

University of Bristol
MSc in Machine Learning and Data Mining

**Reinforcement Learning Directed Search Agent
on Semantic Web**

16/05/2003
MSc Interim Report

Student: Hoi Ian Vong (Stanley) / email: hv2625@bristol.ac.uk
Supervisor: Dr Peter Flach / email: Peter.Flach@bristol.ac.uk

Table of Contents

1	ABSTRACTS	4
2	INTRODUCTION AND CONTEXT	4
3	REINFORCEMENT LEARNING	7
	3.1 STATE, ACTION AND REWARD	7
	3.2 VALUE OF A POLICY	8
	3.3 Q LEARNING	9
	3.4 APPLICATIONS.....	10
4	THE SEMANTIC WEB.....	12
	4.1 WHAT IS SEMANTICS?	12
	4.2 RDF AS METADATA.....	13
	4.3 SEMANTIC WEB AND DATA RETRIEVAL.....	15
	4.4 WHAT ELSE CAN BE DONE?	16
	4.5 REASONING IN SEARCH ENGINE.....	17
5	TIME PLAN.....	20
6	AIMS, OBJECTIVES AND MOTIVATION	22
7	INITIAL DESIGN	23
	7.1 TECHNIQUES AND TOOLS	23
	7.1.1 Java.....	23
	7.1.2 XML Parser	23
	7.1.3 Database and Serialization.....	24
	7.1.4 Threads	24
	7.2 AGENTS.....	24
	7.2.1 Searching Agent.....	24
	7.2.2 Filtering Agent.....	24
	7.2.3 Learning Agent	25
	7.3 CHALLENGE, POTENTIAL PROBLEM AND LIMITATION.....	25
8	REFERENCES & RESOURCES	26

Part I

Background Review

1 Abstracts

The current Web was designed as a huge information space, with the goal that it should not only be useful for human to human communication, but also that machines would be able to participate and help. One of the major obstacles to this has been the fact that most information on the Web is generated and designed for human consumption. The next generation of the Web, characterized as the “Semantic Web”, proposed by Tim Berners-Lee is now commonly discussed among groups with the potential capability to enhance the current Web into machine-understandable format. The objective of this project is to apply Reinforcement Learning, a dynamic machine learning algorithm into search agent to crawl across the Semantic Web, so that the agent can capture and learn from user’s interest, analyze the RDF metadata collected from Semantic Web and return web pages that may be most relevant to the user interest through an optimized path.

2 Introduction and Context

Since the explosion of Internet technology back in the early 1990’s, there have been uncountable number of web pages created, all these web pages facilitate information searching and gathering for human interest. By locating a web search engine and put in the phrase or key words we are looking for, we always receive huge number of web pages as feedback.

So far different kinds of learning algorithms have been adopted into the search process. One of the popular selections is the implementation of reinforcement learning – a dynamic machine learning algorithm that usually involves agents to maximize the reward it receives when interacting with a complex, uncertain environment. It has been proved that by integrating reinforcement learning into the searching algorithm the entire process can be enhanced in the sense of reducing the time by avoiding irrelevant but correctly includes relevant pages. Some examples of applying reinforcement learning in the web searching include:

- ◆ WAIR (Web agents for Information Retrieval) [3] is a profile based web crawling platform for personalized information services on the Web. There are mainly three agents in the platform: a user interface agent keeps track of user behavior, a retrieval agent to construct query depends on the user profile to retrieve web pages; and a filtering agent to evaluate the relevance of web pages and present a particular number of pages to the user.
- ◆ WebWatcher [4] is a browsing assistant that use reinforcement learning to determine the relevance of a document by the words in a hyperlink, although this gives a limitation on the search space is restricted to the current web page only.

In most cases, the task of searching the Web is performed by multi-agents, and the entire process is referred as web spidering. By using multi-agents, the search process can be performed in a more biological sense as these agents can work cooperatively instead of individually, that is, usually agents should be able to communicate with each other and share the knowledge they have obtained so far. Technically it is also beneficial to have multiple agents crawl across the Web in pulling the web pages as normally the bandwidth of the internet connection is not being utilized by a single thread. Examples of application adopting multi-agents are:

- ◆ ARACHNID [1] is a system that uses a collection of competitive agents for finding information on the Web. Each agent competes for limited computational resources, procreating and mutating proportionally to its success in finding relevant documents.

- ◆ Cho et al. [1] introduces a heuristic metric, PageRank, for valuing a Web based on its linkage properties. PageRank is effective spidering metric for locating pages with high PageRank or back link counts but poor in locating pages at a particular topic.
- ◆ GTE'S ILS [8] system integrates learning agents by a central controller through which the agents critique each other's proposals.
- ◆ Web search engines including AltaVista and HotBot.

However, there are two limitations with the search process mentioned above: the result of the search space is restricted to the existence of the search phrase within the web pages, and target for the entire resource discovery process is mainly focused for human consumption only. Although machine and application have been written to crawl across the Web of mark up pages to identify relevant resources, there is no practical way for the machine to relate the phrase and key words input by the user to something else. Even if the content of a web page is derived from a database with well defined meaning for its columns, the structure of the data is not evident to a robot browsing the Web. Such a limitation brings the necessity of the commonly discussed topic, the Semantic Web. Tim Berners-Lee presents Semantic Web as a vision of computing devices and software agents not only can find information more accurately than present, but also applying reasoning to that content to support human decision. The definition of Semantic Web can be found at W3C is as follow:

The Semantic Web is the representation of data on the World Wide Web. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming. [19]

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation." [11]

One of the key importance that Semantic Web emphasis on is a standard definition in metadata, or in other words, knowledge representation in a structured way. By coming into agreement in terms of structured metadata, agents and web user will then be able to have a same language in sharing idea and knowledge. Such a distribution of information sharing will enhance the current web from mainly human oriented consumption to machine consumption, and by applying reasoning to the content of the metadata, the entire search space for the agents is extended, and so will be the result of the search process. To achieve this, we need some mechanism to formalize the standardization of the metadata (grammar of the metadata). Resource Description Framework (RDF) is the approach suggested by W3C:

Resource Description Framework (RDF) is a foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web. RDF emphasizes facilities to enable automated processing of Web resources. [20]

An example application of adopting the benefits of Semantic Web is the RCal [14] (Retsina Calendar Agent). RCal is a distributed meeting scheduling agent, it takes the strength of RDF in shifting unstructured data into structured knowledge and can navigate the Semantic Web content to gather and reason about event and schedules.

Considering XML and Uniform Resource Identifier (URI) as the base of Semantic Web, the Web will be revolved with totally new and extensive power. For example, relationship among terms can be specified, different ontology can be included and understood by machine, and proof of logic and rules can be derived. A talk given by Tim Berners-Lee showed the benefits that can be achieved from:

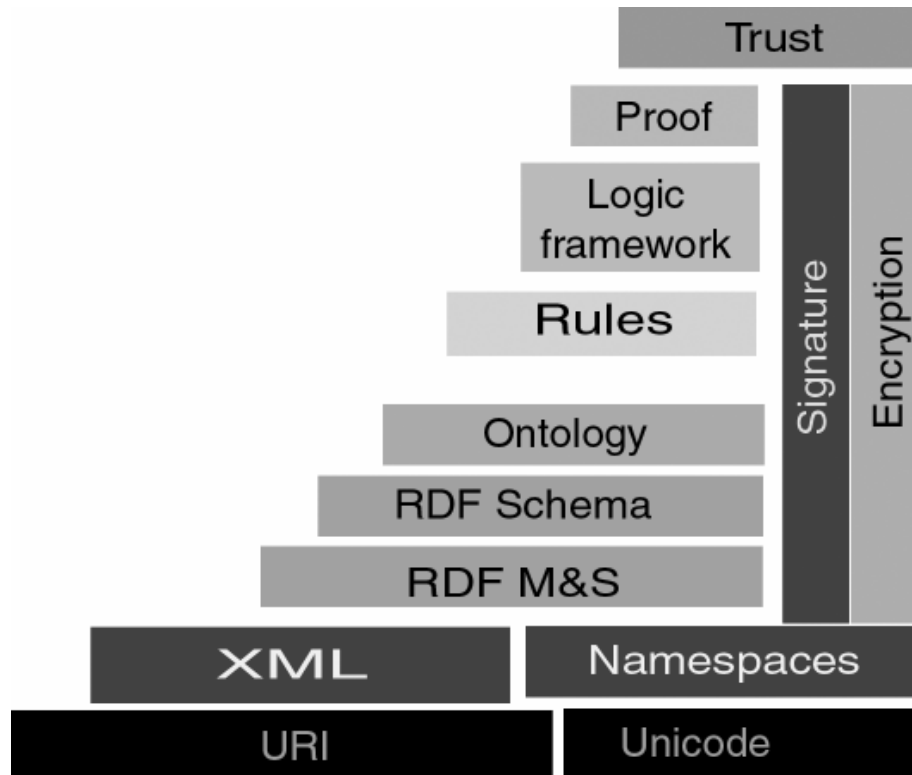


Figure 1 - Expressive power from Semantic Web [12]

As a brief description of the above diagram, we can consider that by layering the entire Semantic Web on top of URI (Uniform Resource Identifier) repositories and Unicode, all objects in the universe will have a unique identifier, despite the difference in language from the world. Through mapping of XML (Extensible Markup Language) and namespaces, objects can be referred in a decentralized manner that anybody can say something (considered as data) about anything (the metadata), simply by pointing the objects they are referring to the URI repository. RDF and RDF schema provide a validated structure for the metadata that anybody attempts to get involve will have to follow. By applying ontology, relationship and relevance between terms and languages can be corresponded to each other. Having metadata described in RDF format, which can be in turn considered as declarative predicates, rules can be inferred from the content in logical sense and hence support the proving of the theories. While all these are backed up by digital signature and encryption, privacy and a web of trust can be obtained as we will then be able to tell the inference rules and the content we retrieved are signed by people we trust.

Referring back to the two limitations mentioned above with the existing search engine, we may be able to enhance the search process by Semantic Web. Focus on RDF metadata provided and the potential rule inferring capability from the declarative predicates, search engine will be able to consume the content retrieved and perform reasoning and expand the result of the search space by relating a concept to another as described in the metadata. By applying reinforcement learning into the Semantic Web search engine, the agent will be able to build profile based searching strategy depends on the interest and feed back from user.

3 Reinforcement Learning

Briefly speaking, machine learning involves sets of computer algorithms such that by following these algorithms, computer program will suppose to learn and improve the efficiency over time. We can consider the definition of learning process as follow: A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E [15].

Being a kind of machine learning algorithm, reinforcement learning differs from other kind of supervised learning in that the learner is never told the correct action to perform in order to reach a particular goal; instead, an optimal policy (a series of state-action pair) is learned by providing rewards or punishment to every step based on how good or bad the previous action is taken. Such a life-long learn-by-experience approach adopted in reinforcement learning is very efficient in situation when the environment, or even the goal for the entire training process, is subject to changes due to user preferences, trend adjustments, etc. This session will review some basic concepts of reinforcement learning, especially focusing on Q Learning, a branch of reinforcement learning algorithm that will be adopted in the search agent.

3.1 State, Action and Reward

The term reinforcement learning refers to a kind of algorithm that attempts to learn an optimal strategy for a series of actions (a policy) to be taken at particular states, in order to maximize the rewards (and minimize the punishment) to be given. One of the strength in reinforcement learning is that it provides formalism for measuring the utility of actions that give no immediate benefit, but is beneficial in the future. Such a delayed benefit is actually represented by learning a mapping from each available action to a scalar value indicating the sum of future discounted rewards expected for executing that action. The ‘discount’ makes later rewards less valuable than sooner rewards, thus encouraging expediency.

This characteristic has been implemented in different kinds of agents that must keep learning to choose actions that alter the state of its environment and where a cumulative reward function is used to define the quality of any given action sequence – be aware that this indicates four main areas in which reinforcement learning differ from other approximation learning algorithm [15]:

- ◆ Delayed reward. Unlike other learning algorithms in which the learner is told if an optimal action a is determined correctly in a state s using a target function π (that is, training examples are in the form of $\langle s, \pi(s) \rangle$), reinforcement learning doesn’t have such training examples. The agent, however, is given an immediate reward values whenever an action is determined at every particular states. The agent has to use this temporal credit assignment to evaluate the best policy that may be lead to an eventual reward.
- ◆ Exploration. Another major difference between reinforcement learning and other learning mechanism is in reinforcement learning, the learner can influence on the distribution of the training examples by determining which actions to take. This characteristic gives the learner a crucial decision in advance, that is, what is the strategy to be used when determining the actions – the learner can either bare higher risk and explore into unknown states and actions for a potential higher rewards; or the learner can play safe to exploit with the states and actions that it is already aware of in returning high rewards.
- ◆ Partially observable states. In most situations, only partial states of the entire environment will be given to the learner in a reinforcement learning problem. The learner will then have to use the

limited information together with a suitable strategy to reach the pre-defined goal; together with the exploration characteristic, usually there is a trade off in observing more different kinds of states by adopting an exploration strategy (which may bring a low reward) against maximizing the amount of rewards in a short period of time.

- ◆ Life-long learning. Every decision made by the learner is another training example, by enriching the experience over time, the learner will then have more and more observed optimal policy for different states. One key point to remember is that, any optimal strategy is determined by the reward (or punishment) that the learner received; and by changing the rewarding function, the learner will be able to behave totally different to adopt the changed environment. This is one of the strengths in reinforcement learning – the learning process can be optimized, but it is not the end as the behavior of the learner can still be adjusted according to a new rewarding mechanism.

3.2 Value of a policy

Under Markovian assumption, Mitchell [15] mentioned that the task of learning a policy can be described as follows, assume:

- ◆ There is a set of states $s \in S$.
- ◆ There is a set of actions $a \in A$.
- ◆ There is a state-action transition function $\delta : S \times A \rightarrow S$.
- ◆ There is a reward function $R : S \times A \rightarrow \mathcal{R}$.
- ◆ And the goal is to learn a policy $\pi : S \rightarrow A$, which is a mapping from states to actions and should maximize the sum of its reward over time.
- ◆ The reward over time is discounted by a discount factor $\gamma : 0 \leq \gamma < 1$ such that an extremely low discount factor (i.e. $\gamma = 0$) will only consider the immediate rewards while a higher discount factor (i.e. γ close to 1) will give greater emphasis on future rewards against immediate rewards.

With the above definition, we can then consider the value of each state when adopting a policy π as:

$$\begin{aligned} V^\pi(s) &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots \\ &= \sum_{t=0}^{\infty} \gamma^t r_t \end{aligned} \quad (1)$$

Having mentioned the value of a policy, we now define the optimal policy π^* by the following notation:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} V^\pi(s), (\forall s) \quad (2)$$

The value of an optimal policy, therefore, is the policy which can give the maximum discounted cumulative reward beginning at state s .

If we consider V^* as the evaluation function a learner should adopt to determine the optimal policy, the agent should then prefer s_1 over s_2 when $V^*(s_1) > V^*(s_2)$. However, this assumes that the learner is choosing among states instead of actions. As suggested by Mitchell, we can instead rewrite the notation as:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} [r(s,a) + \gamma V^*(\delta(s,a))] \quad (3)$$

This indicates that the optimal action in a particular state is the one which can maximize the sum of immediate reward plus the discounted V^* of the successor state $\delta(s,a)$. The above evaluation has a strong assumption, that is, the learner has perfect knowledge and is aware of the result of the transition function and reward function after taking the action.

Unfortunately, it is rarely possible to have complete domain knowledge in the problem regarding the reward and transition function. This is especially true in the situation of this project, when there can be any number of hyperlinks, and the reward is based on the user's preference. Hence we will need something different to evaluate the optimal policy.

3.3 Q Learning

Q learning is one of the reinforcement learning algorithms and by definition, the value of Q is the reward received right after executing an action at a state plus the discounted value following the optimal policy afterwards.

$$Q(s,a) = r(s,a) + \gamma V^*(\delta(s,a)) \quad (4)$$

Combining (3) and (4), we can then tell the optimal policy by:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} Q(s,a) \quad (5)$$

Having rewritten the optimal policy as (5), the learner will then be able to determine the best actions within the state by the Q value even though the reward and transition function is unknown.

The only step we need to concern at this point is how the Q value can be learnt. In a deterministic world when the new state after applying actions to an existing state can be predicted, we can learn the estimated Q as follow:

$$Q(s,a) \leftarrow r + \gamma \max_{a'} Q(s',a') \quad (6)$$

As an explanation of the above notation, we can understand as the estimated $Q(s,a)$ value is determined by the value of immediate reward r received right after applying action a at state s plus the discounted maximum estimated Q value at the succeeding state s' after applying optimal action a' (s' is the succeeding state after taking action a at state s , and a' is the set of actions available at state s').

In a non-deterministic world where the reward function and transition function have probabilistic outcomes, the estimated Q value can be learnt as follow:

$$Q_n(s,a) \leftarrow (1-\alpha_n) Q_{n-1}(s,a) + \alpha_n [r + \gamma \max_{a'} Q_{n-1}(s',a')] \quad (7)$$

and

$$\alpha_n = \frac{1}{1 + \text{visits}_n(s,a)} \quad (8)$$

The key difference of non-deterministic case from deterministic is the use of α_n . Be aware that $\text{visits}_n(s,a)$ indicates the total number of this state-action pair has been visited before, including the current visit. As the number of visits increase, the value of α_n decreased and hence the updates of the estimated Q_s value will become slower as training progresses. This allows the estimated Q function converge to the correct Q function.

The only last piece left behind is the approach to determine which action a to be adopted at any state s . It is important because if we always only select the action a that maximize the estimate Q value, the learner will over commit to the previous experience at the early training and won't be able to find a potential better strategy. However, if we apply probabilistic approach to all the actions in a state such that the probabilities assigned is proportional to the estimated Q value it can receive (i.e. actions with higher estimated Q values will have a higher probability assigned, while none of these actions will bare a zero probability), we can then determine the action to be chosen as follow:

$$P(a_i|s) = \frac{K^{Q(s,a_i)}}{\sum_j K^{Q(s,a_j)}} \quad (9)$$

The constant $k > 0$ here indicates the degree of exploration the learner should adopt. A large value of k will assign higher probabilities to actions that have above average estimated Q values, hence the learner will be exploit to what it has learnt and attempt to maximize the reward; on the other hand, a smaller k value will drive the learner to explore different action even though the selected actions that may currently give below average rewards.

3.4 Applications

Having reviewed some basic background of reinforcement learning, this session will give a brief introduction on how reinforcement learning has been applied into applications.

The key reason that reinforcement learning is getting popular is because it can serve both as a theoretical tool for analyzing how agents can learn to react in different situations and as a practical computational tool for constructing automated systems that improve performance based on increasing experience. In addition, reinforcement learning is very flexible – it can be incorporated with other machine learning algorithms such as neural network and genetic programming to maximize the efficiency. Applications involved reinforcement learning include robotics, industrial manufacturing, game playing, and of course, as mentioned before, search engines.

There are a few variations in reinforcement learning itself, namely Q learning, temporal difference, n step prediction, etc.; and researchers have been asking interesting questions when applying the algorithm into applications. Such questions include instead of adopting a fixed exploration strategy all the time, can the agent adopt different strategy depends on the stage where the agent is? For example, at the beginning of the training stage, the agent may want to explore more observable states while when the agent is close to convergence, it may want to be exploit on the experience and knowledge it already has. Determining the strategy to adopt in the agent is especially efficient in certain areas especially when the

total number of state and action pairs can be predicted in advance (even though the search space is still enormous).

Game playing can be considered as the area where reinforcement learning has been most welcomed. One of the main reasons is because the process can be adapted in very general class of games, and the reinforcement learning agent can learn by playing against human, or even learn by playing against itself – as long as there is a rewarding mechanism telling the agent how good or bad it has been performing.

As mentioned in [15], Tesauro applied reinforcement learning on a back propagation based three-layered neural network in backgammon. After 1.5 million self-played training by simply choosing a move that will always maximize the expected probability of victory, the agent achieve a result of losing 1 point in 40 games when playing with the top human professional players; and the agent now plays at a level competitive with a human world champion.

In addition to gaming, reinforcement learning has been applied in other daily life applications as well. A research has been done on applying Q learning in elevators to minimize the average wait time for passengers [13]. The result of this simulation surpasses the best of the heuristic elevator algorithm by around 7%.

In order to further enhance the performance of application adopting reinforcement learning, [22] mentioned that it may be worthwhile to include some background knowledge of the task in order to minimize the time required for convergence of the learning function; although supplying such background knowledge will come for a price of extra human effort required to foresee the entire process and less automate system.

4 The Semantic Web

The entire web development has been revolving, from hand written HTML codes in the very first stage, to include application server and generate dynamic pages linked to relational database, to deploy service to client through the internet via Web Services. More and more automatic and machine processes are included to provide information for human consumption. While machine has been involved heavily to generate this huge amount of resources, none of these is targeted for machine itself to consume. An example is the search engine we have been taking for granted so far: machines have no idea of the meaning of the search phrase, but simply perform a text matching, users are still the ones doing the reasoning of the process. The main obstacle here is obvious, in order for a machine to consume a piece of information generated by another machine, the agents we use as the bridge between two parties must be communicate in a way to ensure that the meaning of one says is accurately interpreted by the other. Such a semantic processing will require an agreement over a set of concepts and justify the inclusion and exclusion of result purely based on relationships between concepts. The Semantic Web proposed by Tim Berners-Lee since 1998 is targeted to solve this problem – an evolution of the current World Wide Web into a gigantic machine readable and understandable network of resources.

4.1 What is Semantics?

Basically there are three questions need to be understood by two agents when they attempt to communicate with each other.

- ◆ What does that mean? When an agent says the word ‘price’, the other agent should be able to understand it refers to ‘something that one has to give up in exchange of a product/service’, to make it short – money (to push it a little bit further, why not the stones or shells that one had to collect and carry thousands years ago?).
- ◆ Does that mean anything else? What is the ‘film’ an agent is talking about? Is that the movie now showing on theatres? Or the physical rubber wheels put in the camera when taking the movie?
- ◆ How about the variance? Instead of referring to ‘creator’ of a song, an agent may refer to ‘composer’ instead, and agents should be aware that ‘writer’ and ‘author’ are both ‘creator’ of a book.

From the previous examples, we can see that for agent to understand each other, along the message there must be a pointer to what ontology the sender is using. As a matter of fact in the above situations, what we concern most is not the amount of money, the content of the film nor the name of the creator (data), but the description of what money, film and creator (metadata or data about data) are. The importance of getting a common agreement on understanding the metadata between agents comes the ‘semantics’ of Semantic Web.

There are generally 2 kinds of machine-processable semantics that we can define: procedural semantics and declarative semantics [5]. In brief, procedural semantics are symbols (vocabularies, characters, etc.) that human embedded in a procedure fully intended for machine processing, so that when the machine encounters such a symbol, the corresponding procedure will be carried out to determine the meaning of the symbol. Declarative semantics describe the meaning of a new term from a formal, declarative specification under the assumption that the language is shared, the concepts are compatible and shared as well. Given this perspective, RDF is declarative semantics.

The concept of machine-understandable documents does not imply some magical artificial intelligence [7] which allows machines to comprehend human mumblings. It only indicates a machine's ability to solve a well-defined problem by performing well-defined operations on existing well-defined data. Instead of asking machines to understand people's language, it involves asking people to make the extra effort

4.2 RDF as metadata

As described above, the most important part of Semantic Web is the way how machines can agree on the metadata sending back and forth between them. Consider a simple example that if there is only one dictionary in the world, and anything in the world can have an entry or identifier in this dictionary, anyone that needs an explanation of a certain word can simply do a look up in the dictionary to get an explanation. Despite the different languages people are using all over the world, will this approach solve the problem that has been raised regarding the agreement of the terminology?

If there is an organization in the internet that keeps repositories of everything (e.g. definition of vocabularies, ISBN of books, persons, events...) and a unique identifier is assigned to each item, agents can then be able to look up the general metadata by pointing to the identifier of the item in this repository. Assuming items are classified in a hierarchal manner, the agent will then be able to realize relationship and equivalence such as 'movie' is equivalent to 'film'.

Resource Description Framework (RDF) is the W3C recommended way to store generic metadata. RDF is an XML format such that the metadata can always apply to a Uniform Resource Identifier (URI) instead of to the parent item in the XML itself. Such a structure in referring to an external resource instead of the parent node within the XML tree allows one-to-many and many-to-one relationships to be expressed. The characteristic of RDF is that it is designed to be general – so that other prospective applications can be mapped onto the model. Because it is general, it is simple – so simple that not much can be done by RDF alone without layering many things on top [6].

RDF metadata can be used in a variety of application areas; for example: in resource discovery to provide better search engine capabilities; in cataloging for describing the content and content relationships available at a particular Web site, page or digital library; by intelligent software agents to facilitate knowledge sharing and exchange; in content rating; in describing collections of pages that represent a single logical “document”; for describing intellectual property rights of Web pages, and in many others. RDF with digital signatures will be the key element to building the “Web of Trust” for electronic commerce, collaboration and other applications.

The broad goal of RDF is to define a mechanism for describing resources that makes no assumptions about a particular application domain, nor defines (a priori) the semantics of any application domain. The definition of the mechanism should be domain neutral, yet the mechanism should be suitable for describing information about any domain.

In addition to RDF from W3C, a set of common tags for categorizing and describing web pages were created. These tags, called Dublin Core (DC), define the most common metadata that may be adopted in web pages. There are altogether 15 elements in DC, and detail description can be found in <http://dublincore.org/documents/2003/02/04/dces/>.

By considering RDF as a motivation to provide a standard for metadata describing the resources on the Web, the basic construction in RDF is a subject-predicate-object triple (while some other resources refer it as the object-attribute-value triple), and graphically it can be seen as two nodes (subject and object) connected by an arc (predicate). For example a webpage W (subject) has a property Creator (predicate)

value A (object). This ‘has a’ relationship can be written as Creator(W, A). The Creator A is then considered as the metadata of the web page W. Note that the distinction between data and metadata is sophisticated, an item representing an object in an instance can be considered as a subject in another (consider the example ‘W has a creator value A’ and ‘A has a nationality value N’ – ‘A’ is used as object in one and as subject in another). A simple example of a RDF definition may look like this:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description
    rdf:about="http://www.cs.bris.ac.uk/home/hv2625/aboutme.htm"
    dc:creator="Stanley Vong"
    dc:title="Something about me..."
    dc:description="Integrating first RDF statements into my page."
    dc:date="2003-03-09" />
</rdf:RDF>
```

A brief explanation of the above RDF: ‘xmlns’ is the keyword for name space definition that generally used as a group of elements and attributes that are recognizable by the prefix. In the above example, we have 2 name spaces defined ‘rdf’ and ‘dc’ pointing to 2 specific URIs. Note that the URI in the name space itself by nature doesn’t require pointing to anything (i.e. it doesn’t require the existence of a corresponding web page), they are mainly to ensure that the name spaces are unique.

This RDF describes **about** the subject ‘http://www.cs.bris.ac.uk/home/hv2625/aboutme.htm’, which obviously is a web page (it should be pointed out at this moment that the object doesn’t have to be a web page, it can be anything such as an event, a product, a song, a country, etc., as long as it is a unique URI). The subject has predicates **creator**, **title**, **description** and **date** with corresponding object values. Notice the name space (dc:) prefixing the attributes, it indicates that the definition of the attributes refers to whatever can be found in the dc URI, which are 4 out of the 15 elements we can find in Dublin Core.

There are several RDF validation services provided on the internet using Java, Prolog etc. as the parser. Choosing the one provided by W3C [21], we can see that the above RDF definition can be clearly parsed as the following triples and figure.

Subject	Predicate	Object
http://cs.bris.ac.uk/home/hv2625/aboutme.htm	http://purl.org/dc/elements/1.1/creator	"Stanley Vong"
http://cs.bris.ac.uk/home/hv2625/aboutme.htm	http://purl.org/dc/elements/1.1/title	"Something about me..."
http://cs.bris.ac.uk/home/hv2625/aboutme.htm	http://purl.org/dc/elements/1.1/description	"Integrating first RDF statements into my page."
http://cs.bris.ac.uk/home/hv2625/aboutme.htm	http://purl.org/dc/elements/1.1/date	"2003-03-09"

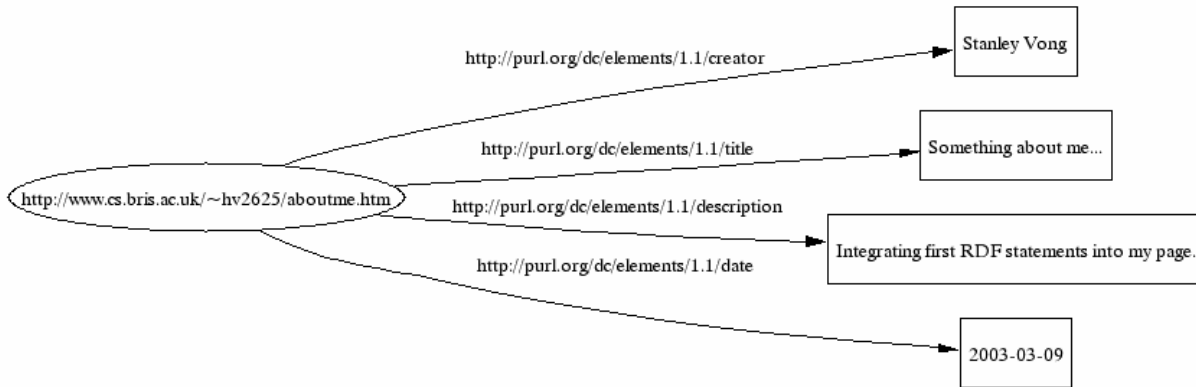


Figure 2 - Data model generated by W3C RDF validation service [21]

4.3 Semantic Web and Data Retrieval

Given the possibility that machine agents can crawl across the Web and retrieve information with a certain degree of understanding the data, we can then consider the entire Web as a gigantic relational database that the agent no longer need a text parser to parse the page word by word, but an XML parser to parse the tags to retrieve the data and the metadata, so that the agent can further look up the description of the metadata if necessary. As a matter of fact, the Semantic Web data model is very directly connected with the relational database model [16]. A relational database consists of tables, which consists of rows, or records. Each record consists of a set of fields. The record is nothing but the content of its fields, just as an RDF node is nothing but the connections: the property values. The mapping is very direct:

- ◆ A record is an RDF node;
- ◆ The field (column) name is RDF property type;
- ◆ The record field (table cell) is a value.

Baring the above mapping in mind, we can then enhance the data retrieving process a little bit more by mapping Relational Database table with RDF. Consider each record in the table has its own URI and the database table D having a schema:

$$D = \{a1,a2,a3,a4\}$$

Where a1 as the title of a book, a2 as the author, a3 as the publisher and a4 as the publish year, we can ‘flatten’ each single record in the database table like:

Subject	Predicate	Object
http://www.book.org/a1.htm	http://purl.org/dc/elements/1.1/title	Machine Learning
http://www.book.org/a1.htm	http://purl.org/dc/elements/1.1/creator	T Mitchell
http://www.book.org/a1.htm	http://purl.org/dc/elements/1.1/publisher	McGraw-Hill
http://www.book.org/a1.htm	http://purl.org/dc/elements/1.1/date	1997

Doesn't the table look familiar? It is exactly the same data model as RDF, and this implies that if the Web can evolve and all web sites can present their data model in RDF, we will have the potential power to build a world-wide-gigantic relational database, which will be priceless in terms of information retrieval and knowledge sharing.

It should be pointed out that instead of a specific string in the ‘Object’ column, it should be replaced with a corresponding URI if there is any (e.g. instead of having the piece of string ‘Machine Learning’ as the value, it should be replaced by a specific URI pointing to, let’s say, the repository of all the books from the publisher). When content in database are defined using URI, we can then be able to relate an item to the others (all the books written by the same author), find equivalence of vocabularies in symmetry (theatre and cinema), and discover certain hierarchal structure (shoes to boots).

4.4 What else can be done?

RDF performs as a basic framework for the entire Semantic Web. Within this framework, as mentioned above, metadata is presented in a structured and common style for combining information from different applications. The applications built under the technology are numerous and building the RDF is only the basic core. Even though these applications totally depend on the final outcome of the Semantic Web and may be beyond the scope of this project, it is worth to take give a brief description to show what might happen in the future [6].

- ◆ Conversion language. Considering all the triples in RDF are prefixed with namespaces, such a requirement will allow the conversion of a document from one RDF schema into another by following rules and a share of a common RDF knowledge at some level. As an example, consider we have two database constructed by two different entities keeping the address of customers, the ‘zip’ code of United States in one database can be automatically mapped to the ‘postal’ code of United Kingdom in another assuming the presentation of the databases follow the requirement of RDF schema and both the ‘zip’ and ‘postal’ code column refer to a same namespace which has the same logical inference but a more general and common presentation, such as ‘region’.
- ◆ Digital Signature. Public key cryptography and digital signature is getting common to enhance the trust of electronic documents. With Semantic Web, we can incorporate this remarkable technology into a reasoning task. After integrate with a signature verification system, the reasoning engine will not only parse the documents into trees of assertions but also into trees of assertions about who has signed what assertions. The beauty of this integration will assist to build a web of trust so that the logical models in the reasoning can be extended to include the keys with which assertions have been signed. By checking the signature, web user will then be able to determine if the result of an inference rule from the engine should be adopted and implemented.
- ◆ Reasoning Engine. Repeated several times in this document already, the current search engine performs only a hit-and-miss on bags of words with the web content, accuracy can be barely evaluated with the strategy if we consider ‘correctness’ as something more than the existence of the search phrase. Building on top of a logical layer, the implementation of Semantic Web means that data are structured into declarative triples – structures that are preferred by machine learning as reasoning can be done with provable correct answer. The combination of a search engine and reasoning engine will construct proofs in a certain number of cases of very real impact, an example can be collecting a complete lists of all occurrences of a search phrase and then draw logical conclusion on those which can be used to solve a given problem after inference rules are generalized from the available documents.

If everything falls into Berners-Lee’s vision, Semantic Web will be powerful in turning the functionality of the entire Internet into a new era. However, it remains a vision and not yet a reality [5]; the fundamental basic requirement of Semantic Web makes a strong assumption that everything has an URI and follows the RDF framework, metadata is centralized in repositories that decentralized data can be referred to – all these aren’t true at the moment, and will not be there for any time soon.

4.5 Reasoning in Search Engine

Back to the original problems we have mentioned in the beginning of this document, that is, result of a search process totally depends on the existence of search phrases in web documents and the target consumer of the knowledge discovery being human, how can the search process improve by using the RDF framework in Semantic Web? Consider the following example, I know the name of a song and I want to know the creator (composer) and all other songs from the same creator. In the current search process based on the existing Web, I will put in the name of the song I know as the search phrase in the engine, and locate manually the name of the creator from all the web pages returned by the engine, and then manually perform another search by using the creator as the search phrase. Finally I should be able to track all the songs from the creator. (As a matter of fact, we can consider the creator as the Least General Generalization (lgg) in the metadata of all the songs we are targeting in the previous example. By definition, we can understand lgg of two terms as the term with the least difference from both t1 and t2 that is more general than both t1 and t2 at the same time.)

With Semantic Web, the task can be solved a little bit easier. Assuming the name of the song I put in the search phrase of the Semantic Web agent is 'Two Of Us', the agents crawl across the Web and find the metadata of the song in RDF format. To illustrate this I have extracted portion of a RDF page from MusicBrainz [18] web site, check the URL below to see the entire RDF file:

<http://mm.musicbrainz.org/mm-2.1/track/3cd2b608-b48e-45c2-9d44-461195ca5207>

Partially extracted RDF:

```
<mm:Track rdf:about="http://musicbrainz.org/track/3cd2b608-b48e-45c2-9d44-461195ca5207">
  <dc:title>Two Of Us</dc:title>
  <dc:creator rdf:resource="http://musicbrainz.org/artist/b10bbbf9e-42e0-be17-e2c3e1d2600d"/>
  <mm:duration>612373</mm:duration>
</mm:Track>
```

Using the RDF validation service mentioned in previous session, the following RDF triple and model are generated for clarity:

Subject	Predicate	Object
http://musicbrainz.org/track/3cd2b608-b48e-45c2-9d44-461195ca5207	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://musicbrainz.org/mm/mm-2.1#Track
http://musicbrainz.org/track/3cd2b608-b48e-45c2-9d44-461195ca5207	http://purl.org/dc/elements/1.1/title	"Two Of Us"
http://musicbrainz.org/track/3cd2b608-b48e-45c2-9d44-461195ca5207	http://purl.org/dc/elements/1.1/creator	http://musicbrainz.org/artist/b10bbbf9e-42e0-be17-e2c3e1d2600d
http://musicbrainz.org/track/3cd2b608-b48e-45c2-9d44-461195ca5207	http://musicbrainz.org/mm/mm-2.1#duration	"612373"

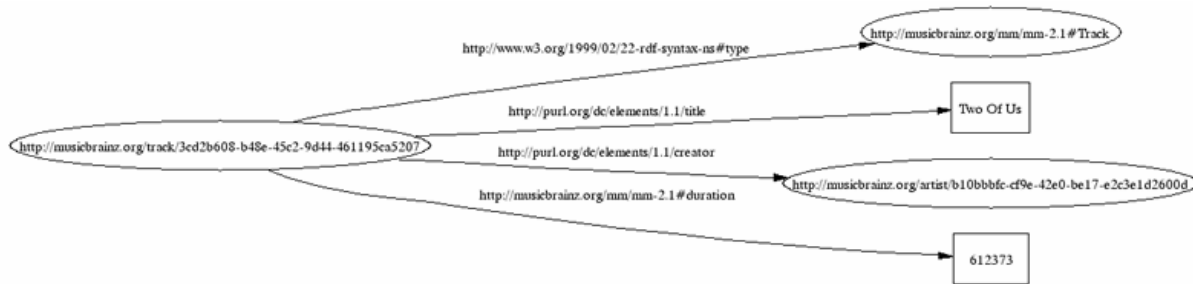


Figure 3 - Data model for RDF from MusicBrainz

Review the metadata `<dc:creator>` tag shown in *italics* above, it indicates the creator of the song (whatever the value may point to). The search process now can then be simply to consider the creator as new search criteria. The following RDF describing the artist will be selected for having the `<dc:creator>` value in `<mm:Artist>` predicate:

```
<mm:Artist rdf:about="http://musicbrainz.org/artist/b10bbbf-cf9e-42e0-be17-e2c3e1d2600d">
  <dc:title>The Beatles</dc:title>
  <mm:sortName>Beatles, The</mm:sortName>
</mm:Artist>
```

Yet that is not the end of the story, another song with the following metadata will be retrieved by the application because it has the same `<dc:creator>` value.

```
<mm:Track rdf:about="http://musicbrainz.org/track/be22e4b4-8296-4d01-9705-b1429e1a42cd">
  <dc:title>I Should Have Known Better</dc:title>
  <dc:creator rdf:resource="http://musicbrainz.org/artist/b10bbbf-cf9e-42e0-be17-e2c3e1d2600d"/>
  <mm:duration>164106</mm:duration>
  <mm:trmidList>
    <rdf:Bag>
      <rdf:li rdf:resource="http://musicbrainz.org/trmid/9c216775-23fc-46af-af8f-bcd4b330a161"/>
      <rdf:li rdf:resource="http://musicbrainz.org/trmid/111f8c82-a0fc-4cee-97ef-04e511360333"/>
      <rdf:li rdf:resource="http://musicbrainz.org/trmid/60246b65-80d6-4572-bcef-f829594a6341"/>
    </rdf:Bag>
  </mm:trmidList>
</mm:Track>
```

As the structure of RDF is declarative triple, if we consider the search process as resolution in predicate logic, it will be something similar to: `Creator(A, 'Two Of Us')`, resolve the value of A and then apply A to other predicates will expand the search space, and the result will be much richer by including the metadata related to the original search criteria compare to only using the piece of string 'Two Of Us'. In the above example the result is expanded from finding web pages having information about 'Two Of Us', to web page having information about the creator 'The Beatles', to another web page with song 'I Should Have Known Better'. In other words, the sequence of the above knowledge discovery can be considered as:

1. Search for the song having title(A, 'Two Of Us').
2. Retrieve the creator of the song by creator(A, B).
3. Search for the creator having about(B).
4. Retrieve the name of the creator having sortName(B, C).
5. Search for the set of songs S such that the song $D \in S$ has creator(D, B).

Following the above example, the value of:

A – ‘<http://musicbrainz.org/track/3cd2b608-b48e-45c2-9d44-461195ca5207>’

B – ‘<http://musicbrainz.org/artist/b10bbbf9e-42e0-be17-e2c3e1d2600d>’

C – ‘Beatles, The’

D – ‘<http://musicbrainz.org/track/be22e4b4-8296-4d01-9705-b1429e1a42cd>’

The learning task should then be able to automatically identify all the relevant pages and return to the user according to the user’s interest. The opened question so far is to determine the predicates the agent should include in refining the search criteria – one obvious example is the user may not be interested in any other songs having the same duration. In order to achieve this, the agent should be able to capture the user’s preference – reinforcement learning is a good choice to perform the task.

5 Time Plan

	June	July	August	September
Getting familiar with API:				
XML Parser	====>			
Serialization	====>			
Threads	====>			
HTTP Connection	====>			
System Design:				
Data Flow	====>			
Incorporating Reinforcement Learning		====>		
Implementation:				
Search Agent		=====>		
Filtering Agent		=====>		
Interim Documentation		====>		
Learning Agent			=====>	
GUI			====>	
Integration & Testing			====>	
Review and Enhancement			=====>	
Documentation:				=====>
Source Code and Document Cleanup:				=====>

Part II

Project Plans

6 Aims, Objectives and Motivation

The main objective of this project is to build a reinforcement learning directed search agent on Semantic Web pages.

Learning from the Web is normally considered as a difficult task due to the lack of structure in documents. Typically machines use a ‘bag of word’ strategy in the learning process. For example, search engines we have been using so far crawl across different web pages and index all these web pages by keywords, metadata of the header session, etc. Users provide a search phrase to the search engine, the engine will then perform lookup on the index table and return relevant web pages to the user. This process is fast and powerful, but the drawback is lack of concept understanding and extensibility – the search phrase must exist in the web page.

If we consider the search process as a confusion matrix shown below, then the on-topic web pages are those which the search phrase exists, while the off-topic web pages are those which the search phrase doesn’t exist. The search engine should attempt to maximize quadrant 1 (true positive) and minimize quadrant 2 (false positive) and 3 (false negative). Although one may argue that there is no way to confirm how many on-topic web pages have been discarded in the entire web space; similarly, it may not be feasible to determine how many off-topic web pages have been discarded in quadrant 4 (true negative), concerning the number of pages in the Web are enormous.

		Existence	
		On-Topic	Off-Topic
Engine	Select	1	2
	Discard	3	4

Introducing the potential capability of Semantic Web, search engine should not be limited on only locating the shortest path to relevant web pages by focusing on the search phrase provide by users, but also automatically relates the search phrase to closely related topics and return the web pages to users as well. One of the example is user provides the name of a song to the search engine, the engine will then be able to identify the song, together with relevant RDF metadata (such as composer, singer or publisher), and identify all other songs that share similar set of metadata (e.g. from the same composer, singer, publisher, etc.). In addition, depends on the background, preference, personal interest of the user, the search engine should be able to behave differently even when the same search phrase is provided.

With the above two ideas (relating topics and user oriented) in mind, I consider this project as a test stone throwing into the infant pool of Semantic Web to investigate the potential capability of applying machine learning technique into the structured RDF metadata in Semantic Web. In addition, since this project focus on the potential capability in enhancing searching experience in Semantic Web, the engine will only consider Semantic Web pages (i.e. pages with metadata described in RDF) in the search space. To have this done, I will limit myself to a particular domain (web site) where Semantic Web pages have been implemented as core. Web pages even with the search phrase exist within the text but not any related RDF will be discarded; there are a lot Bayesian classifiers performing the task already.

The main result I attempt to achieve, is the integration of one of the trendiest technology (RDF in Semantic Web) with a machine learning algorithm (Reinforcement Learning). Even though the project will be limited on certain domain, the limitation can be removed easily once there are sufficient web sites implementing Semantic Web. I hope this project will help to increase the web search experience and be useful to applications that may need to apply other machine learning technique into the Semantic Web.

7 Initial Design

This session will cover the preliminary design that currently have for the application. Briefly speaking, the work flow of the entire process is:

- ◆ There will be a graphical user interface allowing the user to put in a profile name, search phrase and select a particular domain (initial web page), which supposed to be web site implementing Semantic Web with RDF metadata.
- ◆ Multiple search agents will be released by the application and retrieve web pages through a HTTP connection, starting from the initial page provided.
- ◆ Search agents will keep following hyperlinks on web pages until RDF data is found (RDF can be either embedded in a regular HTML page or isolated as a stand alone XML page).
- ◆ The page linked to the RDF (i.e. the path), together with the metadata, will be returned to a filter agent. The filter agent will determine the relevance of the page by comparing the metadata against the search phrase.
- ◆ Hyperlinks will then be returned to user, user reviews the page and give a reward depends on the relevance of the page against the search keyword.
- ◆ Based on the reward given, a learning agent will then evaluate the Q value of the link, and then determine if the additional information in the metadata should be used as extra search keyword. Search process continues.

The last point is important in the sense that the learning agent will not only evaluate the Q value of the existing page, but will also help to determine the relevance of other web pages which share similar metadata to the page.

7.1 Techniques and Tools

To complete the task, I have decided to adopt the following techniques and tools in the application. Basically I have decided to use Java as the programming language; in addition I will need an XML parser for the RDF data parsing; a way to permanently keep the knowledge learnt by the agent and a solution to implement multi-agents.

7.1.1 Java

The program will be implemented in Java. It is only a personal preference, and also because in the future I want to implement the idea back to the IBM mainframe that I am most familiar with, hence Java is a convincing choice for me. Another reason is because Java works closely with XML, there are a lot of XML parsers, two of the most common ones are SAX and DOM.

7.1.2 XML Parser

SAX (Simple API for XML) is an event-based model for accessing XML document contents. SAX invokes methods when certain conditions (e.g. start tags and end tags) are encountered as an XML document is sequentially read from start to finish.

DOM (Document Object Model) is a garden-variety tree structure consisting of a hierarchy of nodes stored in memory. This structure is used to represent the content and model of an XML document. Once a document is parsed using DOM, that tree-like structure will exist in memory and can be processed in various ways.

In brief, DOM should be more dynamic and flexible than SAX because once the XML document is parsed by DOM, the content is provided as a tree-structure in memory, and nodes can be manipulated individually back and forth. SAX uses a read-forward mechanism that 'backing-up' is not possible. However, I am planning to use SAX in my application mainly because:

- ◆ There is a RDF parser call ARP (Another RDF Parser) created by HP Labs. ARP is built on top of SAX. By adopting ARP in my application, I don't have to reinvent the wheel in terms of writing up my own RDF parsing mechanism.
- ◆ DOM parse the XML document and build the entire tree in memory before any manipulation is possible. Creating such trees in memory based on internet content may easily expose the resources of any machine.
- ◆ I don't need any XML manipulation such as adding new nodes, removing nodes, changing content, etc. All I need is get the value of the nodes, read-forward mechanism should be enough for this purpose.

7.1.3 Database and Serialization

Another important concern is the storage of experience learnt by the agent. It will be time-consuming and impractical if the agent has to be trained from the very beginning every time the application starts. Therefore I will attempt to have the Q table serialize into objects and stored in file or using a JDBC connection to store the value of Q table in a relational database such as MySQL. Serialization may be easier to be implemented as the agent won't require a database installed in the machine, however it may be more memory efficient if the Q table is stored in a relational database.

7.1.4 Threads

As mentioned before in this document, by adopting multi-threaded agents we can maximize the utilization of connection bandwidth and increase the search performance when the agents can share their knowledge and experience with each other. The multi-threaded capability in Java should make the implementation of adopting multi-agents in web searching easier.

7.2 Agents

The search process will be carried out by mainly three sets of different agents, namely searching agent, filtering agent and learning agent.

7.2.1 Searching Agent

The primary duty of the searching agents is to follow hyperlinks and crawl over the Web from document to document. Agents should be coordinated in a way that they should share knowledge among each other, such as visited page shouldn't be re-visited, all possible hyperlinks found in one document should be available to all other agents, etc. Once RDF document is found, the searching agent should return the URL and the path which the agent follows in reaching that document to the control application.

7.2.2 Filtering Agent

There are two main tasks performed by the filtering agents. First, compare the search phrase against the knowledge base (i.e. Q table of web pages) and return what is already known to the user after confirming the web page still exists. Second, based on the result collected by the searching agents, the filtering agents will open the RDF documents, analyze the content of the metadata and determine if the web page should be returned to user according to its relevance of user's search phrase and interest (i.e. another Q table stating the user looking for a song may also want to look for the creator and all other songs of the creator as well).

7.3.3 Learning Agent

A set of hyperlinks is returned to user, the user will review each of them one by one and scale the relevance of the web page by giving a reward or punishment. The learning agent will then reinforce the Q tables of hyperlinks and relevance according to the reward or punishment received. Using the example mentioned in session 4.5, the agent should learn the user interest by considering questions like ‘If the user put in the name of a song, should I look for the creator as well? Should I look for other songs as well? Which predicate I should not use as additional search criteria, probably the duration of the song?’. The agent should learn the strategy to adopt by reinforcing a Q table which relates the metadata from one to another.

7.3 Challenge, Potential Problem and Limitation

The main challenge of this project is to learn if a web page should be returned to user based on the knowledge the agents already have (to minimize the time required) and extend the search from a blind search to a reasoning search (to include related web pages even though the search phrase doesn’t exist). By considering the metadata as a tree or lattice, the agents should be able to crawl from one area to another – whether the agent should attempt such a crawl across, should be learnt from the user’s interest.

The challenge, as a matter of fact, is a potential problem at the same time. Due to the infancy on Semantic Web, there aren’t a lot of RDF metadata available. Research has been done on RDF data on the Web and it has shown that RDF is currently hard to find unless a considerable effort and resources are put into the search process [10]. Lack of resources in this area drives my project have to be focus on specific domain since the very beginning, and the potential capability of the search agent targeting on RDF data as searching criteria is somewhat limited at this moment. However once the Semantic Web gets more common and people start including RDF as the metadata of their web content, the limit on search capability should be removed easily.

Even though there are still some web sites incorporating the idea of Semantic Web, the implementation is not in standard yet. Variances exist in terms of embedding RDF data in the header session of HTML page, creating hyperlink in the HTML page referencing a separate RDF (XML) file, etc. Lack of standardization and the freedom of participation increase the difficulty of the search process; which as a matter of fact, will cause problem when building interface and agents to handle all the variances.

The Semantic Web is currently both there and not there. Throughout this project, I try to identify and investigate the possibility of visions that have been discussed so far but not commonly adopted at the moment. It should be aware that the final outcome of the Semantic Web may be totally different from the current vision when the water gets clear, technology involved get mature, and most important of all, people get ready in re-structuring the content.

8 References & Resources

- [1] Jason Rennie and Andrew Kachites McCallum. Using Reinforcement Learning to Spider the Web Efficiently. Proceedings of ICML-99, 16th International Conference on Machine Learning, 1999.
- [2] Sachiyo Arai, Katia Sycara and Terry R. Payne. Experience-based Reinforcement Learning to Acquire Effective Behavior in a Multi-agent Domain. The Sixth Pacific Rim International Conference on Artificial Intelligence (PRICAI 2000), Springer-Verlag, 2000, pp. 125-135.
- [3] Byoung-Tak Zhang and Young-Woo Seo. Personalized Web-Document Filtering Using Reinforcement Learning. Applied Artificial Intelligence 2001, Vol.15(6), pp.665-685.
- [4] Jason Rennie. Efficient Web Spidering with Reinforcement Learning.
<http://www.watson.org/~jrennie/>
- [5] Michael Uschold. Where is the Semantics in the Semantic Web?
<http://cis.otago.ac.nz/OASWorkshop/Papers/WhereIsTheSemantics.pdf>
- [6] Tim Berners-Lee. Semantic Web Road map.
<http://www.w3.org/DesignIssues/Semantic>
- [7] Tim Berners-Lee. What the Semantic Web can represent, 1998.
<http://www.w3.org/DesignIssues/RDFnot.html>
- [8] Bernard Silver, William J. Frawley, Glenn A. Iba, John Vittal and Kelly Bradford. A Framework for Multi-Paradigmatic Learning. Morgan Kaufmann Publishers Inc., 1990, pp. 348-356.
- [9] Sachiyo Arai, Katia Sycara and Terry R. Payne. Multi-agent Reinforcement learning for Planning and Scheduling Multiple Goals. Proceedings of the Fourth International Conference on MultiAgent Systems, July 2000, pp. 359-360.
- [10] Andreas Eberhart. Survey of RDF data on the Web (second run – August 2002). 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002), Orlando, USA.
- [11] Tim Berners-Lee, James Hendler and Ora Lassila. The Semantic Web. May 2001.
<http://www.scientificamerican.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&catID=2>
- [12] Tim Berners-Lee. Standards, Semantics and Survival.
<http://www.w3.org/2003/Talks/01-siia-tbl/>
- [13] R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. Neural Information Processing Systems 8, MIT Press, 1996.
- [14] Terry R. Payne, Rahul Singh and Katia Sycara. RCal: A Case Study on Semantic Web Agents. The First International Joint Conference on Autonomous Agents and Multi-Agent Systems, 2002, pp802-803.

- [15] Tom Michell. Machine Learning. McGraw-Hill 1997. ISBN0-07-115467-1
- [16] David Dodds. XML Meta Data, Wrox Press 2001, ISBN1-861004-51-6.
- [17] Sutton and Barto. Reinforcement Learning: An Introduction. The MIT Press 1998, ISBN0-262-19398-1
- [18] MusicBrainz.
<http://www.musicbrainz.org>
- [19] Semantic Web from W3C.
<http://www.w3.org/2001/sw/>
- [20] Resource Description Framework from W3C.
<http://www.w3.org/RDF/>
- [21] RDF Validation Service.
<http://www.w3.org/RDF/Validator>
- [22] Reinforcement Learning: A Survey
<http://www.cs.washington.edu/research/jair/volume4/kaelbling96a-html/rl-survey.html>

*** *End* ***