# Crawling the Deep Web Using Asynchronous Advantage Actor Critic Technique

Kapil Madan* and Rajesh Bhatia

*Computer Science & Engineering Department, Punjab Engineering College (Deemed to be University), Sector 12, Chandigarh, India*
*E-mail: er.kapilmadan@gmail.com*
**Corresponding Author*

Received 10 March 2020; Accepted 10 February 2021;
Publication 02 June 2021

## Abstract

In the digital world, World Wide Web magnitude is expanding very promptly. Now a day, a rising number of data-centric websites require a mechanism to crawl the information. The information accessible through hyperlinks can easily be retrieved with general-purpose search engines. A massive chunk of the structured information is invisible behind the search forms. Such immense information is recognized as the deep web and has structured information as compared to the surface web. Crawling the content of deep web is very challenging and requires filling the search forms with suitable queries. This paper proposes an innovative technique using an Asynchronous Advantage Actor-Critic (A3C) to explore the unidentified deep web pages. It is based on the policy gradient deep reinforcement learning technique that parameterizes the policy and value function based on the reward system. A3C has one coordinator and various agents. These agents learn from different environments, update the local gradients to a coordinator, and produce a more stable system. The proposed technique has been validated with Open Directory Project (ODP). The experimental outcome shows that the proposed

*Journal of Web Engineering, Vol. 20_3,* 879–902.
doi: 10.13052/jwe1540-9589.20314

technique outperforms most of the prevailing techniques based on various metrics such as average precision-recall, average harvest rate, and coverage ratio.

**Keywords:** Web crawler, deep web, reinforcement learning, A3C.

## 1 Introduction

World Wide Web (www) content is increasing very rapidly in the modern era. This information has importance in various domains, e.g., research, education, business, etc. A web crawler is required to retrieve the information automatically. A web crawler is software that traverses the www systematically and automatically. There are billions of websites that contain large amount of information. Web crawler traverses through the hyperlinks that are interconnected to each other and retrieve the Uniform Resource Locators (URL). This URL list is known as Publically Index able Web (PIW) or surface web. In contrast, deep web/hidden web is an extensive collection of high-quality information hidden behind the search forms. It collects dynamically generated web pages retrieved by filling the search forms and successfully submitting them. The general-purpose search engines cannot index dynamically generated deep web pages. The deep web size is considerable as compared to the surface web [1]. The deep web has structured information in comparison to the surface web. Such large and structured information gives the motivation to crawl the deep web. The discovery of deep web pages is challenging because of the unknown query set and its size [2, 3].

**Research Questions:** Following Research Questions (RQ) have been solved in the current work related to deep web crawling.

RQ1: Why deep web needs to be crawled?
RQ2: What existing Reinforcement Learning (RL) algorithms have been used in the deep web?
RQ3: How large state space search problem of the deep web has been handled? Which RL algorithm has been used to address this problem?

**Motivation:** The deep web contains huge chunk of structured information that general search engines cannot crawl. As information is hidden behind the search forms, it can retrieve only by querying the search forms. Q-learning and learning automata are the RL algorithms applied to explore the search forms and retrieve the deep web content. However, Asynchronous Advantage Actor-Critic (A3C) is an RL-based technique that has not been explored in the context of deep web before this work. A3C has been applied in various other domains such as gaming, traffic management, web-based tasks, vehicular networks, etc. A3C has been proved successful in large state space problems. The deep web has the large state space search problem that encourages the use of A3C.

The main challenge here is to explore the unidentified deep web pages. This paper proposes an innovative algorithm using A3C to crawl the deep web through the search forms. A3C is a type of RL technique based on the policy gradient method [4]. It parameterizes both the policy and value function. It has one coordinator and various agents. The coordinator initializes all the agents with global parameters. Various agents work asynchronously and interact with the environment. Agent updates its local parameters and sends parameters to the coordinator after a fixed time. This parallel functioning leads to improve stability of the system. Each agent has a state that represents a webpage. A webpage can be of two types – PIW or deep web page. PIW is accessed by various hyperlinks, whereas various actions access deep web pages. Actions can be query form submission or value submission. Critic calculates the value function that helps to find the optimal policy and captures how good action at a given state. Here, the actor is a policy function that learns through the value function. It selects the actions based on the critic, which leads to reward. The advantage function captures how an action is better than other actions. Action probabilities of various actions have been initialized, and the selection of action has been made based on the probabilities of various actions. The main contribution of this work is improved precision, recall, coverage ratio, and average harvest rate as compared to the existing state of the art techniques such as random, generic frequency, and learning automata. To the best of our knowledge, A3C based technique entitled Deep Web Asynchronous Advantage Actor-Critic (DW-A3C) has been applied first time to the deep web domain to reach unexplored deep web pages. The rest of the paper is organized as follows. Section 2 emphasizes the related work of deep web crawling techniques and RL based techniques. Section 3 presents DW-A3C architecture and its detailed working. The proposed algorithm, to discover the deep web pages with multiple parameters and functions are presented in Section 4. Experimental evaluation, discussion, and performance metrics are given in Section 5. Finally, Section 6 summarizes the conclusion and its scope for further implementation.

## 2 Related Work

In this section, different deep web crawling techniques have been discussed. Various assessment parameters and datasets used in the literature have been analyzed. RL techniques corresponding to the information retrieval domain have been studied. This section also discusses the significance of A3C techniques in different domains.

### 2.1 Deep Web Crawling Techniques

The deep web term was first introduced by Bergman in 2001 and was given importance due to its large size and structured information as compared to the surface web [1]. He et al. proposed an approach based on IP sampling for different web servers to identify the deep web-scale and structural complexity [5]. Raghavan et al. proposed the architecture of the deep web crawler entitled Hidden Web Exposer [6]. This architecture consists of different modules to manage the URLs, search forms, and retrieves the deep web content. Search forms are the entry nodes for the deep web and are filled by the Label Value Set (LVS) manager. LVS manager retrieves the value from various data sources and updates the LVS table. LVS table consists of key-value pairs used to fill the search form. The discovery of the search forms from the billions of connected web pages is a complicated task. A pre-query survey consists of various methods/techniques related to the discovery of search forms [7]. This survey does not include the form filling research papers, which is post-query based techniques. On the contrary, a post-query survey was discussed by Kantorski et al. to explore the form filling challenges and their solutions [8]. Systematic literature review papers have explained both the pre-query and post-query techniques in detail [9, 10]. Madhavan et al. proposed the deep web crawling system to retrieve more content with fewer submissions [11]. An informativeness test was used to evaluate the search forms. It is based on various input values combination and discards the irrelevant one. This technique is applicable to complex search forms that contain multiple text boxes. Ntoulas et al. proposed a technique to generate the queries without human intervention [12] automatically. This category comes under the post-query selection technique. It is based on an approximation algorithm that finds the near-optimal query set. Assessment parameters such as coverage ratio and impact of the initial query were used. The datasets used for evaluation were Open Directory Project (ODP), PubMed, and Amazon website. Barrio et al. proposed query-based sampling strategies to generate an efficient sample for the deep web [13]. As claimed by the author, this technique is the first large scale evaluation of sampling the deep web. ODP dataset and assessment parameters such as coverage ratio, unique tuples were used for evaluation. Wang et al. recommended a sampling method to learn queries from a set of sample documents [14]. A learnable query set covered a significant region of the deep web. Kumar et al. proposed an algorithm based on learning automata to explore the deep web pages [15]. Dmoz.org website, a replica of ODP, was used as a dataset. Assessment parameters such as precision-recall curve and depth of crawling were used to evaluate the algorithm.

## 2.2 RL and A3C Based Techniques

Nowadays, RL techniques are extensively used in information retrieval domains. RL uses Markov Decision Process (MDP) where action probabilities are unknown. MDP is the combination of 4 tuples (S, A, P, R), whereas 'S' is the set of states $\{s_{1t}, s_{2t}, s_{3t} \ldots s_{kt}\}$, 'A' is the set of actions $\{a_{1t}, a_{2t}, a_{3t} \ldots a_{kt}\}$, 'P' (s'/s, a) is the probability of reaching new state s' on receiving an action 'a' at state s and 'R' is the reward produced by the environment from current state s to new state s' by using some action. If an action helps to achieve the target, the reward is assigned, and policy is learned through experience. If an action takes away from the target, the penalty is assigned, and similar action is discarded in the future. Zhou et al. presented a technique based on the deep Q network for searching the content in social media platforms [16]. This method was used in microblog platforms for security topics. Liu et al. proposed a fast learning method based on Workflow Guided Exploration (WGE) [17]. WGE is useful in sparse reward domains. Learning was achieved through the demonstrations, which is the sequence of state-action pairs. Kumar et al. used an algorithm based on Distributed Learning Automata (DLA) to discover the hidden web pages [18]. Ortiz et al. suggested a deep RL technique to find a subquery generation technique [19]. Shi et al. developed the RL platform to perform the web task with the mouse, keyboard actions, domain object model, and suggested the world of bit platform to do the internet tasks using RL [20]. This technique did not handle the complex queries. In the deep web, the 'Q' value estimation method was recommended to deal with the myopia problem to some extent [21]. Myopia problem arises when the future reward of each query is ignored. Singh et al. proposed a framework based on an intelligent agent to find more relevant information in the deep web [22]. A3C is also a type RL technique that estimates the value function and policy function to learn intelligently. In literature, there are various applications of A3C learning techniques. Some of them are used in Atari games [4], Motion planning such as car racing simulator [4], 3D maze games [4], Rogue video games [23], controlling the traffic signals [24], WGE [17], a vehicular network for efficient resource allocation strategy [25], etc. Yang et al. proposed an enhanced algorithm derived from A3C to tackle the traffic signal control challenges in the multi-intersection [24]. This algorithm proposed a flexible matrix to design an effective policy for traffic control signals. A3C technique is also used in the information extraction domain with the help of parallel agents [26]. Asperti et al. suggested a technique based on A3C to solve the exploration of Rogue dungeon games [23]. The problem was considered a partially observable MDP and partitioned the sample space into situations. Situations were assigned to the agents to solve it. This technique achieved a 98% success rate with ample state space and action space. A3C has been proved successful in dealing with large state space and action space problems.

## 3 Proposed Architecture

This section proposes an architecture diagram for deep web crawling. It is a combination of the A3C and WGE technique. This section also discusses the sample of demonstration and workflow lattice. A3C is a type of policy gradient RL method that parameterize the policy and value function. It has two neural networks that help to learn the policy and value function. It has various agents and one coordinator. The coordinator initializes, controls, and coordinates the working of agents. Agents interact with their corresponding environment, performs an action '$\mathrm{a_j}$' based on the probability that is given by Equation (1). 'Q' is the state transition function that helps to decide the next state based on action '$\mathrm{a_j}$'. Softmax as the activation function that helps to convert into probabilities. Action selection is made using softmax function. Probability of action '$\mathrm{a_j}$' with given state $\mathrm{s_t}$ is represented by

$$\mathrm{P}(\mathrm{a_j}/s_t) = \frac{exp(Q(\mathrm{s_t}, a_j))}{\sum_{j=1}^{N} exp(Q(\mathrm{s_t}, a_j))} \tag{1}$$

The environment responds either by reward or penalty to the agent. Based on the environment response, the parameters of an agent are adjusted. Asynchronous means the parallel functioning of agents that learn from different environments and leads to more stability of the system. Here, main actor is the policy denoted by $\Pi(a_t/s_t,\theta_\mathrm{p})$ whereas '$\mathrm{a_t}$' is the action taken by agent

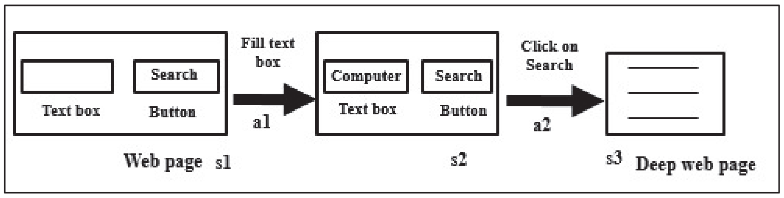

**Figure 1** Sample of demonstration ‘d’ with state-action pair.

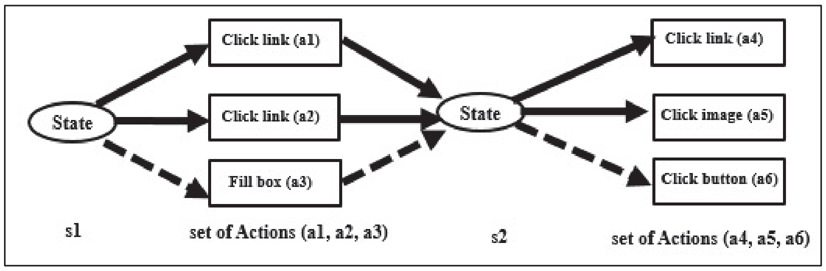

**Figure 2** Sample of workflow lattice with selected path {s1, a3, s2, a6}.

at a time ‘t’ based on $\mathrm{s_t}$, $\mathrm{s_t}$ is the state corresponding to an agent, and ‘$\theta_\mathrm{p}$’ is the policy learning parameter. The critic is the value function denoted by $V(s_t, \theta_\mathrm{v})$ wherein ‘$\theta_\mathrm{v}$’ is the value learning parameter. Deep web problems can be devised as a multi-agent task that uses RL. Here, agents work in parallel, accomplish various actions, receive the response from the environment, and learn the policies. Figure 1 shows the demonstration, i.e., a generalized sequence of state-action pairs. It tells the high level of information, e.g., fill the search form, click on the button, and save the resultant webpage. It does not tell about specific details of the text box and the submit button. It just guides the direction of workflow during crawling. It helps to identify the relevant actions that are similar to the demonstrations so that reward can be generated, and learning does not stagnate.

The selection of demonstration ‘d’ is done by matching with the policy goal ‘g’ and demonstration goal ‘$\mathrm{g_d}$’ as shown in Equation (2):

$$\mathrm{d} \sim \mathrm{P(d/g)}\, \alpha \exp\left(\mathrm{sim}\ (\mathrm{g}, \mathrm{g_d})\right) \tag{2}$$

whereas $\mathrm{sim}\ (\mathrm{g}{,}\mathrm{g_d})$ is the similarity between policy goal ‘g’ and demonstration goal ‘$\mathrm{g_d}$’. Its value is equal to 1 if fundamental goals are same else, this value is $-\infty$. It helps the formulation of workflow lattice. Figure 2 shows the sample of workflow lattice made from the demonstrations. Dotted lines

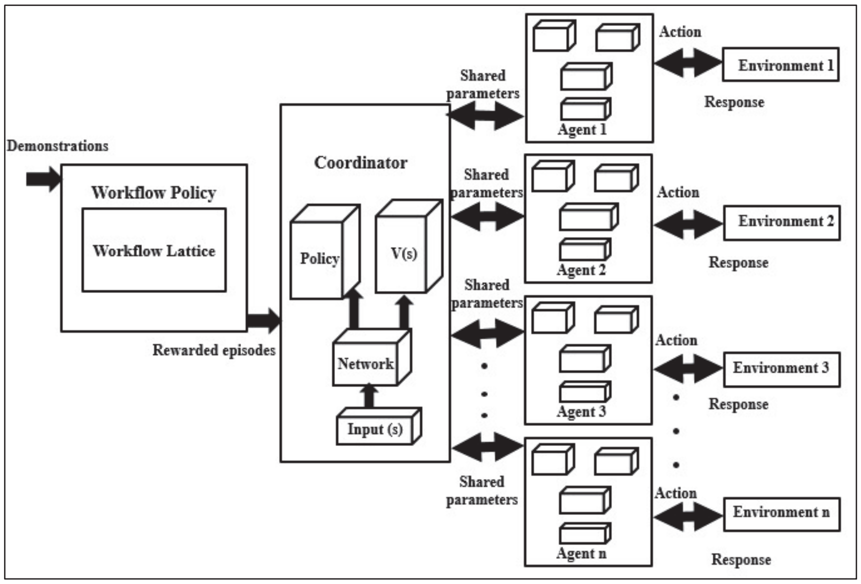

**Figure 3** Proposed architecture diagram.

represent the selected path {s1, a3, s2, a6}. Workflow lattice further builds the workflow policy based on the reinforce algorithm [27].

Figure 3 shows the proposed architecture diagram to learn the policy and its parameters to find the unexplored deep web pages. As there are large number of actions and very few actions lead to reward. So, WGE can be beneficial because it handles sparse reward very efficiently. The selection of those actions that lead to reward is crucial otherwise, agent learning stagnates. Demonstration is an input to the workflow module and helps to generate rewarded episode. This workflow policy is independent of the environment as there is no parameter of state 's'. It is represented by $\Pi_{\mathrm{w}}(z/d\,t)$ whereas z is a workflow step with fewer parameters. It learns more quickly and generates the episodes. Episodes are the state-action pairs that produce either reward or penalty. This policy cannot solve the given problem independently due to environment blindness. However, it helps the coordinator to solve the task and learns quickly. WGE generates the rewarded episodes, which are transferred to the coordinator. The coordinator assigns the rewarded episodes to various agents that guide them to select the action. Agents interact with the environment through action. Environment in turn gives the response to agent-based on action. Agents update their parameters and send the shared parameter to coordinator. Description of the algorithm and mapping of various tuples of deep web are presented in the Section 4.

## 4 Proposed Algorithm

In this section, a technique, namely Deep Web Asynchronous Advantage Actor-Critic (DW-A3C), is proposed to retrieve the deep web pages. It is a type of RL technique that helps to acquire the policy and value function. This policy helps to generate reward by searching unidentified deep web pages.

The DW-A3C is defined with 12 tuples (I, S, A, $\pounds$, H, Q, V, Ad, $\Omega$, $\gamma$, $\Pi$, and $E_H$). These tuples are explained below:

i. I = $\{I_1, I_2, I_3 \ldots I_n\}$ is the set of agents that have deep learning neural networks to learn the local parameters for policy formulation. Here, ‘n’ is the number of agents.

ii. S = $\{s_{1t}, s_{2t}, s_{3t} \ldots s_{kt}\}$ is the set of states/web pages for an agent at a given time ‘t,’ and it can be PIW or deep web pages.

iii. A = $\{a_{1t}, a_{2t}, a_{3t} \ldots a_{kt}\}$ is the set of actions available on a given webpage ‘$s_{kt}$’ for anagent at time ‘t’.

iv. ‘$\pounds$’ is the seed URL of a website that is the starting state. Each seed URL belongs to the state of a website $\pounds$ ∈S. Every seed URL is a subset of the set ‘S’, i.e., $\pounds \subseteq$ S. It can be equal to S for a single-page website.

v. ‘H’ is the set of deep web pages which is a proper subset of S (H $\subset$ S). It is always a proper subset because every website has a seed URL ‘$\pounds$’. ‘H’ can be an empty set or ‘H’ $= \phi$ because some websites may not have deep web pages ‘H’.

vi. ‘Q’ is the state transition function that helps to decide the next state based on action ‘$a_{kt}$’. It can be defined as $S_t \times A \rightarrow S_{t+1}$. ‘$S_t$’ is the present state and ‘$S_{t+1}$’ is the next state after an action ‘$a_{kt}$’.

vii. V($s_{it}$) is the value function which is estimated return for the subsequent policy $\Pi$ from state ‘$s_{it}$’. It captures how good an action ‘$a_{it}$’ is to be at this state.

viii. ‘$A_d$’ is the advantage function that captures how better an action as compared to other actions at $a_{kt}$ with a given state. It is represented by Equation (3):

$$A_d(s, a_{kt}) = Q(s, a_{kt}) - V(s) \tag{3}$$

ix. ‘$\Omega$’ is the response generated by the environment corresponding to a given episode that can be a reward (1) or penalty (0). The episode is the set of pairs consisting of state and action.

x. '$\gamma$' is the discount factor in the range of 0 and 1 $(0 \leq \gamma \leq 1)$.
xi. $\Pi(\mathrm{a_{kt}}/\mathrm{s_{it}})$ is the policy function that decides action based on state '$\mathrm{s_{it}}$'. Here, the policy objective function is $\mathrm{S} - \mathrm{H} \rightarrow \mathrm{H}$ to find the unknown deep web pages from given seed URL '$\pounds$'. $\Pi^*$ is the optimal policy that selects the episodes which have earned maximum reward by selecting suitable actions.
xii. '$E_H$' is entropy used to encourage exploration. The sum of the probability of action multiply the log of the probability of action. It is represented by Equation (4):

$$E_H = -\sum_{k=1}^{K} \Pi_\mathrm{k} \log \Pi_\mathrm{k} \tag{4}$$

Figure 4 shows the flow chart of the DW-A3C algorithm to learn the policy parameter. For each seed URL, LVS table is populated with their corresponding key-value pair [6]. WGE helps to find rewarded episodes that lead to quick learning of the parameter, as shown in Figure 3. The coordinator initializes the agents, passes the rewarded episodes, and assigns seed URLs to each agent. URL queue is maintained to check the status of seed URLs. Agents interact with the environment, learn the local policy, and send local gradient to a coordinator for updating the parameter. The agent starts crawling corresponding to seed URLs. This process stops when the depth of crawling is completed, or the terminal state is reached, or the agent time limit is exceeded. Seed URL is popped out from the URL queue. The coordinator sends a shared gradient to all agents. If the URL queue is empty, then it stops. Else new seed URL is assigned to the agent by the coordinator.

An algorithm is proposed to explore the dynamically generated deep web pages. It has various inputs, i.e., seed URLs, rewarded episodes, and form tag vector. Seed URLs are a collection of starting URLs. Rewarded episodes are a sequence of state-action pairs that lead to reward. Form tag vector provides entry to the deep web and is found by proposed crawler. LVS table contains a key-value pair corresponding to the domain of seed URL. It has a set of deep web pages represented by ¥ as output. A3C architecture has one coordinator and n agents, as shown in Figure 3. The coordinator assigns each agent with different seed URLs. Each agent assigns the probability to each action, as shown in step 2 of the algorithm. The coordinator resets, initializes each agent, and passes the rewarded episodes to each agent, as shown in step 3. Assume shared parameter vector '$\theta$' and '$\theta_\mathrm{v}$', shared counter variables are initialized $\mathrm{t} = 0$, $\mathrm{d}\theta \leftarrow 0$, t=0, and $\mathrm{d}\theta_\mathrm{v} \leftarrow 0$. '$\mathrm{t_w}$' is the maximum agent time. Exit criteria for each agent are time '$\mathrm{t_w}$' or it finds all the deep web

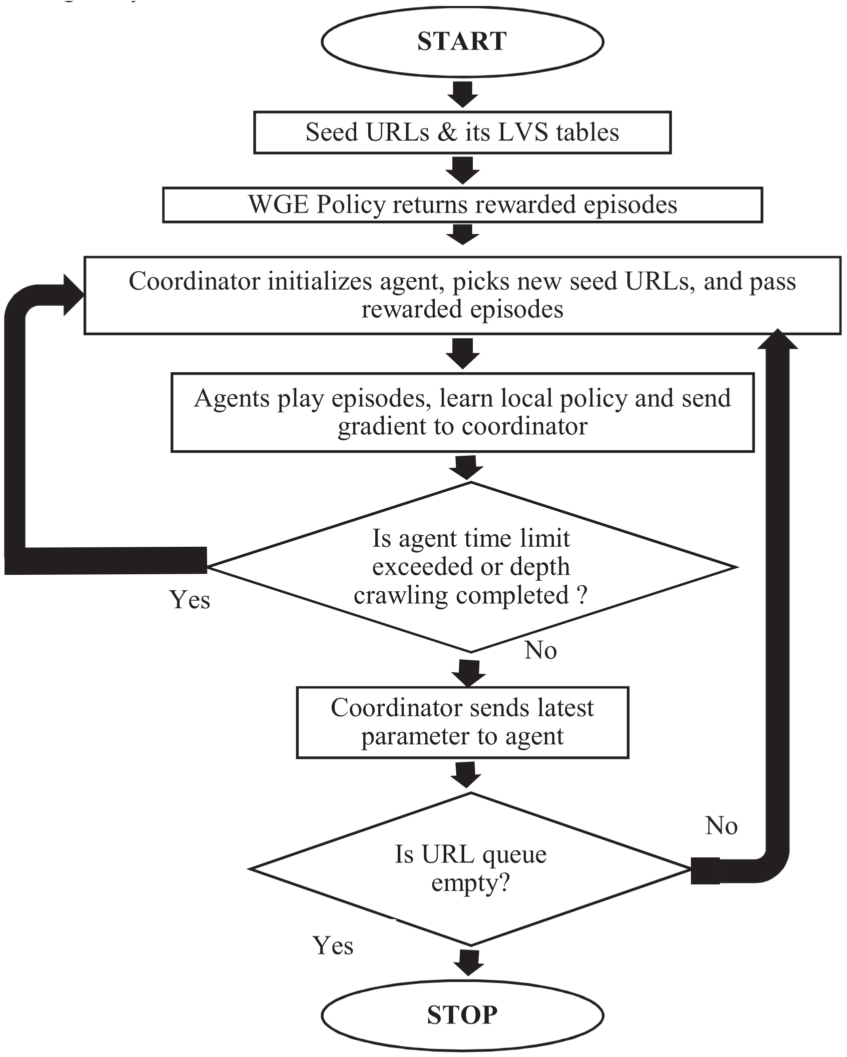
START
Seed URLs & its LVS tables
WGE Policy returns rewarded episodes
Coordinator initializes agent, picks new seed URLs, and pass rewarded episodes
Agents play episodes, learn local policy and send gradient to coordinator
Is agent time limit exceeded or depth crawling completed ?
Yes
No
Coordinator sends latest parameter to agent
Is URL queue empty?
No
Yes
STOP

**Figure 4** Flow chart of DW-A3C algorithm.

pages for the given seed URL as shown in step 6. Embedding technique word2vec converts the rewarded episode and form tag into vector form. Convolution Neural Network helps to learn the features from the rewarded episode vector and the form tag vector, as shown in Figure 5. Policy and value functions are generated from the Long Short Term Memory (LSTM) and fully connected layer. The crawler uses the policy function to find the next action and reaches to the new state '$s_{jt}$'. Reward and penalty criteria are explained in steps 11 to 16 of the algorithm. It updates the action probability, local parameters of the agent, and asynchronous updates the shared counter variable '$\theta$' and '$\theta_v$'. Once the agent time limit is exceeded, the coordinator resets the agent and assigns a new seed URL. If the URL queue is empty, then it stops.

## Algorithm for finding the deep web pages

```
Input: Seed URLs with LVS table values, rewarded episode, and form tag vector.
Output: '¥' is a set of the deep web pages retrieved by crawler. 'H' is a set of deep web
    pages retrieved manually and used for validation purposes. $¥ \subseteq H$.
1:  Create A3C architecture, and each agent is assigned a set of web pages for a given seed
    URL.
2:  Initialize the probability of each action as illustrated in Equation (1).
3:  Coordinator resets, initialize agents, and pass the rewarded episodes.
4:  Let 't' be the time in seconds, 't'=0. '$t_w$' is the agent time limit.
5:  for each agent
6:      while ($¥ = H$ or $t <= t_w$)
7:          word2vec embedding is used to convert rewarded episodes and form tag webpage
            into form vector.
8:          Learn the features from episode vector and form vector through Convolution Neural
            Networks and fully connected layer.
9:          LSTM helps to find the relationship and dependency between rewarded episode and
            web pages. It generates the policy $\Pi(a_{kt}/s_{it}, \theta)$, and value function V(s). Perform
            $a_{kt}$ according to the policy
10:         The crawler uses this policy function $\Pi(a_{kt}/s_{it})$ to find the next action given state
            as defined in DW-A3C.
11:         Increment by time t by 1 unit.
12:         if '$s_{it}$' and '$s_{jt}$' are connected with a dotted line
13:             reward the action '$a_{it}$'.
14:             save the '$s_{it}$' and add it to set 'H'.
15:         else
16:             penalize the action '$a_{it}$'.
17:             update action probability and local parameter
18:         end if
19:         send local parameter and rewarded episodes discovered to the coordinator
20:     end while
21:     Perform update '$\theta$' using '$d\theta$' and '$\theta_v$' using '$d\theta_v$'.
22: end for
```

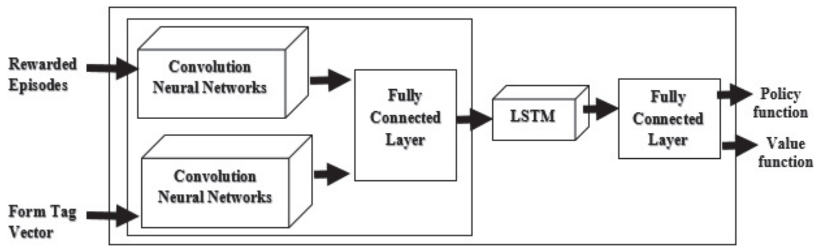

**Figure 5** An agent architecture for learning the policy and value function.

## 5 Experimental Evaluation

All experiments were performed on the Intel Xeon W-2155 and Windows 10 Professional with 64 GB RAM. Various information retrieval metrics were used to validate the proposed technique (DW-A3C). ODP is the most popular open-source subject system used in literature. It is the collection of URLs which have been categorized manually and are not biased by commercial concern. ODP website, i.e., dmoz.org, was closed after March 2017. Since then, the curlie.org domain is used as a dataset to represent the ODP. This dataset contains fifteen domains that were chosen for evaluation. From each domain, one website was randomly chosen. The selected domains were arts, business, computers, games, health, home, news, recreation, reference, regional, science, shopping, society, sports, and kids & teens with the corresponding names as db1, db2, db3, db4, db5, db6, db7, db8, db9, db10, db11, db12, db13, db14, and db15 respectively. The proposed crawler was able to identify various forms, such as search forms, login forms, and subscription forms for a given website. This crawler traversed each website of a domain and crawled both the types of web pages, such as PIW pages and deep web pages. It was developed using the Python language. The main libraries used were selenium web driver, scrapy, regular expressions, urllib2, urljoin, requests, os, enchant, beautiful soup, clipboard, etc.

The LVS table was populated with the corresponding key-value pair entries based on each domain. The proposed crawler used LVS table values to fill the search forms and retrieve the dynamically generated deep web pages. This crawler collected all the web pages, such as PIW and deep web pages. It traversed up to seven depths for a website. Depth seven was selected because very little information of deep web pages was retrieved after depth 7. Manually collected web pages were used to evaluate the crawler performance. A total of 1,52,680 web pages that contain 80,486 deep web pages were collected manually. This manual result helps to validate all new deep web pages retrieved by the proposed crawler. A graph was prepared by connecting the starting seed URL and its corresponding web pages using matplotlib. There were two types of lines in the graph first was the dark line, and the second was dotted line. A dark line connects the two PIW pages or one deep web page followed by a PIW page. A dotted line is connected by two deep web pages or one PIW page followed by a deep web page. This graph was used to calculate various metrics such as average harvest ratio and coverage ratio.

## 5.1 Performance Evaluation Based on Number of Searchable Forms vs. Depth

Deep web pages are generated dynamically after filling the searchable form. Searchable forms are the entry points to the deep web. The proposed crawler was able to identify the searchable forms during crawling. Figure 6 shows the distribution of several searchable forms discovered during the depth of crawling. With an increase in crawling depth, the number of searchable forms also increases and reaches the maximum number of searchable forms at depth d. The value of d for fourteen domains is three except for db3, whose value of d is two. The number of searchable forms starts to decrease for all domains after depth d. As depicted in Figure 6, the number of searchable forms for domains such as db1, db4, db6, db14, and db15 is less than other domains.

## 5.2 Performance Evaluation Based on Fraction of Deep Web Pages with the Depth of Crawling

Figure 7 shows the graphical distribution of deep web pages with the depth of crawling. No deep web page is found at depth 0 because it contains only the seed URL. As the crawler starts to traverse, more searchable forms are discovered that leads to an increase in deep web pages. The fraction of deep web pages is maximum at depth three except for domain db3. Subsequently, a fraction of deep web pages starts decreasing up to depth 7.

## 5.3 Receiver Operating Characteristics

Receiver Operating Characteristics (ROC) curve is the plot between the True Positive Rate (TPR) on the y-axis and False Positive Rate (FPR) on the x-axis. TPR is the ratio of true positive and all actual positive that is the combination of a true positive and false negative. TPR is also named as recall

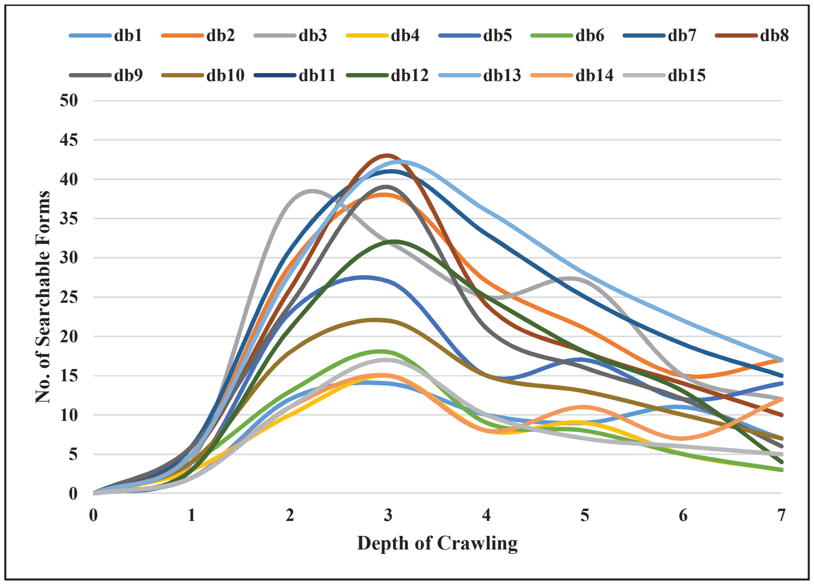

**Figure 6** Distribution of the number of searchable forms w.r.t. depth of crawling.

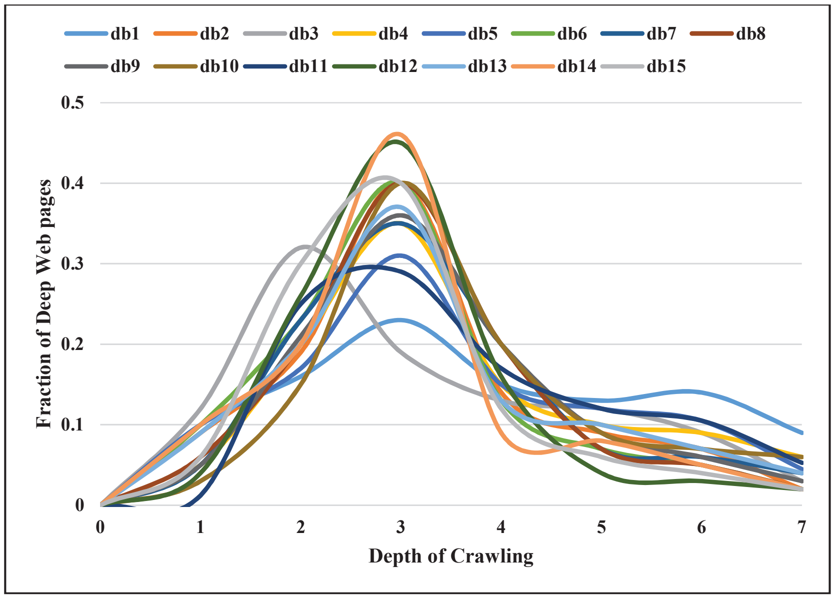

**Figure 7** Fraction of deep web pages with the depth of crawling.

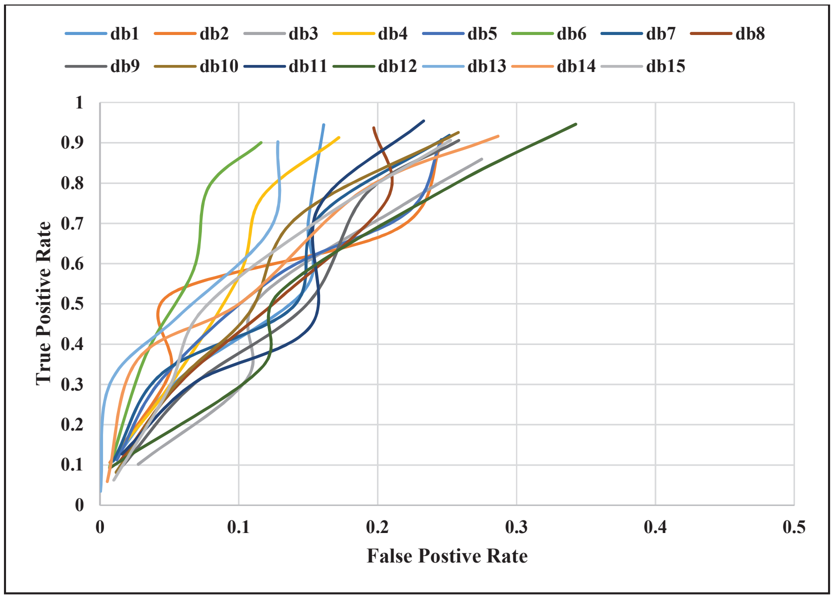

**Figure 8** ROC curve for different domains.

or sensitivity. FPR is the ratio of false-positive and all actual negatives that is the combination of false positives and true negatives. It is desired for ideal scenarios to have point on the top left corner. Figure 8 shows the ROC curve for fifteen domains. Initially, the value of TPR and FPR is minimal because it contains fewer true positives and false positives. As more web pages are crawled, the value of true positive increases rapidly as compared to false positive. The value of FPR increases much rapidly at TPR of 0.5 because more false-positive are added in the queue as compared to low value of recall. This behavior is observed for all the fifteen domains, and the average value of TPR reaches the value of 0.93. The average FPR value is 0.22 for all fifteen domains.

## 5.4 Precision-Recall Curve

The precision-recall curve helps to measure the performance of crawler wherein recall is on the x-axis & precision is on the y-axis. A recall is the fraction of true positive and all actual positive. All actual positive is the combination of true positive, and false negative. Precision is a fraction of true positive and all predicted positive. All predicted positive is combination

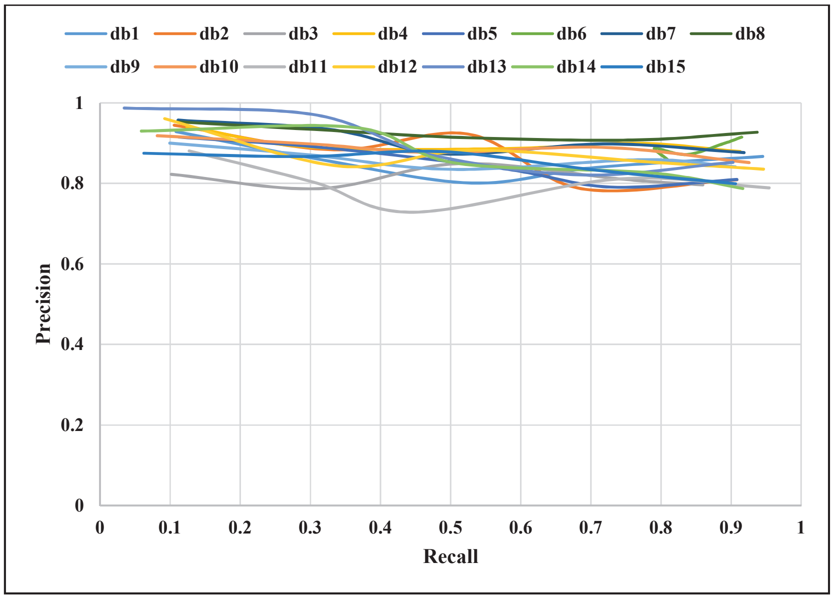

**Figure 9** Precision-Recall curve for different domains.

of true positive, and false positive. As crawling starts, the value of precision is high and the value of recall is low, as shown in Figure 9. As crawling proceeds further, the value of false-positive increases more than true positive. It leads to a decrease in precision value. The value of true positive increases which in turn increases the recall value. Later on, the value of precision increases and then starts to decrease again. It is observed the same for all fifteen domains. The minimum value of precision, i.e., 0.72, is observed at a recall value of 0.44 for domain db11.

### 5.5 Comparative Analysis

This subsection contains the comparative analysis of the proposed technique (DW-A3C) with existing techniques such as DLA [18], Random, Generic frequency, and Adaptive [12]. Various metrics such as deep web pages vs. PIW pages, average harvest rate, and coverage ratio were used for evaluation. DLA technique was chosen for comparison because it also detects the deep web pages. Random, Generic, and adaptive techniques are related to query selection problem of the deep web. ODP dataset was used by all these techniques for evaluation.

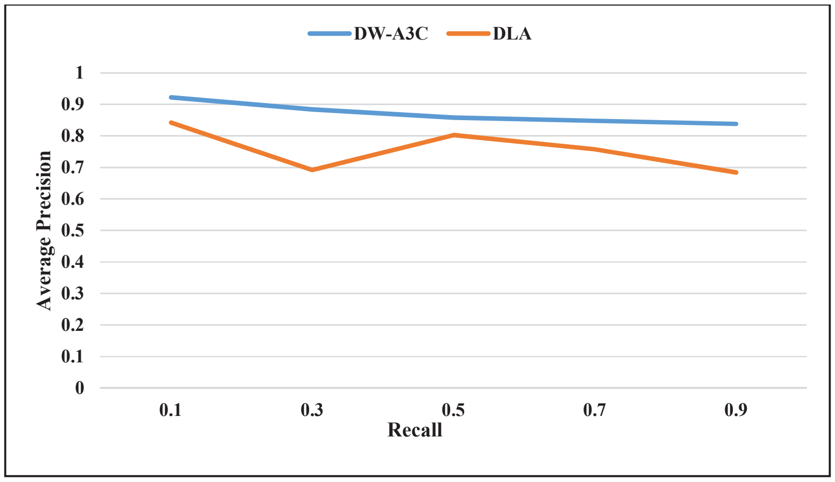

**Figure 10** Average Precision-Recall curve for DW-A3C and DLA.

### 5.5.1 Average precision-recall curve

The average precision-recall metric is used for comparing the proposed technique with the DLA technique. Average precision is the sum of different precision points of various domains divided by the total number of domains n for a given value of recall. Figure 10 shows the average precision-recall curve for DW-A3C and DLA technique [18]. DW-A3C outperforms the DLA technique for all the values of recall. Initially, the value of average precision is high. As the more number of web pages are crawled, average precision starts decreasing with an increased value of recall.

### 5.5.2 Fraction of deep web pages vs. PIW pages

As shown in Table 1 below, the fraction of deep web pages retrieved from DW-A3C is more than DLA. As the number of deep web pages is more, this small difference leads to the large number of web pages. The difference between DW-A3C and DLA in terms of fraction of PIW pages is more significant. DLA crawls all PIW pages, whereas DW-A3C explores less fraction of PIW pages because DW-A3C is based on WGE that explores reward paths rather than all paths.

### 5.5.3 Average Harvest ratio

The average harvest ratio is the fraction of deep web pages retrieved, and all the crawled web pages retrieved containing PIW and deep web pages.

**Table 1** Comparison of proposed technique with DLA

| | Fraction of Deep Web Pages | Fraction of PIW Pages |
|---|---|---|
| **DLA** | 0.90 | 1 |
| **DW-A3C** | 0.94 | 0.72 |

**Table 2** Comparison of different techniques with proposed based on coverage ratio

| Techniques | Random | Generic | DLA | DW-A3C | Adaptive |
|---|---|---|---|---|---|
| Coverage ratio | 0.26 | 0.72 | 0.83 | 0.93 | 0.99 |

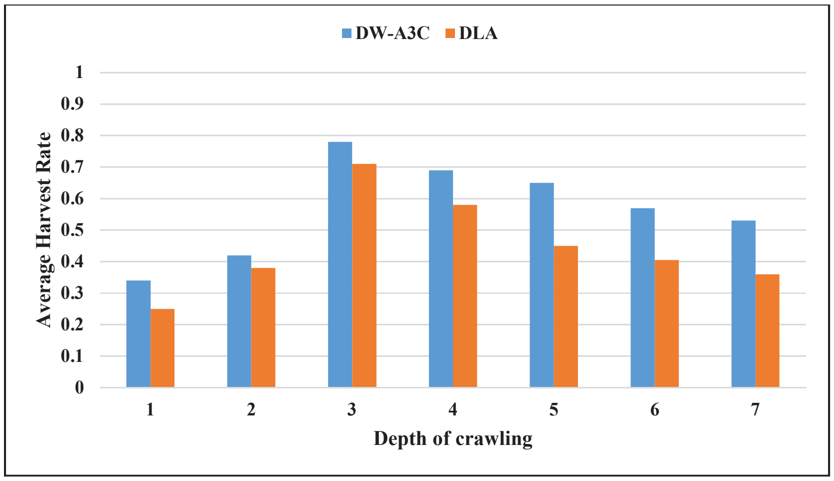

**Figure 11** Average harvest rate versus depth of crawling for ODP.

$H_{avg}$ = ¥d/Sd, whereas $H_{\text{avg}}$ is the average harvest ratio, '¥d' is the deep web pages retrieved during depth 'd' and 'Sd' is the total web pages retrieved during depth 'd'. Figure 11 shows a graph of the average harvest ratio with crawling depth. The increase in the average harvest rate is due to the asynchronous nature of DW-A3C that makes the architecture more stable and uses WGE to find the reward path. DW-A3C outperforms as compared to DLA during all the depths of crawling.

### 5.5.4 Coverage ratio

Figure 12 shows the coverage ratio of documents w.r.t query number for the art section of ODP. The adaptive technique is based on a query generation problem that is a part of deep web crawling [12]. Whereas DW-A3C, a proposed technique, contains the whole deep web crawling process. The

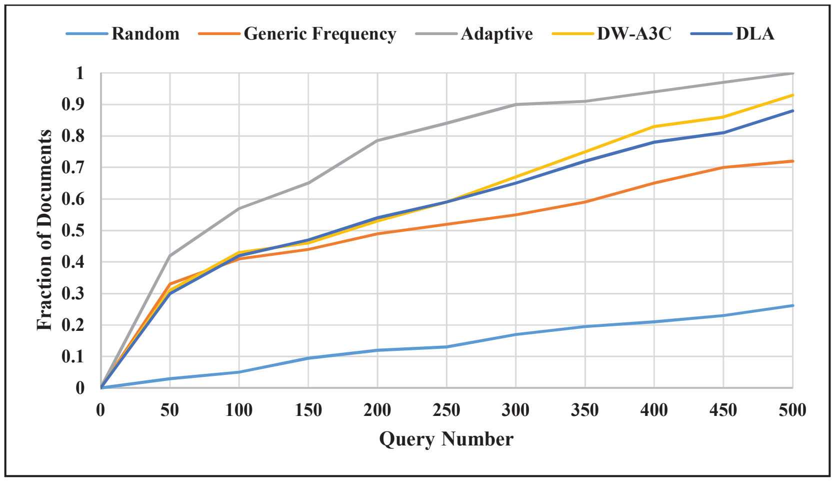

**Figure 12** Coverage ratio of documents for the art section of ODP.

query generation part is compared with the adaptive technique, Generic frequency, and DLA [18]. From Table 2, it is clear that the coverage ratio of DW-A3C is 0.93, which outperforms the coverage ratio of 0.83 in DLA, 0.72 in generic technique, and 0.26 in random technique. However, the coverage ratio of adaptive technique is 0.99, which is more as compared to DW-A3C because of the adaptive technique ability to find new queries from the previous queries.

In our work, coverage ratio and precision depend on initial keywords selected to represent the domain. The current achieved coverage ratio of 0.93 is quite promising. This work can be further improved with an automated and efficient way of selecting keywords for domain representation.

## 6 Conclusion and Future Scope

Crawling the deep web is a challenging task as structured information is hidden behind the search forms. Different types of RL techniques have been applied to extract the deep web data. RL technique namely A3C, has not been applied for deep web crawling. In this paper, a new technique has been proposed to utilize A3C for crawling the deep web and is termed as Deep Web Asynchronous Advantage Actor-Critic (DW-A3C) technique. This proposed technique was validated on fifteen domains of ODP. Various metrics such as the number of searchable forms vs. depth, a fraction of deep web pages vs. depth, and ROC curve were used to assess the performance of the proposed technique. It is revealed that the average precision-recall curve, number of deep web pages vs. PIW pages, and average harvest rate shows the improvement of DW-A3C over DLA for retrieving the deep web pages. The proposed technique also performed better in terms of coverage ratio than other existing techniques such as Generic frequency, Random, and DLA in the domain of art section of ODP. DW-A3C can further be improved by incorporating query optimization techniques. It will help to further increase the coverage ratio using the lesser number of queries.

## References

[1] M. K. Bergman, "White Paper: The Deep Web: Surfacing Hidden Value," *J. Electron. Publ.*, vol. 7, no. 1, 2001.

[2] I. Hernández, C. R. Rivero, and D. Ruiz, "Deep Web crawling: a survey," *World Wide Web*, vol. 22, no. 4, pp. 1577–1610, Jul. 2019.

[3] M. Kumar, R. Bhatia, and D. Rattan, "A survey of Web crawlers for information retrieval," *WIREs Data Min. Knowl. Discov.*, vol. 7, no. 6, p. e1218, 2017.

[4] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," *33rd Int. Conf. Mach. Learn. ICML 2016*, vol. 4, pp. 2850–2869, 2016.

[5] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang, "Accessing the deep web," *Commun. ACM*, vol. 50, no. 5, pp. 94–101, May 2007.

[6] S. Raghavan and H. Garcia-Molina, "Crawling the Hidden Web," in *27th VLDB Conference - Roma, Italy*, 2001, pp. 1–10.

[7] M. C. Moraes, C. A. Heuser, V. P. Moreira, and D. Barbosa, "Prequery Discovery of Domain-Specific Query Forms: A Survey," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 8, pp. 1830–1848, Aug. 2013.

[8] G. Z. Kantorski, V. P. Moreira, and C. A. Heuser, "Automatic Filling of Hidden Web Forms," *ACM SIGMOD Rec.*, vol. 44, no. 1, pp. 24–35, May 2015.

[9] Y. Ru and E. Horowitz, "Indexing the invisible web: a survey," *Online Inf. Rev.*, vol. 29, no. 3, pp. 249–265, 2005.

[10] J. Madhavan, L. Afanasiev, L. Antova, and A. Halevy, "Harnessing the Deep Web: Present and Future," *Syst. Res.*, vol. 2, no. 2, pp. 50–54, 2009.

[11] J. Madhavan, D. Ko, £. Kot, V. Ganapathy, A. Rasmussen, and A. Halevy, "Google's Deep Web crawl," *Proc. VLDB Endow.*, vol. 1, no. 2, pp. 1241–1252, Aug. 2008.

[12] a. Ntoulas, P. Pzerfos, and J. C. J. Cho, "Downloading textual hidden web content through keyword queries," *Proc. 5th ACM/IEEE-CS Jt. Conf. Digit. Libr. (JCDL '05)*, pp. 100–109, 2005.

[13] P. Barrio and L. Gravano, "Sampling strategies for information extraction over the deep web," *Inf. Process. Manag.*, vol. 53, no. 2, pp. 1339–1351, 2017.

[14] Y. Wang, J. Lu, J. Liang, J. Chen, and J. Liu, "Selecting queries from sample to crawl deep web data sources," *Web Intell. Agent Syst.*, vol. 10, no. 1, pp. 75–88, 2012.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. 1998.

[16] N. Zhou, J. Du, X. Yao, W. Cui, Z. Xue, and M. Liang, "A content search method for security topics in microblog based on deep reinforcement learning," *World Wide Web*, vol. 23, no. 1, pp. 75–101, 2020.

[17] E. Z. Liu, K. Guu, P. Pasupat, T. Shi, and P. Liang, "Reinforcement Learning on Web Interfaces Using Workflow-Guided Exploration," in *6th International Conference on Learning Representations, ICLR 2018 – Conference Track Proceedings*, 2018.

[18] M. Kumar and R. Bhatia, "Hidden Webpages Detection Using Distributed Learning Automata," *J. Web Eng.*, vol. 17, no. 3–4, pp. 270–283, 2018.

[19] J. Ortiz, M. Balazinska, J. Gehrke, and S. S. Keerthi, "Learning State Representations for Query Optimization with Deep Reinforcement Learning," *DEEM'18 Int. Work. Data Manag. End-to-End Mach. Learn.*, 2018.

[20] T. Shi, A. Karpathy, L. Fan, J. Hernandez, and P. Liang, "World of bits: An open-domain platform for web-based agents," in *In Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 4834–4843.

[21] Q. Zheng, Z. Wu, X. Cheng, L. Jiang, and J. Liu, "Learning to crawl deep web," *Inf. Syst.*, vol. 38, no. 6, pp. 801–819, Sep. 2013.

[22] L. Singh and D. K. Sharma, "An architecture for extracting information from hidden web databases using intelligent agent technology through reinforcement learning," in *2013 IEEE Conference on Information and Communication Technologies*, 2013, no. Ict, pp. 292–297.

[23] A. Asperti, D. Cortesi, and F. Sovrano, "Crawling in Rogue's Dungeons with (Partitioned) A3C," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11331 LNCS, 2019, pp. 264–275.

[24] S. Yang, B. Yang, H.-S. Wong, and Z. Kang, “Cooperative traffic signal control using Multi-step return and Off-policy Asynchronous Advantage Actor-Critic Graph algorithm,” *Knowledge-Based Syst.*, vol. 183, p. 104855, 2019.

[25] M. Chen, T. Wang, K. Ota, M. Dong, M. Zhao, and A. Liu, “Intelligent resource allocation management for vehicles network: An A3C learning approach,” *Comput. Commun.*, vol. 151, no. 2019, pp. 485–494, 2020.

[26] A. Sharma, Z. Parekh, and P. Talukdar, “Speeding up reinforcement learning-based information extraction training using asynchronous methods,” in *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, 2017, pp. 2658–2663.

[27] R. J. Willia, “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning,” *Mach. Learn.*, vol. 8, no. 3, pp. 229–256, 1992.

## Biographies

**Kapil Madan** is a Ph.D. student from Department of Computer Science and Engineering at Punjab Engineering College (Deemed to be University), Chandigarh, India. He attended the Thapar Institute of Engineering Technology (Deemed to be University), Patiala, India where he received his M.E. degree in Software Engineering. He received his B.Tech. degree in Computer Engineering from Kurukshetra University, Haryana, India. He has more than 8 years of Teaching and Research experience. His research areas include Information retrieval, Focused crawling, and Reinforcement learning.

**Rajesh Bhatia** is currently working as a Professor in the Department of Computer Science and Engineering at Punjab Engineering College (Deemed to be University), Chandigarh, India. He received his Ph.D. and M.E. degrees in Computer Science Engineering from Thapar Institute of Engineering Technology (Deemed to be University), Patiala, India. He has received B. Tech. degree from Dr. B. Ambedkar Marathwada University, Aurangabad, India. He has more than 25 years of Teaching and Research experience. His research areas include Automated Software Debugging, Semantic Software Clones detection and Automated Test Cases Generation, Information Retrieval, and Search Based Software Engineering. He is also undertaking various Sponsored Research Projects. He has about 85 research publications in various reputed journals and conferences.