with actions. A state, *is represented by the text of a sentence and by a Conceptual Graph*. As the agent processes a sentence, words are removed, and are converted into vertices of the Conceptual Graph. Once no more words are left, edges are created between the vertices of the Conceptual Graph. This two-phased construction, is essentially the semantic parsing that the agent learns via demonstration.

Therefore, a state is described as shown in (2).

$$s_t : \left\{ \begin{array}{l} T : \{w_1, w_2, \cdots, w_n\} \\ G : \{V, E, O\} \end{array} \right\}. \tag{2}$$

Token set *T*, is a list of tokenised words from the original sentence, and each time a word is converted into a vertex for graph *G*, it is removed from that list.

Conceptual Graph *G*, is a per the Conceptual Graph literature, a directed graph. However, the graph has two types of vertices: **Concepts** and **Relations**.

$$V_{G_t} : \{C, R\}. \tag{3}$$

Thus, we describe a conceptual graph, as shown in (4).

$$G_t : \left\{ \begin{array}{l} C : \{c_1, c_2, \cdots, c_n\} \\ R : \{r_1, r_2, \cdots, r_n\} \\ E : \{\{r_1, c1\}, \{r_1, c_2\}, \cdots, \{r_n, c_n\}\} \end{array} \right\}. \tag{4}$$

Finally, the **order** in which edges are connected is also important, and is denoted by the order of edge appearance.

The agent learns how to construct the graph *G* by learning the optimal Policy *Q* for all possible State-Action pairs s_t, a_t. The *Q-Policy* value is calculated by the SARSA algorithm [1], as shown in (5).

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

The algorithm back-propagates that reward R_{t+1} for terminal states, when re-iterating previous states, using a discount factor γ and learning rate α. We chose SARSA over Q-learning, as it is assumed to be more greedy with respect to immediate rewards [1].

An action, quite contrary to the state, is a decision, and in the conceptual graph context, its a decision to alter the graph by inserting a new vertex or a new edge. Whereas state learning is a matter of approximation (via semantics mostly) and has been the focus of our previous research, actions can only be approximated via statistics, probabilities or machine learning.

We presume actions to be non-linear, multidimensionally related and governed by semantics. Statistical information about actions is recorded during training, whilst probabilities are inferred directly from statistical information, by observing how often an action is correct or erroneous. Therefore, during the agent's training, an observed paradigm where a sentence is translated into a graph, the demonstrated actions and states are recorded, and assumed correct as they are given by a human expert. When querying the agent, a different process is invoked, which has access to all that information.

IV. Semantics

Semantics governing the tokens of a sentence given as input to the agent, are important, as they can express similarity between previous states and current ones. Using WordNet[15], we acquire the super-ordinate graph of node *n* and *n'* (known as hypernyms). A node in wordnet, denotes a word in the English dictionary. Establishing a semantic path from one another, is done by iterating super-ordinates and their layers. Finding a direct path from *[n,n']* might not be possible always, in which case, the algorithm uses a breadth-first search, by finding if nodes *n* and *n'* share a common super-ordinate. Searching in a semantic super-ordinate (hypernym) graph is usually very fast. Previous anecdotal evidence [15][4], indicates that WordNet senses (returned as hypernym graphs), are narrow and long directed tree-graphs, suggesting that there is a better probability of finding a path between nodes, when searching breadth-first, with little impact on performance when compared with a depth-first search.

For the first common super-ordinate obtained, semantic distance, expressed by value $\delta_{[n,n']}$ is calculated, as shown in (6).

$$\delta_{[n,n']} = w_{s_i}\left(\sum tr_{[n,n']} + \gamma\big(dr_{[n,n']} * \sum tr_{[n,n']}\big)\right). \tag{6}$$

Constant discount factor γ can be adjusted, and is simply a way to alleviate the importance of *direction* $dr_{[n,n']}$, e.g., the traversal direction in the semantic graph, and the *semantic distance* accumulation $tr_{[n,n']}$.

Direction is represented by a -1 when iterating towards super-ordinates, as +1 when iterating towards sub-ordinates or same layer nodes.

The min-max normalised sense weight w_{s_i} biases towards more frequent senses, which always appear first, as they are sorted in WordNet from most frequent to less frequent, as shown in (7).

$$w_{s_i} = \frac{s_i - s_{min}}{s_{max} - s_{min}}. \tag{7}$$

The weight, is assigned to a specific wordnet sense *i* from the total of all senses, and affects the semantic distance.

When searching for common super-ordinates for *n* and *n'*, both values are summed, as shown in (8), where *c* is a common super-ordinate.

$$\delta_{[n,n']} = |\delta_{[n,c]} + \delta_{[n',c]}|. \tag{8}$$

V. Action Decision

At the very core of Reinforcement Learning, lies the notion that an action is performed in order to arrive to the next state. Regardless of episodic or continuous learning, deciding on an action is crucial. In traditional practises, RL implementation uses random decision making. The premise that learning will converge after an amount of exploring various policies, relies on the availability of a fitness function which will reward an episode, hence, the agent.

However, that is simply not possible when dealing with human users. A user will not go through millions of action-decisions which are randomly made, and thus appear to make no sense. As such, the agent has to iterate an episode and arrive at a correct conceptual graph, as often as possible.

Whilst we have separated the training phase from the testing phase, even during testing, we do not wish for the algorithm to iterate the entire space of possible actions before arriving onto the correct solution.

An action in this context, is the decision to convert a token into a node, or the decision to create an edge between two nodes (a relation and a concept).

VI. Action Search Space

Deciding on an action can be a time consuming process, on which the correct or erroneous generation depends upon. Since the human users will most certainly not provide feedback for each possible decision made randomly by SARSA, another fitness function is required. That fitness function, when given as input a possible action, will have to decide if the agent should go ahead with it or not.

The action search space can vary, and is denoted by $S_p a_t$, as shown in (9), where *n* is the number of all vertices in a conceptual graph *G*.

$$S_p a_t(f_n) = 2^n. \tag{9}$$

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 sentence, 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 (10).

$$S_p a_t(f_c) = \int_1^c (r * x_r). \tag{10}$$

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 [2].

It should be evident, that performing all possible actions is inconvenient, computationally expensive and time consuming.

VII. Neural-Based Decision Making

The usage of a Restricted Boltzmann Machine, comes into play when deciding which actions should be preferred over the entire space of possible actions. In a sense, the RBM acts as a predictor, in order to better estimate which action should be chosen. The input for the RBM is the *sparsely encoded* information that is attributed to an action, rather than the action itself which is highly symbolic. That information is:

- The probability that a Token is a Concept or Relation (calculated from previous statistics).
- The probability that the Token's morphosyntactic tag is a Concept or Relation.
- The similarity of a known Token to an unknown Token.

In the event that there exists no information recorded about a specific Token and the possible decisions related to it, the agent may opt to find the most "semantically" similar one, by using WordNet. It that operation takes place, then the input to the RBM will be the previous information, as well as the weight δ as shown earlier in (8). In the event that the information about a Token is already recorded, the weight would be *1*.

Sparsely encoding such information would require a minimum of 3 input neurons to the RBM. The statistics for the probabilities are recorded during training, when the agent decomposes paradigm graphs into MDP and translates them into reinforced episodes. The morphosyntactic tags are *Brown corpus Part-Of-Speech Tags* obtained by a *Lookahead POS tagger*[16].

The RBM's *n hidden* layer should be assigned empirically and through trial-and-error, but for *Token to Node* conversion we do not pressume it will be very large, as the possible action space should not be very large either, as shown in (9).

Edge creation, as an action decision, is quite different from Token to Node conversion. The requirement is that a second RBM will be used, which will decide only on Edge creation, whereas the first RBM will decide only on Node creation. The information encoded and given as input to the RBM that will decide on edges, is:

- The probability of a Relation connecting to a Concept.

- The probability of a specific morphosyntactic tag, having an edge to another morphosyntactic tag.
- The similarity of a known Relation (with its respective Token) to a currently unknown Relation.

In the event that a possible action encapsulates a Relation with a Token that has not been recorded before, and for which no statistical information exists, the agent will find the most similar Relation in memory using a derivation of the formula (8), as shown in (11).

$$\epsilon[r_i, c_i] = \frac{\delta_{[r_i,r_k]} + \left(\sum \delta_{[c_i,c_k]} \right)}{e[_{r_k,r_i}]}. \quad (11)$$

The *isomorphic similarity* is expressed by the similarity of relations r_i, r_k, as well as the total similarity of available concepts in their respective graphs, and is expressed by $\sum \delta_{[c_i,c_k]}$. What that formula (11 describes, is the availability of concepts to which a relation may connect to, in combination with how similar the concepts and relations are. A value of *1* is used when both concepts and relations have been recorded during training, and there exist statistical information through which the probabilities given as input should not be affected by non-identical token values.

VIII. Unsupervised Training via Demonstration

Training of the reinforcement learning algorithm is supervised. It is done in a single batch, where the user provides a data set of sentences and corresponding conceptual graphs. However, training of the RBM is unsupervised, as the user has no control over it. The agent, self-controls the training of the RBM, once sufficient information has been logged, indirectly, by decomposing conceptual graphs into episodes. We presume the RBM to be trained at the end of the RL training session, as at that point, enough information about actions and probabilities, will exist in the agent's memory.

We use Hinton's simple RBM training technique[14], as shown in (12).

$$E(v,h) = -\sum_{i \in visible} a_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{i,j} v_i h_j w_{ij}. \quad (12)$$

The Hopfield configuration (v,h), where v_i, h_i are the states of visible unit *i* and hidden unit *j*, a_i, b_j are their biases and w_{ij} is the weight between them.

The network assigns a probability using the energy function (13).

$$p(v,h) = \frac{1}{Z} e^{-E(v,h)}. \quad (13)$$

Where *Z* is calculated by summing all visible and hidden pairs (14).

$$Z = \sum_{v,h} e^{-E(v,h)}. \quad (14)$$

The learning rule[14], when simplified, is given by (15).

$$\Delta w_{ij} = \epsilon\big(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}\big). \quad (15)$$

Where ϵ is the learning rate. The network is assumed to be initialised in a random state, and using Gibbs sampling, whereas updating uses Contrastive Divergence[7][14].

IX. Discussion

We have demonstrated that an intelligent agent, can be taught how to semantically parse, with minimal heuristics and algorithms. Our primary focus has been to create an adaptive and dynamic agent, which does not rely on hardcoded heuristics or other types of algorithms, but rather learns in a interactive manner, how to correctly represent knowledge onto conceptual graphs, through examples.

The binary fusion of Reinforcement learning and Restricted Boltzmann machines, creates a novel model, which can self-control its action making policies, without having to re-iterate episodes numerous times in order for it to converge on a policy.

The agent is based upon proven and well performing models of neural-based artificial intelligence technologies. Both RL and RBM have been used for decades, and known to work, whereas the newly emerging Deep Learning subfield of artificial intelligence has proven how well RBM can perform when using massively parallel platforms, such as CUDA implementations.

This combination, should provide high performing results, when those two models are appropriately paired.

[TODO: Alexandra, would you like to add something here? I'm running out of stuff to write]

X. Conclusions

[TODO: Alexandra, would you like to add something here? I'm running out of stuff to write]

References

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

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

[3] H. Larochelle and Y. Bengio, "Classification using discriminative restricted Boltzmann machines", Proceedings of the 25th international conference on Machine learning - ICML '08, 2008, pp. 536.

[4] A. Gkiokas and A. I. Cristea, "Training a Cognitive Agent to Acquire and Represent Knowledge from RSS feeds onto Conceptual Graphs", IARIA COGNITIVE 2014, Venice, Italy, 25 May 2014, pp 184 - 194.

[5] 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.

[6] Y. LeCun, "Backpropagation Applied to Handwritten Zip Code Recognition", Neural Computation, 1, 1989, pp. 541–551.

[7] G. E. Hinton., "Learning multiple layers of representation", Trends in Cognitive Sciences, vol. 11, 2007, pp. 428–434.

[8] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory", Neural Computation, vol. 9, 1997, pp. 1735–1780.

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

[10] M. Croitoru, N. Oren, S. Miles, and M. Luck, “Graphical Norms via Conceptual Graphs”, Knowledge-Based Systems, Vol. 29, May 2012, pp. 31-43.

[11] D. Gildea and D. Jurafsky, “Automatic Labeling of Semantic Roles”, Computational Linguistics, Vol. 28, 2002, pp. 245-288.

[12] H. Poon and P. Domingos, “Unsupervised Semantic Parsing”, Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Vol. 1, 2009, pp. 1-10.

[13] I. Beltagy, K. Erk and R. Mooney, “Semantic Parsing using Distributional Semantics and Probabilistic Logic”, Proceedings of ACL 2014 Workshop on Semantic Parsing, June 2014, pp. 7-11.

[14] G. Hinton, “A Practical Guide to Training Restricted Boltzmann Machines”, “Neural Networks: Tricks of the Trade”, Second Edition, Lecture Notes in Computer Science, Springer, 2014, pp. 599-619, ISBN: 978-3-642-35289-8.

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

[16] Y. Tsuruoka, Y. Miyao, and J. Kazama, “Learning with Lookahead: Can History-based Models Rival Globally Optimized Models?”, Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL), 2011, pp. 238-246.