CRAWLING AJAX-BASED WEB APPLICATIONS: EVOLUTION AND STATE-OF-THE-ART

***Shah Khalid*[1], *Shah Khusro*[2], *Irfan Ullah*[3]**

[1]School of Computer Science and Communication Engineering, Jiangsu University, China

[2,3]Department of Computer Science, University of Peshawar, 25120, Pakistan

Email: shahkhalid@ujs.edu.cn[1], khusro@uop.edu.pk[2], cs.irfan@uop.edu.pk[3]

DOI: https://doi.org/10.22452/mjcs.vol31no1.3

ABSTRACT

The innovation of AJAX resulted in more responsive, interactive and faster web applications due to the clever amalgamation of JavaScript, HTML, and Cascading Style Sheets (CSS). However, from the user's perspective, this achievement places many challenges before web search engines. One major challenge is due to the complexities in crawling such web applications because multiple states are associated with one uniform resource locator (URL) that cause a mismatch with search model of web search engines, where a web document is uniquely identified by a single unique URL with a single state. Crawling AJAX-based web applications means giving strength and capability to web search engines so that information produced in these highly-interactive web applications is downloaded and indexed. The need here is to investigate the technicalities of AJAX that shatter the metaphor of a web page which the current web search engine utilize during crawling in order to improve the capabilities of web search engines. Although some academic tools have been developed, they produce some false positives which greatly affect the performance of web search engine. We aim to investigate AJAX and AJAX-based web applications as well as the state-of-the-art in crawling these applications along with some prominent issues, challenges and recommendations.

Keywords: ***AJAX, Crawling, Document Object Model (DOM), Information Retrieval*.**

1.0 INTTRODUCTION

The World Wide Web is a giant source of information in which a continuous change occurs in the way of information storage, retrieval and display. [1]. Simple HTML pages are now being replaced with AJAX-embedded web pages making information retrieval (IR) challenging because of complexities in executing JavaScript, constructing the navigation model and analysis of Document Object Model (DOM)[2]. AJAX, which is short for Asynchronous JavaScript and XML (Extensible Markup Language), is one of the prominent new techniques that are being used to develop rich and more interactive web applications such as Facebook, YouTube and Google Maps. Unlike a new programming, scripting language or technology, AJAX is a new way to think, design and develop web applications [3, 4]. As content is dynamically and asynchronously produced in AJAX-based web applications, web crawlers are unable to detect AJAX event and execute calls just like humans do using a web browser. Furthermore, a lot of applications on the Web are AJAX-based and are least searchable. A methodology is needed to present content produced by these applications to crawlers for indexing purposes just like in traditional Web IR.

A lot of research has already explored the technical aspects of AJAX, challenges and the benefits it provides to web application developers. However, these articles are limited in evaluating and comparing the relative performance of AJAX web crawlers. In this review article, we critically and analytically review the available literature in order to report the state-of-the-art in crawling AJAX-based web applications along with some prominent issues and challenges. We also briefly discuss the nature of AJAX-based web applications and their differences from traditional web applications. We also cover the similarities and differences between traditional web crawlers and existing AJAX crawlers as well as highlight the limitation in AJAX crawlers. For this purpose, we searched the major Computer Science digital libraries and other related databases to collect relevant articles that best describe this new technology. We carefully reviewed and analyzed all the selected articles and reported the state-of-the-art accordingly. We hope that this research paper will open new research avenues for researchers interested in this domain. Rest of the paper is organized as follows: Section 2 presents the journey from conventional web applications to AJAX-based web applications focusing on the philosophy behind AJAX, the underlying differences of AJAX-based web applications from traditional web applications and some merits and demerits of AJAX-based web applications. Section 2 also contributes an evaluation framework for comparing traditional web application model with AJAX-based web application model. Section 3 presents crawling traditional as well as AJAX-based web applications along with its importance, the behavior of these applications towards crawlers and an evaluation framework for comparing different AJAX crawlers.

2.0 FROM WEB APPLICATIONS TO AJAX-BASED WEB APPLICATIONS

In the beginning of the Web in 1990s, web pages were static and updating a portion of the page required reloading the whole page from the server, resulting in an increase in refresh ratio as well as greater consumption of the bandwidth. This was a great impediment, and researchers started thinking of making web applications faster and more productive just like desktop applications. To overcome these problems, AJAX was introduced in 2005 and standardized by World Wide Web Consortium (W3C) in 2006 [4]. Because of its rich interactivity with client-side, AJAX attracted developers and researchers in a very short time interval and resulted in an explosion of questions about how to use AJAX in web applications [4]. Based on studies presented in [4, 5], we can plot the historical timeline of AJAX as shown in Fig. 1. Most of the modern web applications have started using AJAX as it provides very fast asynchronous communication with the client-side. Today, many popular companies are using AJAX e.g., Google, Yahoo and Microsoft in their applications.

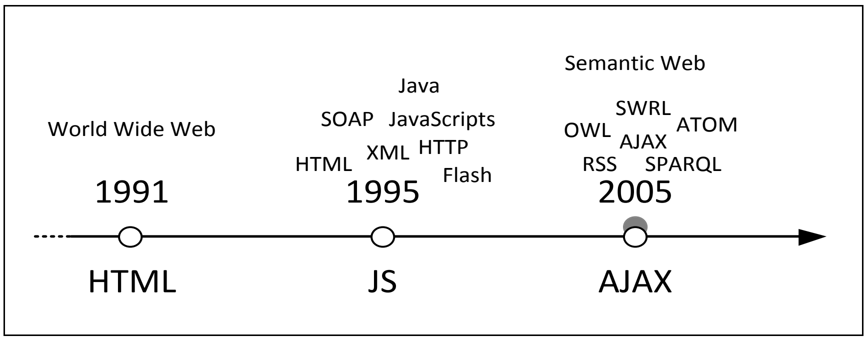

Fig. 1: History of AJAX

AJAX is neither a new programming nor new scripting language, rather, it is a new way of thinking to use existing technologies in producing web applications that are better, faster and more interactive [3, 4]. AJAX uses the existing technologies, languages, levels and protocols in a new way to gain rich interaction with client-side that is very analogous to desktop applications [4, 6]. These well known, mature, stable and widely used tested technologies work together in different levels for creating a new and more powerful web application development model in order to overcome the problems of request-wait and response-wait patterns. These technologies include markup languages including HTML, XHTML and XML. Presentation languages including CSS and DOM for display and update. Client-side scripting languages such as JavaScript, XMLHttpRequest object for asynchronous communication with server. XML, XSLT and JSON for data exchange and JSP, JSF, Perl, Ruby, PHP and ASP for server-side scripting. DOM is a tree data structure that is assembled at the client-side following the hierarchal construction of corresponding web page [7].

AJAX is not tied to a specific markup, scripting, data exchange format or a specific server-side scripting language, rather, an AJAX application can be developed using any combination of tools, e.g., instead of JavaScript, one can use VB Script, and instead of XML one can use JSON and so on. This way of re-using available technologies makes AJAX distinctive on the Web. AJAX allows web applications to make asynchronous connections with the web server i.e., in AJAX-based web applications, clients do not have to wait for a server response after sending request. This enables AJAX-based web applications to give users a more responsive experience. AJAX makes it possible for a part of a web page to be updated, deleted and created independently.

2.1 AJAX-Based Web Applications Model

AJAX-based web applications also called Rich Internet Applications (RIAs)[8] are a new breed of highly interactive and highly dynamic web applications [9]. The client-side part of AJAX-based web application is composed of three layers. These layers include user interface, JavaScript code and AJAX engine. On the user interface, the HTML, CSS and DOM work together. HTML and CSS provide format and structural attributes to the interface being processed as a request to the server for output, and DOM offers means for dynamically accessing and updating content, structure, style, and other elements of the web page. The second layer where JavaScript code resides and records all events generated by users through EventListener in order to provide input to the user interface for presentation to users and increase web application interactivity and responsiveness [10]. The third layer, which comprises of AJAX engine, acts as a middle layer between server and client. AJAX engine makes the asynchronous communications between server and client using XHR object i.e. EventListener sends data to XHR object [11]. XMLHttpRequest is the core of AJAX which enables AJAX engine to receive new data from the server without reloading the whole page [12]. The XHR object asynchronously requests the server through HTTP or HTTPS protocol. The server processes the request accordingly and sends the result back to XHR object in order to update DOM. Some of the processing is performed on server-side whereas some data verification and processing is carried out on the client-side. Only that data is requested through XHR object which is needed from the server [11]. While in traditional web application modal, the whole page is requested from the server because there is no processing related to the application logic on client-side. AJAX-based web applications offer several advantages over classical web applications but we should not over estimate it because in AJAX-based web application, the data is JavaScript-driven, created on the fly which is not SEO-friendly [13].

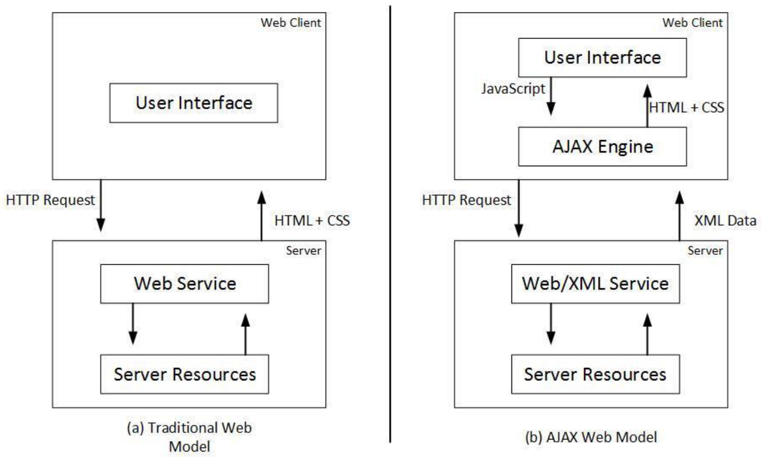

Fig.2: AJAX interaction with server: (a) Traditional Web Model vs. (b) AJAX Web Model [11]

AJAX allows web applications to make asynchronous connection with the server where users need not wait for a response from the server but are able to utilize resources and work in the same page during request (as shown in Fig. 2). At the client-side, a user generates an event that calls the desired JavaScript function, which creates and configures an XMLHttpRequest object. This object calls the server in an asynchronous way which processes the request and returns results in the form of XML document. In response, the XMLHttpRequest object calls the call-back function, checks out that the server has printed the results and keeps these in variables [12]. The DOM is updated accordingly with the new response and AJAX process ends.

2.2 Traditional vs. AJAX-based Web Application– An Evaluation Framework

In traditional web application, (1) the user sets up an HTTP request to the web server on the user interface. (2) In response, the web server processes the request upon receiving data, crunching numbers and communicating with different legacy systems, and responds back to the user with an HTML page [14]. This is not a fruitful approach because user is waiting for response and thus resources are wasted. This start-stop-start-stop nature of traditional approach has some shortcomings including: (1) overall execution is performed by server, (2) repeated transmission of similar data, and (3) slow retrieval due to requesting whole page. In contrast, AJAX abolished this start-stop nature of client-server interaction using an intermediate AJAX engine between client and server. This intermediate layer allows client-server communication to be carried out asynchronously in background for giving immediate answer to the client-side [4, 15]. Thus AJAX updates only specific components and allows direct interaction with the browser resulting in more interactive, responsive and faster user interfaces which ultimately reduces the overload on server side and the waiting time [16]. Table 1 presents an evaluation framework for comparing traditional web applications with AJAX-based web applications using several evaluation criteria.

Table 1: A comparison of traditional Web Application with AJAX-based Web Applications

Models / ***Properties***	***AJAX-based Web Application***	***Traditional Web Application***
Full Page Refresh	No	Yes
Stateful	No	Yes
Start-Stop-Start-Stop	No	Yes
Response	Fast	Slow
Connection	Asynchronously	Synchronously
Servers	Converting data to HTML	Need no conversion
Development	Complex	Easy
Network Performance	High	Low
Visibility to Search Engine	Low	High
Traffic to & from the Server	Low	High
Rich Assets	Interaction-Based	Text-Based
Development Time	Max	Min
Security	Poor-fixed	Well-fixed
Over Load on Server	Low	High

3.0 CRAWLING AJAX-BASED WEB APPLICATIONS

This section investigates crawling of conventional as well as AJAX-based web applications, importance of crawling AJAX-based web applications, AJAX-based web crawlers and behavior of AJAX-based web applications towards crawlers. An evaluation framework for comparing different crawlers used for crawling of AJAX-based web applications is also contributed.

3.1 Crawling Conventional Web Applications

Crawlers are the heart of web search engines [17] . They are programs designed specifically for searching the Web and downloading web pages that are subsequently indexed and searched against user search queries [18]. In addition to web search engines, [19] crawlers are also used in mining the Web, comparison shopping engines and other applications by processing large number of pages [20]. After reaching a particular website, the crawler first requests robot.txt file to know whether it is allowed by the site owner to download web pages and other content. The information such as which portions of the site are allowed to visit and how many times the crawler can visit the site is also present in robot.txt file [21]. It is important for a site to have robot.txt file, otherwise, the crawler assumes that indexing any content of the site is allowed. After reading robot.txt, the crawler recursively explores the website as (i) fetch a page (ii) parse it to extract all linked URLs (iii) for all URLs not seen before, repeat steps (i) and (ii) [14]. In other words, crawling is the process of traversing the Web automatically by retrieving a web page and then recursively retrieving all web pages [22]. Examples of well know crawlers include Googlebot from Google, and MSNBot (from MSN). Web crawling has been the focus many research articles including [17, 23-28], which are worth reading.

In spite of their simplicity, web crawlers have problems and challenges regarding the engineering of web applications such as scalability of the Web and technological advancements [18, 20]. Some of these challenges include[1]: how many pages they can crawl without becoming bogged down? How speedily they can crawl pages

[1] http://www.semclubhouse.com/advances-in-crawling-the-web

without overwhelming the sites that they visit? How many resources do they have to utilize for crawling and repeating the process? Most importantly, crawling dynamic resources such as AJAX-based content systematically and efficiently is a big challenge. Therefore, web crawlers should have a module capable for addressing these challenges to cope with advancement in technologies and information structure.

3.2 Crawling AJAX-based Web Applications

Although the introduction of AJAX-based web applications to the Web results in increased interactivity and responsiveness, it creates some problems for traditional web crawlers in crawling them. To the best of our knowledge, there is no tool developed so far to address the asynchronous communication of AJAX-based web application with the server, in which one URL have several page states. This is noticeably a main problem that must be covered because much of the applications on the Web are AJAX-based. An up-to-date crawler only goes out and collects pages from the Web, it neither ranks nor crawls AJAX content but only provides extracted URLs to search engine for indexing and ranking [29].

Traditionally, it is difficult for search engine to process AJAX content because AJAX content is produced dynamically and asynchronously by browser actions and thus not visible to crawlers. Search engines like Google either ignore AJAX-based web applications or produce false negatives [30]. Because there is no module in current crawlers to understand and detect AJAX events and execute AJAX calls just like human do using a web browser, the result is the non-availability of crawlers that can crawl AJAX-based web applications both systematically and efficiently.

3.2.1 AJAX Web Crawlers

In last 18 years AJAX-based web applications crawlers made significant progress in the domain of web search engines [26, 29-33]. In this way, several papers have been published (described in current state-of-the-art). For example, [34, 35] focus on crawling strategies, [29, 30] focus on searching and crawling AJAX-based web applications, [36] focus on making AJAX-based web applications accessible to search engine. This facilitated Information Retrieval, however the research is still not sufficient in reporting crawling AJAX-based web applications and we are trying to address the fundamental problem of crawling [15].

3.2.3 Importance of Crawling Ajax-based Web Applications

Much of the information on the Web is AJAX-based and can play an imperative role in the effectiveness of the Web. Consider comments on YouTube and social blogs etc., which are AJAX-based and can be loaded from the server without refreshing the complete page and the altering of URL. YouTube[2] is a website that includes both user view and AJAX parts. Fig. 3 shows a typical YouTube page consisting of both user view and AJAX part, which can better explore the benefits of crawling AJAX- part of this web application along with other content.

In the first page by default, comments are displayed below the video in comment box. The comments are uploaded seamlessly from the server when requested without changing the URL [29]. The available web search engine cannot index such content [30], because they cannot understand the logic on the client-side i.e. for each click, the desired JavaScript function is triggered. AJAX call is made to the server, which seamlessly uploads the content to the same area, leaving the rest of the page untouched. For instance, consider the particular President Obama's speech called "message for America's students" which is addressed with information (President Obama's message for America's students) included in the title. By using traditional search, one can only find this speech by entering the title information in the search query. An interested user may not know all information needed for finding this video; they may either know the country name or other information which is not available to the search engine. This can only be possible if AJAX content becomes useful for search engine. Thus, by crawling AJAX content, one can easily find out this speech by entering any of the relevant information. Furthermore, crawling provides basis for indexing which is important for searching. Also, more and more of the applications on the Web are becoming AJAX-based because these modern applications are user friendly and provide an enhanced experience to end user [31]. Therefore, crawling AJAX-based Web applications is indispensable.

[2] http://www.youtube.com

3.2.3 The Behavior of AJAX-based Web Applications towards Crawlers

Today, most of the Web is occupied with AJAX-based web applications and it has been realized that AJAX-based web applications are faster, richer and more responsive. However, these features are costly because AJAX makes crawlers unable to see the content that is created dynamically.

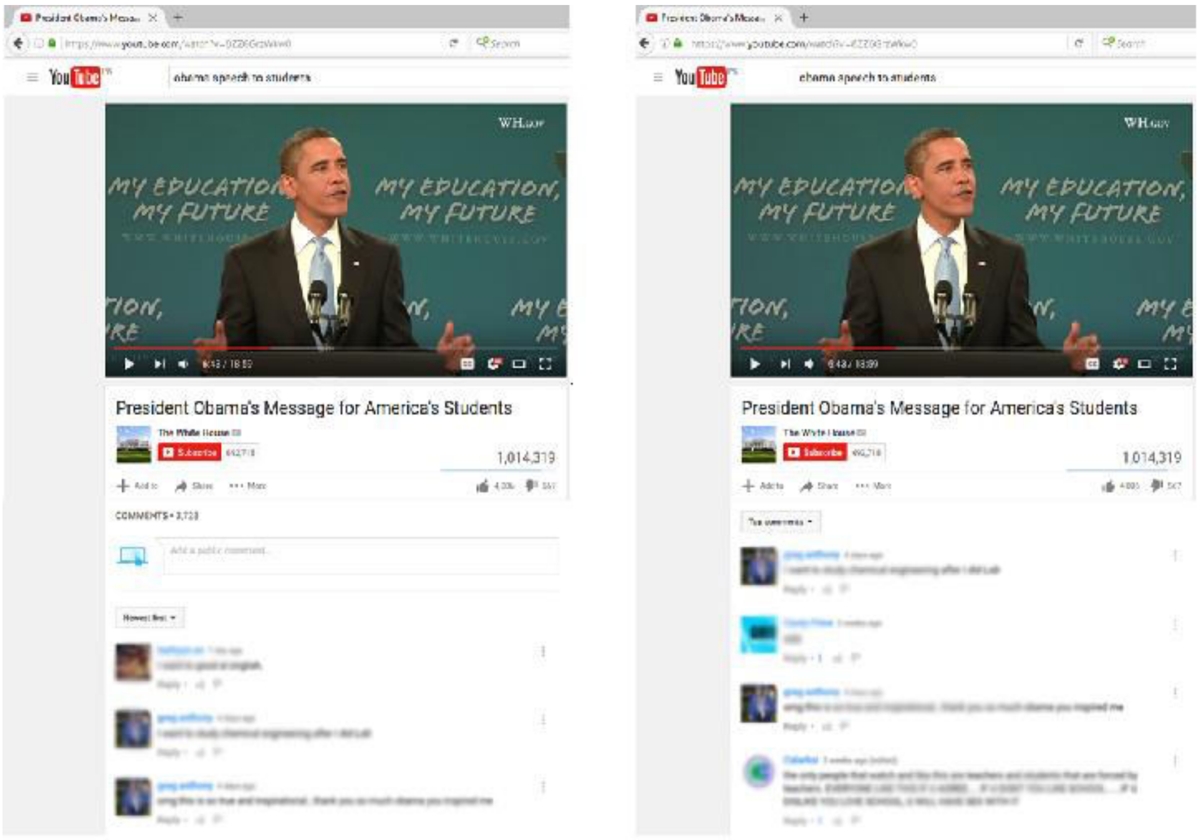
obama speech to students
President Obama's Message for America's Students
1,014,319
COMMENTS • 3,723
President Obama's Message for America's Students
1,014,319

Fig. 3**:** YouTube: Loading AJAX-based comments

Since AJAX-based web application can present all states of the application seamlessly without changing the URL [36], it halts the rule that one URL is for representing and accessing a single unique web page. Resolutely, it means that in AJAX-based web applications different page states may be associated with one single URL which can be seen by the user but not by the crawler.

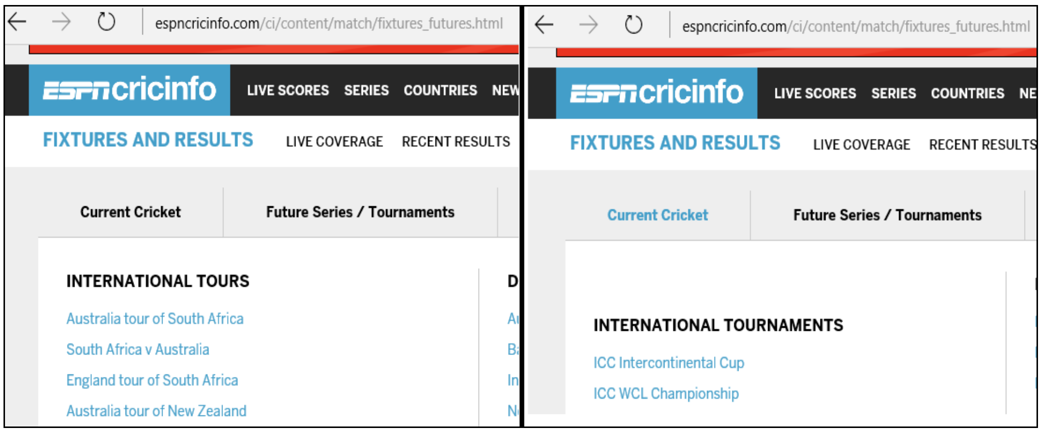

Fig. 4: States of AJAX-based application under the same URL

To explain diagrammatically that AJAX-based web applications are not crawlers friendly, we present the bumpy DOM of cricket website[3] as shown in Fig. 4 and Fig. 5 which clearly show that one URL has several page states which cause a mismatch with up-to-date search model of search engines. Currently no crawler exists that can handle and save complex client-side code (JavaScript etc.) that is existing in AJAX-based web applications for asynchronous communication due to some challenges caused by how AJAX crawling deals with JavaScript execution, construction of navigation model and analysis of DOM. Consequently, the desired state cannot be crawled, indexed and searched resourcefully because the application logic that dynamically loads content is not understood to web search engines [16].

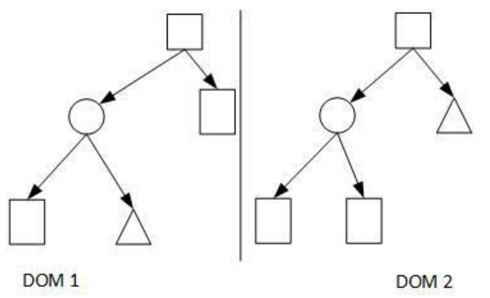

Fig. 5: The states of DOM (DOM1; DOM2) having same URL[4] after granular JavaScript functions execution

3.2.4 Comparison of AJAX Crawlers

We are presenting here a simple evaluation for comparing different AJAX crawlers developed so for. The experimental results obtained while crawling AJAX-based web applications using WSPINX as a traditional crawler are shown in Fig. 6, whereas statistics resulted when crawling AJAX-based web applications using AJAX crawlers are summarized in Fig. 7.

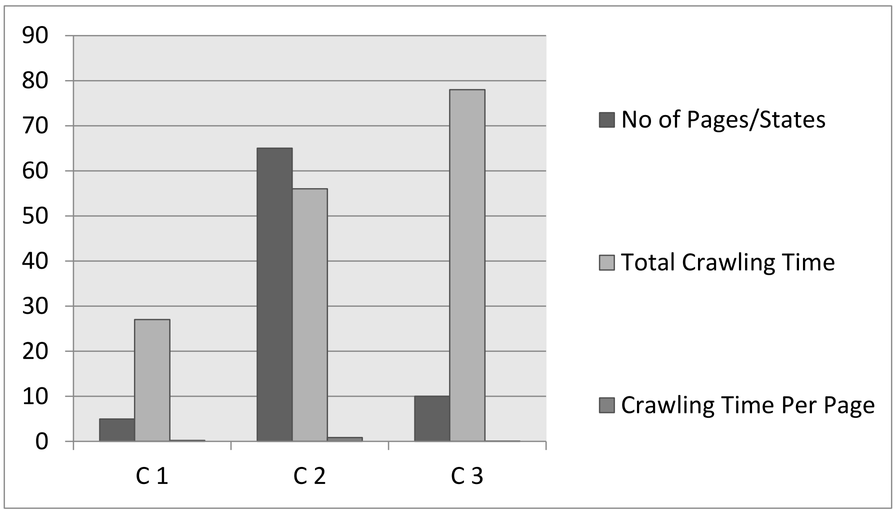

Fig. 6: Results of static crawler

From the evaluation, it is clear that WSPINX takes less time in crawling. However, it is a static crawler and can only crawl static hyperlinks while the rest can crawl dynamic states under the same URL but take more time. Therefore, it can be concluded that static crawler is much better in performance than AJAX crawlers. In case of static web pages however, it is imperative to optimize the technique in order to make search engine able to index useful information of the today rich Internet applications (RIAs). According to literature review that we have carried out, we found no crawler to efficiently and systematically crawl RIAs. As we have discussed and

[3] http://www.espncricinfo.com/,
[4] http://www.espncricinfo.com

examined some academic tools [2, 7, 29-31, 34, 36-38] developed so far for crawling AJAX-based web applications , we deemed them unsatisfactory and unable to fulfill the desired requirements of web search engines. Consequently, crawling AJAX-based web applications is very limited in practice.

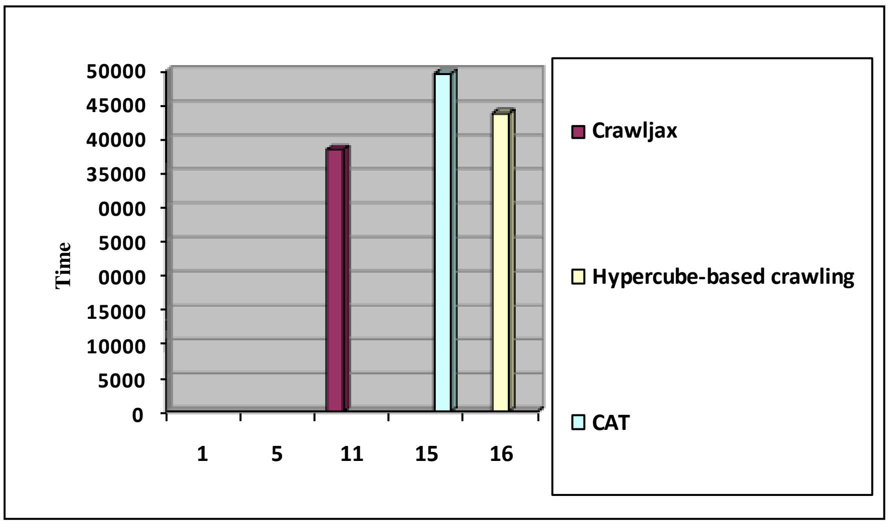

Fig. 7: Number of states discovered by various tools

4.0 THE STATE-OF-THE-ART IN CRAWLING AJAX-BASED WEB APPLICATIONS

With the advancements in technologies, the Web has become a huge repository of informational resources where static content has been replaced by dynamic as well as AJAX-embedded web pages, which is a barrier to information retrieval in crawling and indexing [39] as existing web crawlers are unable to crawl AJAX-based web applications [34]. Currently, search engines are limited in fully understanding the application logic of AJAX-based web applications that load content dynamically because all states of the page are identified by a single URL which is incompatible with the current model of web search engines such as Google, and Yahoo [38].

Although several approaches have been proposed in the literature in order to systematically and efficiently crawl AJAX-based web applications, these approaches are limited as the crawling engines are usually "protocol-driven" (i.e., make a socket connection on the target host). Once the connection is established between client and server, client sends HTTP requests and tries to interpret responses. All parsed resources are then collected and saved for future access [37]. This approach, however, does not work when the crawler comes across an AJAX embedded page because all target resources embedded in the DOM context are the part of JavaScript code which can't be crawled by traditional crawling engine. To understand and trigger this DOM-based activity, an "event-driven" crawling in [37] has been proposed for AJAX-based web applications. An event driven crawling is carried out in three steps: analyzing and interpreting client-side JavaScript; handling DOM; and dispatching DOM model content extraction [40]. It uses modern browsers as underlying platform for accomplishing AJAX crawling with automated information extraction using tools (namely Watir and Crowbar) to extract page data after encountering changes in DOM. This way, it changes the way requests are made and meets some of the requirements of AJAX search crawler using both "protocol-driven" and "event-driven" approaches. However, no automatic state traversal algorithm exists to traverse the web application in detecting all possible events and their desired JavaScript functions like humans do using a web browser [38]. This problem is addressed in [29] by devising AJAXSearch. The architecture of AJAXSearch is shown in Fig. 8, which is an AJAX-aware search engine designed for indexing AJAX-based web applications.

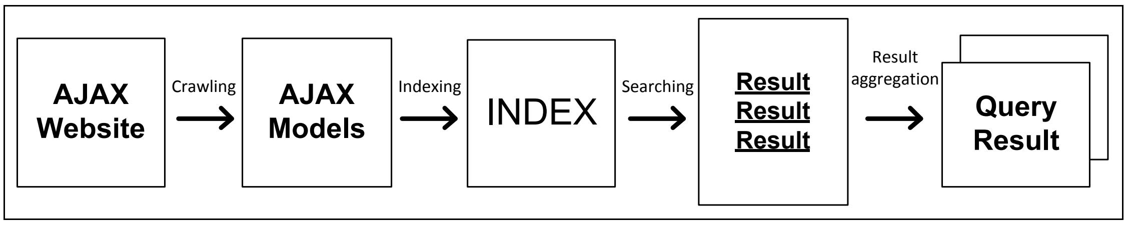

Fig.8: Architecture of AJAXSearch

In AJAXSearch, the crawler goes through the application data for obtaining transition graph that can be used in indexing later by the search engine. Using states and transitions, the crawler creates an application model which is read by the indexer in order to build an inverted index of all AJAX-specific information. The query process is used to handle querying the indexed file for returning application states. The AJAXSearch identifies events in web page and starts crawling using standard breadth-first search. For introducing new states, the approach compares the content of each state and come across DOM with the already-discovered states. During the crawling process, the result of each state is maintained in special application model annotated with latest information. The other components of the architecture also get information from the application model for their desired use. This way, AJAXSearch has resolved several challenges regarding crawling AJAX-based applications. The issue of completeness and closeness of the application model were resolved by setting a maximum depth limit to avoid numerous event invocations. Identical states are identified by comparing the content of new DOM with the existing state where AJAXSearch chooses a new state only when the transition generates a new DOM. For maintaining contextual information, all JavaScript variables are maintained as part of the state during crawling. For personalized content, it uses minimum intervention from user to access the content in an enterprise-only crawling or user-specific crawling [29]. However, the problem of handling duplicate JavaScript function invocations and accessing previous states is problematic.

To avoid duplicate JavaScript function invocations, Matter [30] extends AJAXSearch using the concept of “Hot Node” which also handles access state problem. A hot node is one that fetches content from the server, and a call from such a node to the server is called a hot call. He contributed the modeling of AJAX sites as transition graphs containing all application entities including states, events and transitions based on user events for exploring states. For AJAX crawling, the approach takes into account the events in a web page where crawling starts by triggering all events in a web page using breadth-first crawling algorithm for building AJAX model. Some of challenges solved using this approach include the challenges of infinite state expansion, infinite loops, identical states, irrelevant events, and handling number of AJAX calls. For improving crawling efficiency, this approach uses heuristics that helps in constructing the JavaScript invocation graph so that irrelevant AJAX invocations to the server are avoided. Here, each node in the graph represents a JavaScript function and its dependences. By the use of caching, the AJAXCrawl has resolved the problem of state access. However, caching AJAX states is not so easy because, for such purposes, it is important to save the DOM and maintain JavaScript runtime environments. Further, the JavaScript engine i.e., Cobra has a poor and very limited support for complicated JavaScript functions [38].

In determining and understanding the state of user interface, both dynamically and asynchronously, Mesbah et al [36] present an auto-full-scan crawling algorithm by implementing a tool called Crawljax that automatically analyzes and reconstructs user interface states and can identify all clickable events that change the state of the application. From the discovered information, Crawljax infers a state flow graph for capturing states of user interface and possible transitions between them. Finally, a multipage state version of the original AJAX applications is produced in a methodical way. Crawljax performs crawling in two phases: inferring the state of machine and then generating pages that can be indexed. An embedded browser is used for executing AJAX technologies such as CSS, JavaScript, DOM and XMLHttpRequest. A robot simulates user inputs in the embedded browser. The controller is the key module which accesses the embedded browser in finding out changes in DOM and controlling robot actions. The module also shows responsibility for updating finite state machine when DOM encounters relevant changes and also calls the sitemap generator and mirror site generator process when the crawling is over. The finite state machine module maintains the state flow graph as well as pointers to the current state.

Crawljax uses Levenshtein [41] method for DOM tree comparison to differentiate one state from the other. Mesbah et al [reference] also developed specific language for AJAX crawling called Crawling AJAX Specification Language (CASL) for defining all clickable elements and their order in which AJAX-based web application should be crawled. They also suggest that in finding shortest path between initial and desired state, Dijkstra shortest path algorithm can be used for optimization. The proposed crawler may be used in carrying out state-based testing and using depth-first algorithm in automatically executing all user interface elements. However, reload and click-through functions are inefficient [38].

Benjamin et al. [34] demonstrate that the algorithms presented in [29, 36] are not efficient, and to date there is no tool which can truly crawl rich Internet applications (RIA). They proposed another strategy for efficient crawling of these applications called model-based crawling. The tool automatically infers an accurate model of RIA by conceptualizing the application as finite state machine. The states that this machine represents are distinct DOMs and transitions, and on the current DOM, they are JavaScript events such as MouseOver, OnSubmit, Onclick etc. [31]. They also present that a useful and accurate model must fulfil three conditions: (1) the model is complete i.e., it must represent all the transitions as well as all reachable states, (2) the model must be built using a deterministic way, and (3) the model should be produced efficiently. The first two conditions are easy to achieve but efficiency is obviously the most challenging one. To define an efficient strategy, a methodology named “Model-Based Crawling” was proposed in [8].

In model-based crawling, first the behavior of the AJAX-based web applications is explained with the assumption that the application will show such type of behavior. Based on the assumption, one can then define the anticipated model of the AJAX-based web application called the “meta-model”. This transforms the crawling process from the discovery activity to the activity of validating, the hypotheses do not need to be truly valid. Second, once the model is created based on assumed hypothesis, an efficient strategy for crawling may be defined to verify the model. Without assumption, it is impossible to create any strategy that will be efficient. Finally, in order to reconcile the predicted model with the reality of the application that is crawled, we immensely need to establish an approach. Whenever there is a notable difference encounters between the application model and the hypotheses, we need to take a corrective action for adapting the crawling strategy. The hypercube meta-model was developed in this context, based on the assumptions that the sequence of events execution does not change the reached state and that once the event execution completes, it becomes disabled and does not enable or disable other events [8].

In order to fulfill the requirement of efficiency, an efficient strategy is developed to crawl rich Internet applications that follow the hypercube meta-model. The strategy will visit each state of the hypercube as soon as possible. After visiting all states, the strategy completes the crawling by planning resourcefully to traverse with all such transitions. As in finite state machine, there are n! different paths from the bottom state to the top state. They also developed an algorithm that completes crawling in (n chose (n/2)*[n/2]) paths rather than n!. But for practical use, the proposed method is very complex and complicated and their current hypercube model can establish strategies for creating hypercube of maximum 16 dimensions [38]. Peng et.al [38] proposed graph-based AJAXCrawl to crawl AJAX-based web applications with minimal edge visits [38]. The approach has distinguished different AJAX events into reload events, prohibited events, state independent events and state dependent events in order to define an approach for triggering. The proposed graph-based AJAXCrawl adopts greedy approach [42] by looking at an event from the current state if there is an unexplored event. Otherwise, the crawler moves to the closest state with new events. The authors have also [38] proposed two other procedures, namely DFS and BFS and concluded experimentally that all three procedures have same performance by testing these on AJAX-based web applications. However, to answer the importance of information retrieval in dynamic client-side hidden AJAX content, Google also developed strategy for crawling AJAX-based web applications.

Google proposed two methods [32] (Hijax approach, AJAX crawling scheme). Hijax modifies the way of writing URL to explore dynamic content of a web page to search engines. The approach has several limitations as it does not cover the domain where the AJAX content is created dynamically based on the input of the user. According to Google AJAX crawling scheme there should be an agreement between website owner and crawlers to acknowledge these steps [32]. In brief, according to Google, first the crawler finds the pretty URL (a URL having #! Hash fragment) and transforms it into ugly URL, it then requests the server for the desired content. The server returns the result in the form of HTML snapshot, which is then processed by the crawler. However, there is no tool to automatically generate HTML snapshots, therefore the developers have to do a lot of work for its incorporation. Currently only Google makes use of it [13].

Zhang [7] proposed another approach to address the limitations of Google AJAX crawling scheme. The approach is based on HTML and JavaScript analysis through DOM using breadth-first search algorithm. The

DOM-based AJAX crawling approach generates static mirror of the website using State Transition Graph (STG) which is a directed graph DG <V, VR>, where V represents vertices i.e., states {v1, v2…} and VR represents transitions VR= {<v,w>|v,w ∈ V} between vertices v & w. The approach is simple, however there is no way to identify clickable and duplicate JavaScript functions and caring infinite loop at each level resultantly not focusing on efficiency of crawling and takes more time. It is can be concluded from the discussion that while several approaches have been proposed in literature in order to handle the issues regarding crawling AJAX-based web applications, we are still miles away from an ideal web crawler that handles the multiple states of a web page associated with a single URL. Further research should be conducted to explore ways to handle the issues regarding crawling of AJAX-based web applications discussed in this research article.

5.0 CONCLUSION AND RECOMMENDATIONS

Crawling AJAX is the process of making search engines capable of crawling today’s highly-interactive AJAX-based web applications. To date, web crawlers do not read AJAX-based web applications efficiently and systematically because AJAX-based web applications are highly dynamic even very granular change adds content to DOM through JavaScript functions. Crawling AJAX-based web applications is a very challenging problem, and has not been addressed by current search engines resourcefully. In order to enable web search engines to incorporate an AJAX crawler, it is necessary to keep in mind its high optimization. Most of web applications are AJAX-based so it is important to study more specific AJAX-based web applications and their technicalities for expanding AJAX crawler and taking care of search engine optimization.

There are several avenues for future work. The performance of AJAX crawler can be improved by an in-depth insight into request time, network latency, selection of clickables (on-click, double click, mouse-over etc.). Future work also includes improving the performance of technique by reducing the frequency with which duplicate transitions occur in the state machine. This can be done by avoiding invoking events on nested elements. For instance, consider the HTML tag: <td><b>shah</b></td>. Here clicking a nested tag produces the same effect as clicking on the <td><b>shah</b></td>. It means there is no need to invoke the same event again so to reduce the number of duplicate transition in STG.

REFERENCES

[1] R. R. Korfhage, *Information Storage and Retrieval*: John Wiley & Sons, 1997.

[2] G. C. P. Suganthan, "AJAX Crawler," in *Data Science & Engineering (ICDSE), 2012 International Conference on*, 2012, pp. 27-30.

[3] S. Batra, "AJAX - Asynchronous Java Script and XML," *ITS - Information Technology and Systems Management,* 2006.

[4] J. S. Zepeda and S. V. Chapa, "From Desktop Applications Towards Ajax Web Applications," in *Electrical and Electronics Engineering, 2007. ICEEE 2007. 4th International Conference on*, 2007, pp. 193-196.

[5] S. M. Mirtaheri, M. E. Dinçktürk, S. Hooshmand, G. V. Bochmann, G.-V. Jourdan, and I. V. Onut, "A brief history of web crawlers," *arXiv preprint arXiv:1405.0749,* 2014.

[6] L. Zhijie, W. Jiyi, Z. Qifei, and Z. Hong, "Research on Web Applications Using Ajax New Technologies," in *MultiMedia and Information Technology, 2008. MMIT '08. International Conference on*, 2008, pp. 139-142.

[7] X. Zhang and H. Wang, "AJAX Crawling Scheme Based on Document Object Model," in *Computational and Information Sciences (ICCIS), 2012 Fourth International Conference on*, 2012, pp. 1198-1201.

[8] M. A. Shayegan, S. Aghabozorgi, R. G. Raj, “A Novel Two-Stage Spectrum-Based Approach for Dimensionality Reduction: A Case Study on the Recognition of Handwritten Numerals,” Journal of Applied Mathematics, vol. 2014, Article ID 654787, 14 pages, 2014. doi:10.1155/2014/654787.

[9] N. Matthijssen and A. Zaidman, "FireDetective: understanding ajax client/server interactions," in *Software Engineering (ICSE), 2011 33rd International Conference on*, 2011, pp. 998-1000.

[10] F. S. Ocariza, G. Li, K. Pattabiraman, and A. Mesbah, "Automatic fault localization for client-side JavaScript," *Software Testing, Verification and Reliability,* vol. 26, pp. 69-88, 2016.

[11] S. Z. Z. X. Z. Li, "The Web asynchronous communication mechanism research based on Ajax," *Education Technology and Computer (ICETC),* vol. 3, pp. V3-370- V3-372, 2010.

[12] Q. Tang and C. Zhao, "The application of Ajax asynchronous refresh in general used database maintenance system," in *Geoscience and Remote Sensing (IITA-GRS), 2010 Second IITA International Conference on*, 2010, pp. 108-110.

[13] bbriniotis. (2012). *Google's AJAX crawling scheme and its effects on SEO*. Available: http://www.webseoanalytics.com/blog/googles-ajax-crawling-scheme-and-its-effects-on-seo/

[14] j. j. garrett, "Ajax: A New Approach to Web Applications," February 18, 2005.

[15] S. Choudhary, Mustafa, M. Seyed, Mirtaheri, M. Ali, v. B. Gregor, J. Guy-Vincent, and O. Iosif-Viorel, "Crawling Rich Internet Applications: The State of the Art," *Research and Development, IBM Security AppScan* 2012.

[16] L. G. W. Huan-min, "Research and application of the Ajax page's dynamic access technology based on .NET," *Robotics and Applications (ISRA), 2012 IEEE Symposium on Digital Object Identifier,* vol. 101- 104, 2012.

[17] P. J. Gupta, K, "Implementation of Web Crawler," *Emerging Trends in Engineering and Technology (ICETET), 2nd International Conference on Digital Object Identifier,* pp. 838 - 843 2009.

[18] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer Networks and ISDN Systems,* vol. 30, pp. 107-117, 1998.

[19] Monica Peshave, Kamyar Dezhgosha, "HOW SEARCH ENGINES WORKAND A WEB CRAWLER APPLICATION," *University of Illinois at Springfield One University Plaza, Springfield USA,* 2005.

[20] M. Najork, "Web Crawler Architecture," in *Encyclopedia of Database Systems*, L. Liu and M. T. ÖZsu, Eds., ed: Springer US, 2009, pp. 3462-3465.

[21] L. Baker. (2005). *Anatomy of a Search Engine Crawler*
Available: http://www.searchenginejournal.com/anatomy-of-a-search-engine-crawler/2230/

[22] V. Shkapenyuk and T. Suel, "Design and implementation of a high-performance distributed Web crawler," in *Data Engineering, 2002. Proceedings. 18th International Conference on*, 2002, pp. 357-368.

[23] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*: McGraw-Hill International, 1983.

[24] M. Shokouhi, P. Chubak, F. Oroumchian, and H. Bashiri, "DESIGNING AND IMPLEMENTATION OF" REGIONAL CRAWLER" AS A NEW STRATEGY FOR CRAWLING THE WEB," *e-Society 2004,* p. 711, 2004.

[25] B. C. Boldi, M. Santini, and S. Vigna, "UbiCrawler: a scalable fully distributed web crawler," *Software: Practice and Experience,* vol. 711–726, 2004.

[26] P. Chubak and D. Rafiei, "Efficient indexing and querying over syntactically annotated trees," *Proceedings of the VLDB Endowment,* vol. 5, pp. 1316-1327, 2012.

[27] A. Heydon and M. Najork, "Mercator: A scalable, extensible Web crawler," *World Wide Web,* vol. 2, pp. 219-229, 1999.

[28] S. Raghavan and H. Garcia-molina, "Crawling the Hidden Web," *In Proc. of VLDB,* pp. pages 129–138, 2001.

[29] C. Duda, G. Frey, D. Kossmann, and C. Zhou, "AJAXSearch: crawling, indexing and searching web 2.0 applications," *Proc. VLDB Endow.,* vol. 1, pp. 1440-1443, 2008.

[30] C. Duda, G. Frey, D. Kossmann, R. Matter, and Z. Chong, "AJAX Crawl: Making AJAX Applications Searchable," in *Data Engineering, 2009. ICDE '09. IEEE 25th International Conference on*, 2009, pp. 78-89.

[31] K. Benjamin, "A Strategy for Efficient Crawling of Rich Internet Applications," PhD, University of Ottawa, 2010.

[32] Google. *AJAX Crawling*. Available: http://code.google.com/intl/en/web/AJAXcrawling/index.html

[33] R. Matter, "AJAX Crawl:Making AJAX Applications Searchable," *Master Thesis,* July 31, 2008.

[34] K. Benjamin, G. V. Bochmann, M. E. Dincturk, G.-V. Jourdan, and I. V. Onut, "A strategy for efficient crawling of rich internet applications," presented at the Proceedings of the 11th international conference on Web engineering, Paphos, Cyprus, 2011.

[35] R.G. Raj, A.N. Zainab. “Relative Measure Index: A Metric to Measure the Quality of Journals”, Scientometrics, vol. 93, no. 2, 2012, pp. 305-317. doi: 10.1007/s11192-012-0675-z.

[36] A. Mesbah, E. Bozdag, and A. van Deursen, "Crawling AJAX by Inferring User Interface State Changes," in *Web Engineering, 2008. ICWE '08. Eighth International Conference on*, 2008, pp. 122-134.

[37] S. Shah. (2007, Crawling ajax-driven web 2.0 applications. Available: http://www.net-security.org/article.php?id=973

[38] M. Mansourvar, M.A. Ismail, R.G. Raj, S.A. Kareem, S. Aik, R. Gunalan, et al. "The applicability of Greulich and Pyle atlas to assess skeletal age for four ethnic groups", *Journal of forensic and legal medicine*. vol. 22: pp. 26–29, 2014. pmid:24485416.

[39] IRange. (2012). *AJAX Projects*. Available: http://www.ajaxprojects.com/ajax/news.php

[40] W3C. *Document Object Model*. Available: http://www.w3.org/DOM/

[41] Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," 1996.

[42] Y. Wang, J. Lu, and J. Chen, "Crawling Deep Web Using a New Set Covering Algorithm," presented at the Proceedings of the 5th International Conference on Advanced Data Mining and Applications, Beijing, China, 2009.