Complex & Intelligent Systems (2023) 9:3635–3653
https://doi.org/10.1007/s40747-021-00471-1

ORIGINAL ARTICLE

IHWC: intelligent hidden web crawler for harvesting data in urban domains

Sawroop Kaur[1] · Aman Singh[1] · G. Geetha[2] · Xiaochun Cheng[3]

Received: 23 October 2020 / Accepted: 10 July 2021 / Published online: 24 July 2021

Abstract

Due to the massive size of the hidden web, searching, retrieving and mining rich and high-quality data can be a daunting task. Moreover, with the presence of forms, data cannot be accessed easily. Forms are dynamic, heterogeneous and spread over trillions of web pages. Significant efforts have addressed the problem of tapping into the hidden web to integrate and mine rich data. Effective techniques, as well as application in special cases, are required to be explored to achieve an effective harvest rate. One such special area is atmospheric science, where hidden web crawling is least implemented, and crawler is required to crawl through the huge web to narrow down the search to specific data. In this study, an intelligent hidden web crawler for harvesting data in urban domains (IHWC) is implemented to address the relative problems such as classification of domains, prevention of exhaustive searching, and prioritizing the URLs. The crawler also performs well in curating pollution-related data. The crawler targets the relevant web pages and discards the irrelevant by implementing rejection rules. To achieve more accurate results for a focused crawl, ICHW crawls the websites on priority for a given topic. The crawler has fulfilled the dual objective of developing an effective hidden web crawler that can focus on diverse domains and to check its integration in searching pollution data in smart cities. One of the objectives of smart cities is to reduce pollution. Resultant crawled data can be used for finding the reason for pollution. The crawler can help the user to search the level of pollution in a specific area. The harvest rate of the crawler is compared with pioneer existing work. With an increase in the size of a dataset, the presented crawler can add significant value to emission accuracy. Our results are demonstrating the accuracy and harvest rate of the proposed framework, and it efficiently collect hidden web interfaces from large-scale sites and achieve higher rates than other crawlers.

Keywords Hidden web · Intelligent crawling · Urban planning · Smart cities

Introduction

Smart cities are the essence of new age comfortable living in urban areas such as towns and cities. The Indian government has launched smart cities project intending to promote sustainable cities. There are certain objectives for the development of smart cities. One such objective is the reduction in air pollution and making better area-based developments. To implement solutions regarding this objective, the data are required to be crawled. The objective of this study is to implement intelligent location-aware hidden web crawling focused on urban pollution data. A supervision-based hidden web crawler is developed for collecting data and it is implemented for both hidden web domains and for crawling pollution data from the web. From the numerous ways to collect data, web search is one of the most used search methods. It is claimed that 85% of the users rely on search

✉ Aman Singh
amansingh.x@gmail.com

Sawroop Kaur
srbal87@gmail.com

G. Geetha
gitaskumar@yahoo.com

Xiaochun Cheng
xiaochun.cheng@gmail.com

[1] Department of Computer Science and Engineering, Lovely Professional University, Phagwara, Punjab, India

[2] Advanced Computing Research Society, Chennai, Tamil Nadu, India

[3] Department of Computer Science, Middlesex University, London NW4 4BT, UK

engines to find information. Two-thirds to three-quarters use the web as their primary source of information, while two-thirds to three-quarters were unable to get the information they want [1]. We are living in the modern age of the web. Search engines play a prominent role in our lives. Though information retrieval is not only confined to web search, a web crawler is also a more practical and reliable way. Web crawling is either surface web crawling or hidden web crawling. The former is publicly and directly accessible and has a statistical address while the latter is hidden behind the query interfaces is accessible via registration, search interface and paid access. For instance, a publisher has published some research articles. To get access to those articles, one has to search through the search engine of the publisher or get a paid access. When you are not able to search something through the index of easy to access search engines which mean data are intentionally hidden or masked or protected by a password. Those articles belong to the hidden web. An additional layer to get authorization to the hidden web requires information from the user. Discrete and premium content can only be accessed via authorisation and is not easily available. For example, advanced citations on PubMed can be accessed only after paying the required fee.

The web is getting more hidden due to the presence of a large number of online databases. The user has to pose a query to the database to get the answers. Hidden web crawler has to carefully discover and classify hidden web pages and forms. It requires an automatic web classification mechanism to classify webpages in relevant classes. The following factors add to the complications in hidden web crawling:

- The existence of web databases is wide and diverse. Due to the explosive growth of web-based data, distribution of web forms is sparse and heterogeneous. Finding domain-specific databases from the vast amount of web-based data is not an easy task. Web databases and forms are meant to be searched by human users. Automation of a similar interaction is a difficult task.
- Web forms rarely have the same structure and content it further restricts crawlers to get acquainted with web databases.
- Forms act as an entry to web databases. Just detecting a <form> tag is not sufficient to go beyond the walls of the hidden web. The forms even if non-searchable may appear similar in structure to searchable. It is expected from the web crawler to categorize the forms into searchable and non-searchable. There exist numerous webpages which have form tag but are not further searchable.

As a new attempt, an intelligent hidden web crawler for harvesting data in urban domains (IHWC) effectively detects and submits the forms and categorizes them into a searchable and non-searchable category. The approach is also tested for collecting pollution data from the web. At present, the approach is tested for web-based crawling of particulate matter (PM): PM10 and PM 2.5 in the air of Amritsar, Jalandhar and Ludhiana. These three cities are proposed smart cities in Punjab. The geographic location awareness of a crawler is based on these cities. The study is divided into two parts. First, a web crawler is developed then the approach is tested and validated for pollution data along with five other crawling domains. The major contributions of the research are as follows:

- Effective three-step classification strategy for domain classification.
- Rejection rules to decide which forms are non-searchable. This not only saves time and resources but also prevent exhaustive form crawling. First, the potentially interesting web pages are located. Then rules of rejection are followed to find the web pages that belong to the hidden web category.
- Introduction of rules for stopping criteria to save the crawler from falling into spider traps.
- This approach works for both get and post methods. It does not only detect and locate the searchable web pages but also submits them automatically.
- The crawler is scalable as it can adapt to the increasing size of the hidden web. It is also extensible to other third-party components like indexing.

The remaining part of the paper is structured as follows: the next section throws light on the existing pioneer works in hidden web crawling. The third section describes the steps designed for the proposed work. The fourth section discusses experimental results and subparts of the framework in detail. While the last section provides a conclusion and outline of future work.

Related work

Exhaustive crawling is a waste of resources, as the world is now moving towards the domain-specific search. A web crawler is designed to crawl the web data [2]. Focused web crawlers are one such type that contributes to this area. The cooperation of intelligent, focused and hidden web crawlers is required to design a special crawling strategy that determines the degree of relevance of a predefined web page with a web page that is crawled. From their time of development [3], focused crawlers have improved in a variety of ways. On being categorized into various types, in the hidden web, these are called form-focused crawlers [4]. Based on the given topic, focused crawlers attempt to find the most promising links. As far as domain-specific web databases are concerned, focused crawlers find relative pages as well

as the correlation between domains and domain-specific web databases [5].

The first step in hidden web crawling is the detection of web forms which act as an interface to search an online database. This step gives the crawler preliminary access to web forms. The problem with the hidden web is finding form tag is easy for any crawler but not all forms are meant to search. Some forms such as an email subscription and mailing list should be discarded by a crawler. This generates a need for a crawler to automatically discern these pages from searchable forms and then discard them. On this basis, techniques are divided into first heuristic-based techniques such as the presence of search, find and query [6], form with at least one text box [7], or discard forms with short input. The second type is to get form access based on machine learning. In these techniques, a classifier is trained to correctly classify forms, topic, link and page. A technique proposed in [8], the crawler extracts and analyzes the features of web forms, based on name and action of attributes in addition to name of fields and their values. Using these feature, C4.5 algorithm identifies entry to hidden web, perform and learn classification. Another technique called form-focused crawler (FFC) that consist of four classifiers to classify, links, forms topic and page. This technique has limitation of manual training of link classification [9]. Hicks et al. [10] worked on a crawler that used an ordered list of terms related to domain search. A crawler is provided with a manually created source description file with example queries.

To automatically locate the potential hidden websites, Li et al. [5] worked on four classifiers: if the web page consists of desired and relevant information is classified by term-based classifier, which URL will lead to relevant web pages is classified by a link-based classifier. To decide which form is searchable, a search form classifier is implemented. The fourth one called domain-specific classifier which selects only those pages which have related information to the domain. All the existing classification techniques can be compared using heuristic and machine learning, whether the machine-learning technique is supervised or not, whether the web pages are classified before submission or after submission, or which features are used to classify forms, or whether the technique is focused towards the hidden websites or not. To submit forms most commonly used technique is the pre-query technique. Table 1 shows the comparison of techniques to find entry to the hidden web.

Once the entry is identified, next step is to automatically fill and submit the forms. It was first introduced in [11]. The HiWe crawler can fill the forms semi-automatically. Their work has proved as an important step in the automation of submission of the form. Shopbot is another crawler that helps the user to compare the prices of the selected products and help the consumer while shopping [12]. To fill out forms, domain heuristics are used but only for shopping-related web forms. Liddle et al. [13] designed a model to submit forms to their default values. While He et al. [14] considered only one field for form modeling and rest all are submitted using default values. Doan et al. [14] used a flat compilation where labels can be extracted automatically. These techniques can be classified into supervised or non-supervised. Non-supervised extraction performs an HTML code analysis to identify the labels from it. Two methods either DOM tree [11, 15], based, or using visual techniques [16–18], have been reported.

The next step is to fill the form with valid values. The combinations of values are used to fill and submit the form. The submission request is received by the web database. For a web site server, these values are termed as the query. If a crawler submits every possible combination of the form values to retrieve all the information, it will considerably increase the burden on the resources consumed by the crawler. Furthermore, some queries will retrieve the same results, some web pages on submission do not respond. This will affect the scalability of the crawler. To fetch more results, crawler can also get blocked because in this way, it will contravene the politeness policy. The crawler must crawl within the limits of politeness policy. If a web form is a simple form, the keywords values can be chosen from the webpage itself. The webform and the webpage are always related. In this case, a method is required to which can automatically extract the keywords but without human intervention.

Table 1 Comparison of reported techniques based on automated discovery of hidden web interfaces

References	Type of proposal (heuristic/machine learning)	Pre-query (pre)/ post-query (post) method	Focused/ non-focused	Algorithm for classification	Supervised or not
[6]	Heuristic based	Pre	No	Not mentioned	No
[27]	Classifier based	Pre/post	No	C4.5	No
[28]	Classifier based	Pre	Yes	C4.5	Yes
[29]	Heuristic based	Pre	No	Not mentioned	No
[30]	Classifier based	Pre	Yes	Naïve Bayes	Yes
[31]	Heuristic based	Post	Yes	Not mentioned	Not mentioned

Link selection algorithm highly contributes to the performance of focused hidden web crawler. A crawler needs to decide which URLs will help to locate the searchable form, finding the relation between web documents and which links to keep or discard otherwise. The existing ranking technique such as PageRank and HITS are not suitable for hidden web crawling [19]. Experimental results from [20] proved that without using link selection, only 94 relevant searchable forms were found from 100,000 crawled pages. This problem was overcome by [9] and [21] by working on form-focused crawler (FFC) and adaptive form-focused crawlers for hidden web entries (ACHE). The latter has a high harvest rate than the former. One another factor that affects crawling performance is breadth or depth-first crawling. Experimental results from [22–24] have favored breadth-first. The Naïve Bayes-based focused crawlers developed in [25] proved that naïve Bayes and breadth-first search have performed better as compared to page rank and breadth-first crawler. A URL-based classifier in [26] has been used for topic-specific search. URLs are analyzed to find the patterns and formal representation. To discover similar pages, regular expressions are utilized.

The studies discussed above cover either of the steps in hidden web crawling. Web crawling is one of the prominent method being applied in data collection for applications such as crawling user-generated blogs for recognition of modern traditional medicine [32], information extraction from social web forums [33], industrial digital ecosystem [34] and for carbon emission [35]. Crawling for all the domains is difficult, so a crawler is required to be focused on certain domains as well as intelligent rules are required to stop unproductive crawling and spider traps. To our knowledge, it has been the first kind of work where hidden web crawling has been applied for fetching and analyzing pollution-related data in three cities of Punjab state. The dataset required for this goal is entirely based on web data. To precise, our goal for pollution-related data, yet only two features PM 10 and PM 2.5 are included. The following sections explain the framework of the proposed crawler and experimental results.

Framework of ICHW

Web databases do not present their internal view. Web forms are meant to be filled by user no matter if it is a simple form with one or two fields or a complex form with multiple attributes. Web forms give the user access to a hidden web database. Each attribute is labeled with relevant information that guides the user to fill the form. The performance measure for the crawler is harvest rate. Suppose the total number of web pages be denoted by W_n, W_c denotes the number of web pages crawler has crawled. From W_n, suppose N_j pages are searchable forms. N_c is the searchable forms crawled by the crawler. H_r is a harvest rate:

$$\sum H_r = \frac{N_c}{W_c}, \quad \text{where} \quad 0 < W_c, Nc. \tag{1}$$

It is required from the crawler that it should have a high harvest rate, use criteria to prioritize the URLs to focus towards more coverage and relevant webpages and also save itself from the crawlers' traps. It is mentioned in [36] that most of the search interfaces are found on the home page. Therefore, if the crawler found the form tag, it is checked if it is searchable or not. If found searchable then it is classified into a relevant class. The framework of ICHW is shown in Fig. 1. It mainly has the following components:

- *The frontier for seed URLs*: Frontier for seed URLs: seed URLs play an important role in focused web crawling. There are two types of techniques for the selection of seed URLs—Bootstrap based and machine learning

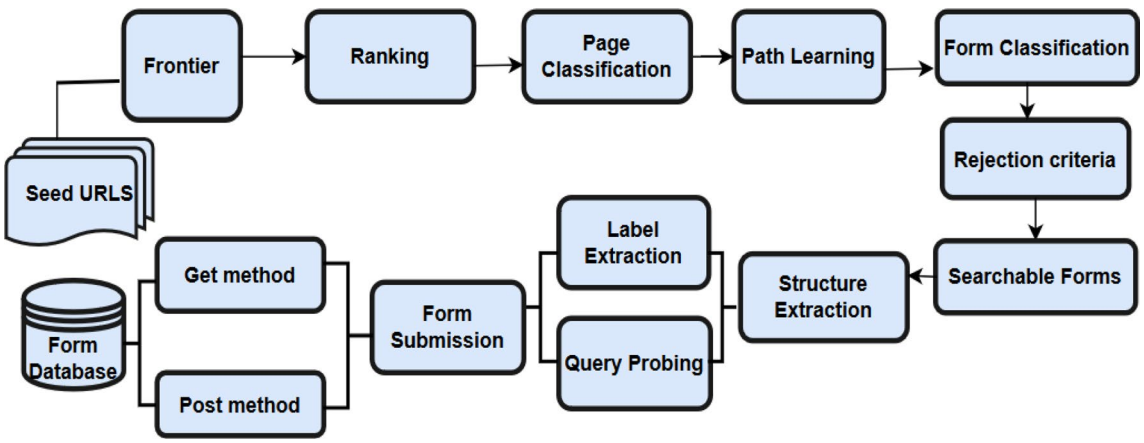

Fig. 1 Framework of IHWC

based. The main function of the frontier is to keep the ordered list of URLs. Suppose W is the web page, and W_i is the URLs available on W, for each value of i in W_i, there exist some URLs denoted as W_{ij}. The links from W are kept in frontier for seed URLs. Further links are kept in fetched link frontier [37]. The proposed crawler is focused on property, book, flight, hotel, music, premier and product domains.

- *Ranking module*: Ranking plays a core role in hidden web crawling [38]. The goal is to crawl the top k sources. This module is focused on ranked crawling of hidden web sources. The conventional crawling techniques with unranked data sources and the crawling algorithms are not suitable for the hidden web. Unranked data sources face query bias problem [39]. A hidden web database is said to be ranked if it returns k top sources. The ranking in this approach is not dependent on queries. The ranking formula is triplet factor of out of site links, term weighting and site similarity. The following formula is proposed for ranking:

$$€(\text{ranking reward}) = w_{ij} + S + \text{SF}, \tag{2}$$

where w_{ij} denotes the weight of term, S denotes similarity or value of cosine function, and SF denotes the weight of term.

$$(rj) = (1 - w).\delta j + w.(€)/cj, \tag{3}$$

where w is the weight-balancing factor for ranking reward and c_j is the network and bandwidth consumption factor. δj is the total number of the new document retrieved.

- *Page classification*: It analyses the web page to find that if a searchable form belongs to a certain domain. It selects the relevant searchable forms and adds them to the form database wherein they are not already present.
- *Path learning*: It leads the crawler to the good links, i.e., the searchable forms.
- *Form classification and rejection rules*: Form classification judges whether a form is searchable or non-searchable and filters out those non-searchable forms.
- *Structure extraction*: It extracts the structure of the form for form filling. Forms are parsed to create a repository for values. The repository consists of a control element, label, domain, type of domain size and status after the structure of forms is extracted.
- *Form filling and submission*: Forms are filled with suitable values from the repository. Form's submission includes GET and POST method.

Employing the above components, the crawler coordinates its effective search towards finding the entry into the hidden web. It avoids misuse of resources and unproductive crawling by implementing effective stopping criteria and rejection rules. Component (1)–(3) help finding the sources of hidden web content, i.e., the first step in hidden web crawling. While components (4) and (5) makes the second step is extracting the underlying content. The components called form filling and submission is the third step in hidden web crawling, i.e., extracting the underlying content. Figure 1 shows detailed components and the next section explains the underlying algorithms.

Experimental results

Web page classification

The page classification is based on the similarity index between the web page extracted by the crawler and the seed pages of a specific domain. Before the actual crawling begin, the URLs are pre-processed to develop a feature vector. The success of classification relies heavily on the feature vector. The following subsection explains the pre-processing of URLs.

Pre-processing

URL and in-links are first divided into their baseline components. All the words present will be fed as features of URLs. A similar process is followed for anchor pre-processing and text around the anchor. A website is called a hidden website if it contains searchable forms along with the database at the back end. Feature space for a hidden web site is based on URL, anchor and text around the anchor. Each hidden website has further associated links. For further links, feature space is constructed with the path. Let feature space for the hidden website be denoted as FS and feature space for links be denoted as FL:

$$\text{FS} = [\text{Url,anchor, text around an anchor}], \tag{4}$$

$$\text{FL} = \big[\text{Url, anchor, path}\big]. \tag{5}$$

First, stop words are removed. The next step is stemming, using the Porter stemming algorithm. The top m terms are selected. After pre-processing, the URL is represented as

$$U = \big[u, a, t, p\big],$$

where u is the URL, a is the anchor, t is the text around URL, and p is the path of URL. Now different weights are assigned to vector U:

$$Tf_{ij} = u \times tf_{ij1} + a \times tf_{ij2} + t \times tf_{ij3}, \tag{6}$$

$$Tf_{ij(\text{link})} = u \times tf_{ij1} + a \times tf_{ij2} + t \times tf_{ij3}, \tag{7}$$

$$w_{ij} = \frac{tf_{ij} \times idf_j \times Ig_j}{\sqrt{\sum_{N=1}^{N} (tf_{in} \times idf_m)2}}, \tag{8}$$

where W_{ij} is the weight to term t_j in document d_i, t_f denotes term frequency, id_f denotes inverse document frequency, N denotes a total number of documents and IG stands for information gain.

$$\text{IG}_j = h(d) - h\left(D|t_j\right), \tag{9}$$

$$h(d) = -\sum\nolimits_{di\varepsilon D} p(di) \times \log_2 p(di), \tag{10}$$

$$H\left(d|t_j\right) = -\sum\nolimits_{di\varepsilon D} p\left(d_i|t_j\right) \times \log_2 p(d_i|t_i). \tag{11}$$

Similarity computation

The goal of similarity computation is how to efficiently construct a similarity index [40]. The crawler needs to explore similar URLs that link to a particular class.URL, anchor and text around anchor are used for the construction of feature vector. The web pages in web directories are organized in hierarchal order. For example, the URL abc.com/holiday/new-year/music will be similar to other URLs in its class. In addition, the URL will be similar to abc.com/holiday in other class. Let the distance between the webpage be denoted by D. S be the source document. L be another document in a class hierarchy. The formula for similarity computation is used similarly as defined in [37]. It is computed between new-found URL and already discovered URLs. The function of determining the similarity is cosine similarity, defined by Sim $(U_1, U2)$:

$$\text{Sim}\left(U_1, U_2\right) = \frac{U_1.U_2}{|U_1| \times |U_2|}. \tag{12}$$

Algorithm: a novel three-step algorithm for ICHW

Step 1: A web page is encountered after pre-processing feature vector is generated. Equation (12) is used to measure the degree of similarity between the new webpage and the focused domain. If the value of the similarity function is above the threshold, then the page is considered relevant. All the links present on the page are extracted and checked for form tag. Otherwise, the next step is followed.

Step 2: Calculate the similarity of other domain (property, book, flight, hotel, music, premier and product) and page to find the domain with the highest similarity value. If it is relevant, extract the links.

Step 3: Parse the web pages that are found relevant. Form element table in a repository for a domain-specific database is developed as shown in Fig. 4. It also shows the outline of the steps involved in creating the repository and when the forms are submitted with correct values how the response has been generated.

Path learning

The goal of path learning is to extract only those links which with minimum hops can lead the crawler to the hidden web databases. Some of the links are considered good, while others are discarded. Along with jasmine directory and amazon, 20 real websites from Alexa's list of top sites are exhaustively crawled to check at which depth most web pages are found. Our observation is similar to [37]. Below the depth of 6, the crawler was not able to find a considerable percentage of forms. The simplest reason for this is that form is designed for human interaction. In addition, for this, most of the times forms are put on upper levels. Due to this reason, the depth of the crawler is limited to the 3. It is also observed that from the crawled URLs the number of URLs for book domain are high as compared to others. Figure 2 justifies the observation. Backlinks also impact the performance of the focused web crawler. Following the connection between the web pages, crawler the good target pages. Feature's vector is constructed for FS and FL as explained in Eqs. 2 and 3. FL is calculated at each level. From a webpage, a huge number of feature vector can be extracted. But due to length and space constraints, the top 10 features are used and are constructed as explained in "Pre-processing". The good links are either immediate benefit link or delayed benefit links. Immediate benefit links are at level 1, while delayed benefit links are at levels 2 and 3. The next step is to compute the similarity between the FS and FL. The similarity is computed between the already discovered source and the new-found source as explained in "Similarity computation".

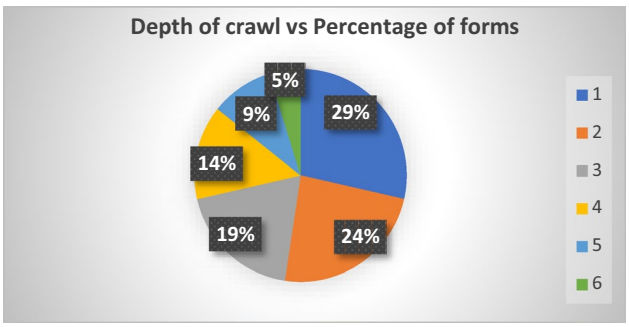

Fig. 2 Depth of crawl vs percentage of forms found at particular depth

Searchable form classification

The ultimate goal of the crawler is to grab maximum searchable forms. First, the crawler has to make a distinction between searchable and non-searchable forms. This study introduced a rejection framework for non-searchable forms. When URL is encountered, it is checked for <form> tag. If it has <form> tag, it is considered a hidden website. Then, it is parsed for attributes type, a number of attributes, submit button, button marker, login, mailing subscription, and registration to find non-searchable forms.

Figure 3 shows the proposed rejection framework. After this step, the system has URLs that belong to the hidden web category and the searchable forms. We have manually extracted 150 URLs that are real hidden web sites from the jasmine directory and Alexa's list of top URLs. 100 negative samples are extracted manually over the mentioned domains. The experiments have been constructed using k-fold cross-validation for the Support vector machine (SVM) and K nearest neighbor (KNN). Results also show the impact of the ratio of the split of data. From the parsed form representation, a repository is created which act as the source for form filling.

Repository construction

After parsing, the form values are extracted for repository creation and are filled with the help of possible values of associated controls afterwards. The forms are submitted to the webserver. Now either the form will respond with suitable data otherwise based on response status, a web page with a certain response code will be returned. If the status code is 200, it indicates asynchronous response, if the status code is 400—bad request error, 413—payload too large- request entity is large, 414—payload too large- URI too large. If status code is 500 or 513—internal server error, and service unavailable, respectively. Initially, the repository is manually populated with instances from seed sites. Labels and associated values are extracted to create the repository. The crawler is based only on a finite domain, as it encounters a form with a finite domain, it extracts the label and domain values. This will help the crawler to adaptively learn and fill forms with suitable values. Let us suppose a user wants to book a flight (Table 2). Table 3 shows the value of the form element table after parsing the form. Figure 4 shows the steps involved in the form submission.

Forms either have already available value or there are text fields that a user fill. Automatic text field submission is difficult. For the sake of simplicity of the approach, we have skipped the submission of forms using text fields. The

Table 2 Parsed values for form element table of flight booking

Control element	Domain attributes
Select	Depart from
Select	Going to
Select	Traveller
Select	Class
Radio	Economy
Radio	Premium economy
Radio	Business
Checkbox	Non-stop flight

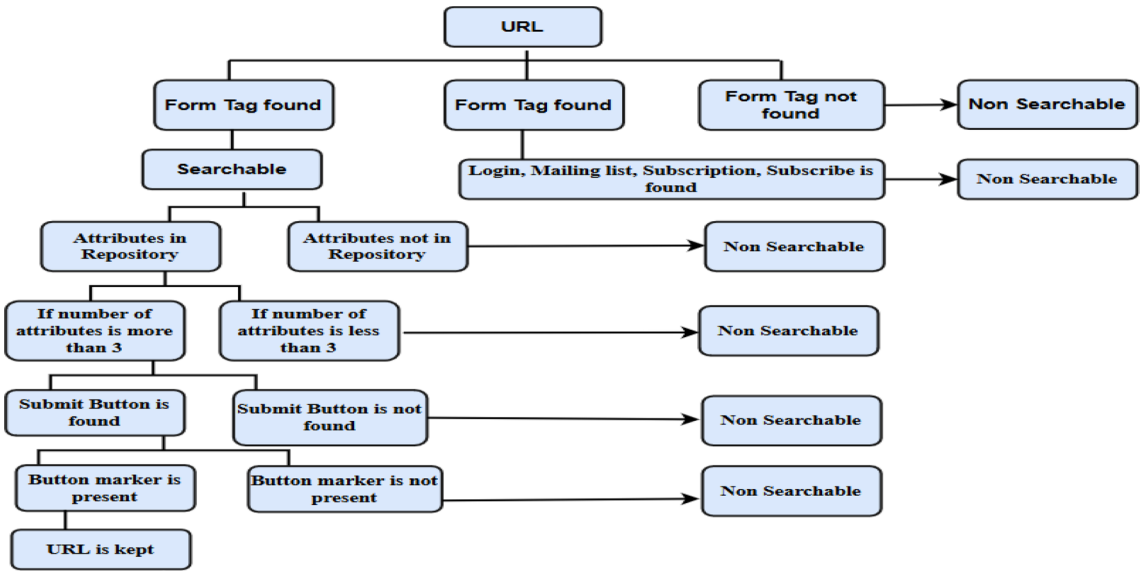

Fig. 3 Rejection framework for URL

Table 3 Domain of experiment and description of the search term

Domain	Description
Property	Property search
Book	Book search
Flight	Airfare search
Hotel	Hotel search
Music	Music CDs search
Product	Household product search

searchable form is parsed for creating the form element table (FET) as shown in Table 3.

It acts as a source for forms values. Forms are submitted either by GET method or POST method. After submission, a repository of crawled hidden web pages is created. From those URLs, further analysis is made. In our case, it is observed that all the crawled URLs have either HTTP or HTTPS prefix. From all the status codes, the number of web pages with status code 524 is only 2. Most of the forms are from depth 2 and 3.

Stopping criteria and threshold

To stop crawler from unproductive exhaustive crawling, stopping rules such as maximum crawl depth = 3 and the threshold is designed. While the assumptions are the same as in [37]. The problem with the database-driven web is that the crawlers keep crawling the data under the infinite loop and actual valuable web page are usually skipped by the crawler. Stopping criteria are designed to save the crawler from the trap of infinite searching loops. The crawler uses rejection rules, stopping criteria and a threshold of 80 new URLs and 100 new forms. As the crawler has reached 80 new URL, it will start in-site searching. After 100 new forms at each depth, it jumps to the next depth. Figure 2 shows the percentage of forms found at each depth. The forms after depth 6 were not considered as they are less in number. Table 10 shows the running time of the crawler with and without using rejection rules.

URL classification based on soft marginal formulation

Almost all the real-world web data have linear inseparability. Support vector machine (SVM) is used to classify the blocks, and k-fold cross-validation is used for evaluation. Under the soft margin formulation, the linear kernel-based SVM classifier makes a certain number of mistakes and keeps the class margin (CM) as wide as possible to correctly classify the points. It is expected that the system must choose a decision boundary that perfectly separates the features to avoid overfitting. Under soft marginal formulation, SVM is allowed to make mistakes to keep the margin wide. In this way, other points can be still be classified correctly:

$$L = \frac{1}{2}\left\|w^2\right\| + \nu(\text{number of mistakes}). \tag{13}$$

Hyperparameter ν chooses the trade-off between maximizing the margin and minimizing the mistakes.

- If ν has a small value, classification mistakes are given less importance. More focus is given to maximize the margin.
- If ν has a large value, the focus is more on avoiding misclassification.

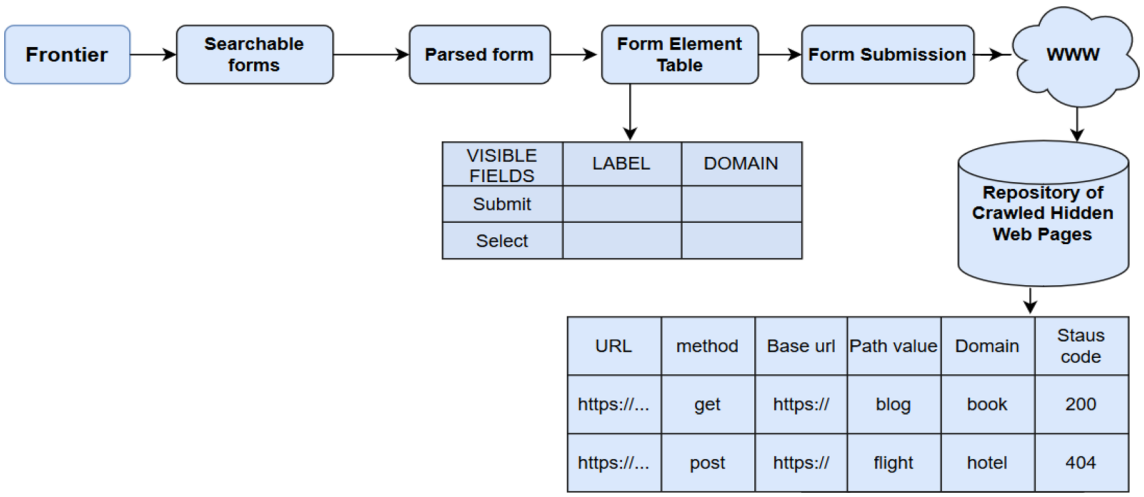

Fig. 4 Steps involved in the submission of forms

More penalty is incurred by the points which are far away on the wrong side of the decision boundary. For every data point x_i, there exists a slack variable ξ_i

- ξ_i = distance of x_i from the *CM*, if x_i is on the incorrect side of the margin,
- $\xi_i = 0$, if x_i is on the right side.

Each x_i has to satisfy the constraint of

$$y_i\left(\vec{w} \cdot \overrightarrow{x_i} + b\right) \geq 1 - \xi_i. \tag{14}$$

The LHS of the equation is the confidence score denoted by CS.

- For CS ≥ 1, the classifier has classified the point correctly.
- For CS ≤ 1, the classifier did not classify the point correctly, and a penalty of ξ_i is incurred.

Each point *P* is represented by *P(x,y)*, ϕ is transformation function for point *P* as follows:

$$\phi^{(P)} = \left(x^2, y^2, \sqrt{2xy}\right). \tag{15}$$

Minimization function is defined as

$$L = \frac{1}{2}\vec{w}^2 + C\Sigma_i \lambda i y_i\left(\vec{w} \cdot \overrightarrow{x_i} + b\right) \geq 1 - \xi_i, \tag{16}$$

$$L = \Sigma_i \lambda_i - \frac{1}{2}\Sigma_i \Sigma_j \lambda_i \lambda_j y_i y_j x_i \cdot x_j, \tag{17}$$

$$k(x, y) = \langle \phi(x), \phi(y) \rangle, \tag{18}$$

$$\left(P_1, P_2\right) = \left\langle \phi\left(x_1 y_1\right) +, \phi\left(x_2 y_2\right) \right\rangle, \tag{19}$$

$$k\left(P_1, P_2\right) = x_1^2 x_2^2 + y_1^2 y_2^2 + 2x_\perp y_1 x_2 y_2, \tag{20}$$

$$K\left(P_1, P_2\right) = \left(x_1 x_2 + y_1 y_2\right)^2, \tag{21}$$

$$k\left(P_1, P_2\right) = \left\langle P_1, P_2^2 \right\rangle. \tag{22}$$

In real-world web data, it is difficult to find exact similar data. Therefore, we have kept the notion of similarity as to how close the points are. The main takeaway from this is we have implemented linear classification in higher dimensional space.

Similarly, in the case of KNN, to work with maximum separability, for example, a dataset has *N* number of classes. μ_b is the mean vector, where b = I, 2, 3,…,*N*. Let x_b be the total number of samples.

$$x = \sum_{b=0}^{N} x_b, \tag{23}$$

$$M_P = \sum_{b=1}^{N} \sum_{c=1}^{X_c} \left(y_c - \mu_b\right)\left(y_c - \mu_b\right)^T, \tag{24}$$

$$M_Q = \sum_{b=1}^{N} \left(\mu_b - \mu\right)\left(\mu_b - \mu\right)^T, \tag{25}$$

$$\mu = \frac{1}{A} \sum_{b=1}^{N} \mu_b. \tag{26}$$

Distance of all instances is measured from each other using Euclidian distance metric. The instance with maximum distance is selected and is called training distance. If the boundary is 1.5 or 2 times of training distance, it indicates classes are closer to each other. The approach has implemented non-exhaustive cross-validation. Under which k-fold cross-validation is implemented. For our approach, the value of $K = 5$ comes out to be most suitable. With the aim of maximizing the prediction accuracy, non-perimetric neighborhood component analysis is used for selecting features. After the domains are classified as relevant, using the varied queries forms are submitted. If the form is correctly submitted, its status code is 200. Precision, recall and F1 score are computed using SVM and KNN algorithms.

Form identification and analyses

After the crawler has identified the form, these are analyzed to explore the form elements. Each form is equipped with text, HTML elements, and bounded or unbounded controls. The proposed approach is based on bounded controls only.

The above table shows the domains and the search terms used for the experiment. We have selected six domains for the dataset. This dataset will be used to run machine-learning algorithms. This dataset contains 51,295 associated URLs. Initially, the jasmine directory and the top 20 real websites from Alexa's list of URLs are used as a dataset. The dataset is cleaned by excluding the non-responsive web pages. The performance metrics are precision, recall and f1 and accuracy defined as follows:

$$\text{Precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}},$$

$$\text{Recall} = \frac{\text{true positive}}{\text{true positive+false negative}},$$

$$\text{F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision+recall}}.$$

Macro-average is the harmonic mean of the precision, recall and F1 score. Macro-average is computed to know the overall performance of the system with various sets of data. Varied values of testing and training have been used. Once the URL is correctly classified, the next task is to fill

Table 4 Status code and their description

Status code	Description
200	OK
400	Bad request response status code
401	Unauthorized
403	Forbidden client error status response code
404	Page not found
405	Method not allowed response status code
413	Payload too large
414	URI too long response status code
500	Internal server error
503	Service unavailable
524	A time out occurred

the form values with correct values. k nearest neighbor and SVM classifier is implemented to check the accuracy of the form submission. The submission status 200 shows that the system had submitted the form (Table 4).

The analysis of the status code is required because when the crawler will submit the web page, only the correct submission will yield a new URL. These URLs can be used for further analysis. Tables 5 and 6 show the number of forms submitted using GET and POST methods. Under these two methods, status code 524 has only one submission using the GET method and one using the POST method. From the total harvested URLs, it is observed that only two URLs correspond to status code 524. This are very little data to analyze for the machine-learning algorithm. The total number of web pages with status code 200 is 14033, it makes the harvest rate 27%. Table 7 shows the computation of precision, recall and F1 score using KNN algorithm for varied values of K. Table 8 shows the computation of precision, recall and F1 score using SVM for variation of 20–50% of testing data. Table 9 shows the computation of precision, recall and F1 score using KNN for variation of 20–50% testing data. On comparing the values of Tables 8 and 9, results are more promising for $k=5$. Table 9 shows that for $k=5$ in KNN, the value of accuracy is high as compared to the SVM algorithm. The optimal values are obtained at $k=5$ and 40/60 ratio of training and testing data.

Table 7 shows that the weighted average of precision is more when there is a 40/60 ratio of testing and training data. But the values of the weighted F1 score is more promising in the case of $k=5$. The ratio of testing and training data is tested for other values of k as well, but we found our approach working well for $k=5$. The results for $k=2$ and $k=5$ is presented, while others are skipped due to space constraints.

In the case of SVM, Table 6 shows the value of precision and recall when testing and training data ratio is 20/80, 30/70, 40/60 and 50/50. The weighted average of F1 is the same for 30%, 40% and 50%.

Table 10 shows computation of precision, recall and F1 score using KNN for variation of 20–50% of testing data for $K=5$.

Table 5 The number of forms submitted using the POST method

Domain	Number of URLs	200	400	401	403	404	405	413	414	500	503	524
Book	9741	3078	0	0	0	13	2285	893	0	0	3471	1
Product	2	2	0	0	0	0	0	0	0	0	0	0
Auto	29	29	0	0	0	0	0	0	0	0	0	0
Flight	723	183	0	0	0	0	0	0	0	0	540	0
Hotel	0	0	0	0	0	0	0	0	0	0	0	0
Music	0	0	0	0	0	0	0	0	0	0	0	0
Premier	0	0	0	0	0	0	0	0	0	0	0	0

Table 6 The number of forms submitted using the POST method

Domain	Number of URLs	200	400	401	403	404	405	413	414	500	503	524
Book	7677	6891	3	24	24	450	0	21	25	2	236	1
Product	633	494	0	0	24	115	0	0	0	0	0	0
Auto	99	29	9	0	0	21	0	0	3	2	35	0
Flight	4422	2892	254	0	0	404	0	26	57	12	540	0
Hotel	959	398	0	0	17	452	0	0	15	50	27	0
Music	84	35	10	0	0	0	0	6	2	0	31	0
Premier	12	2	0	0	10	0	0	0	0	0	0	0

Table 7 Computation of precision, recall and F1 score using KNN algorithm for varied values of K

Status code	Precision				Recall				F1 score			
	$K=2$	$K=3$	$K=4$	$K=5$	$K=2$	$K=3$	$K=4$	$K=5$	$K=2$	$K=3$	$K=4$	$K=5$
200	0.86	0.88	0.88	0.87	0.96	0.94	0.95	0.95	0.91	0.91	0.91	0.91
400	0.36	0.33	0.44	0.44	0.28	0.25	0.22	0.29	0.27	0.29	0.31	0.35
401	1.00	1.00	1.00	0.86	1.00	0.65	0.79	0.60	0.78	0.79	0.72	0.71
403	0.94	0.96	0.82	0.98	0.90	0.97	0.88	0.93	0.83	0.96	0.84	0.96
404	0.76	0.77	0.76	0.78	0.60	0.62	0.60	0.59	0.67	0.69	0.70	0.67
405	0.69	0.71	0.71	0.73	0.91	0.80	0.92	0.83	0.77	0.73	0.82	0.78
413	0.25	0.23	0.34	0.25	0.06	0.15	0.10	0.13	0.16	0.19	0.17	0.17
414	0.17	0.25	0.00	1.00	0.08	0.24	0.00	0.19	0.12	0.28	0.25	0.32
500	1.00	1.00	0.00	0.00	1.00	0.11	0.00	0.00	0.22	0.18	0.10	0.00
503	0.94	0.91	0.90	0.92	0.82	0.86	0.86	0.86	0.87	0.89	0.88	0.89
524	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Macro-average	0.72	0.73	0.51	0.65	0.61	0.58	0.53	0.53	0.62	0.60	0.62	0.56
Weighted average	0.91	0.91	0.91	0.92	0.92	0.92	0.92	0.93	0.91	0.92	0.91	0.92

Table 8 Computation of precision, recall and F1 score using SVM for variation of 20–50% of testing data

Status code	Precision				Recall				F1 score			
	20%	30%	40%	50%	20%	30%	40%	50%	20%	30%	40%	50%
200	0.79	0.80	0.78	0.79	0.92	0.95	0.96	0.97	0.82	0.87	0.86	0.87
400	0.00	0.75	0.00	0.00	0.00	0.06	0.00	0.14	0.00	0.12	0.00	0.23
401	1.00	0.00	0.00	0.78	0.20	0.00	0.00	0.00	0.33	0.00	0.00	0.00
403	0.83	0.71	0.00	0.77	0.35	0.68	0.00	0.78	0.49	0.70	0.00	0.78
404	0.66	0.71	0.76	0.73	0.59	0.54	0.49	0.35	0.44	0.62	0.59	0.48
405	0.71	0.71	0.68	0.65	1.00	1.00	0.99	1.00	0.62	0.00	0.81	0.84
413	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.83	0.00	0.00	0.05
414	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
500	0.00	0.00	0.00	0.92	0.00	0.00	0.69	0.68	0.00	0.00	0.00	0.78
503	0.81	0.90	0.91	0.00	0.67	0.68	0.00	0.00	0.73	0.78	0.78	0.00
524	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Macro-average	0.53	0.41	0.38	0.49	0.42	0.41	0.38	0.39	0.44	0.41	0.39	0.39
Weighted average	0.89	0.90	0.99	0.90	0.90	0.90	0.99	0.80	0.87	0.88	0.88	0.88

Table 9 Computation of precision, recall and F1 score using KNN for variation of 20–50% of testing data for $K=2$

Status code	Precision				Recall				F1 score			
	20%	30%	40%	50%	20%	30%	40%	50%	20%	30%	40%	50%
200	0.79	0.80	0.78	0.79	0.92	0.95	0.96	0.97	0.82	0.87	0.86	0.87
400	0.00	0.75	0.00	0.00	0.00	0.06	0.00	0.14	0.00	0.12	0.00	0.23
401	1.00	0.00	0.00	0.78	0.20	0.00	0.00	0.00	0.33	0.00	0.00	0.00
403	0.83	0.71	0.00	0.77	0.35	0.68	0.00	0.78	0.49	0.70	0.00	0.78
404	0.66	0.71	0.76	0.73	0.59	0.54	0.49	0.35	0.44	0.62	0.59	0.48
405	0.71	0.71	0.68	0.65	1.00	1.00	0.99	1.00	0.62	0.00	0.81	0.84
413	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.83	0.00	0.00	0.05
414	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
500	0.00	0.00	0.00	0.92	0.00	0.00	0.69	0.68	0.00	0.00	0.00	0.78
503	0.81	0.90	0.91	0.00	0.67	0.68	0.00	0.00	0.73	0.78	0.78	0.00
524	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Macro-average	0.53	0.41	0.38	0.49	0.42	0.41	0.38	0.43	0.44	0.41	0.39	0.39
Weighted average	0.89	0.90	0.90	0.90	0.90	0.90	0.99	0.88	0.87	0.88	0.88	0.88

Table 10 Computation of precision, recall and F1 score using KNN for variation of 20–50% of testing data for $K=5$

Status code	Precision				Recall				F1 score			
	20%	30%	40%	50%	20%	30%	40%	50%	20%	30%	40%	50%
200	0.80	0.86	0.86	0.85	0.96	0.95	0.96	0.95	0.87	0.90	0.91	0.90
400	0.00	0.51	0.43	0.57	0.00	0.26	0.22	0.24	0.00	0.34	0.29	0.34
401	0.00	0.93	0.90	0.75	0.00	0.70	0.73	0.71	0.00	0.80	0.81	0.73
403	0.66	0.85	0.83	0.78	0.71	0.82	0.63	0.60	0.69	0.83	0.71	0.68
404	0.72	0.78	0.81	0.79	0.46	0.57	0.55	0.57	0.56	0.66	0.66	0.60
405	0.72	0.72	0.73	0.72	0.98	0.83	0.81	0.85	0.83	0.77	0.77	0.78
413	0.00	0.26	0.26	0.33	0.00	0.13	0.14	0.16	0.00	0.17	0.18	0.22
414	0.00	1.00	0.46	0.50	0.00	0.18	0.13	0.05	0.00	0.30	0.21	0.10
500	0.00	0.00	0.25	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.06	0.00
503	0.91	0.89	0.92	0.88	0.69	0.81	0.85	0.80	0.78	0.85	0.88	0.84
524	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Macro-average	0.44	0.71	0.57	0.55	0.44	0.57	0.47	0.46	0.43	0.60	0.50	0.48
Weighted average	0.88	0.91	0.91	0.91	0.90	0.92	0.92	0.91	0.88	0.91	0.91	0.91

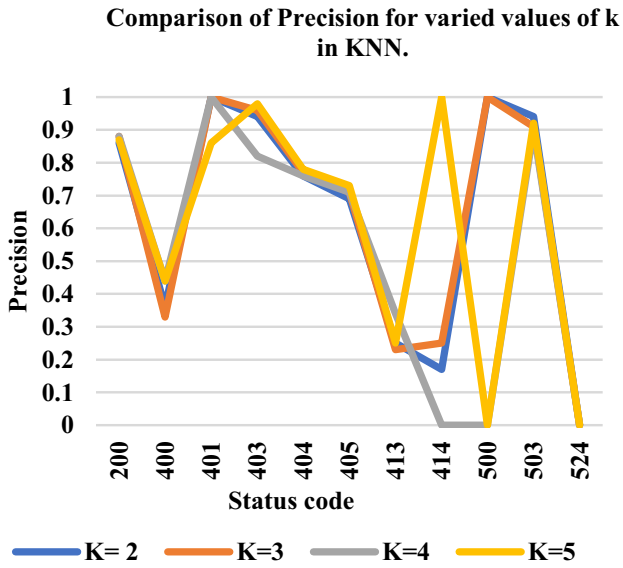

Fig. 5 Comparison of precision for varied values of K in KNN

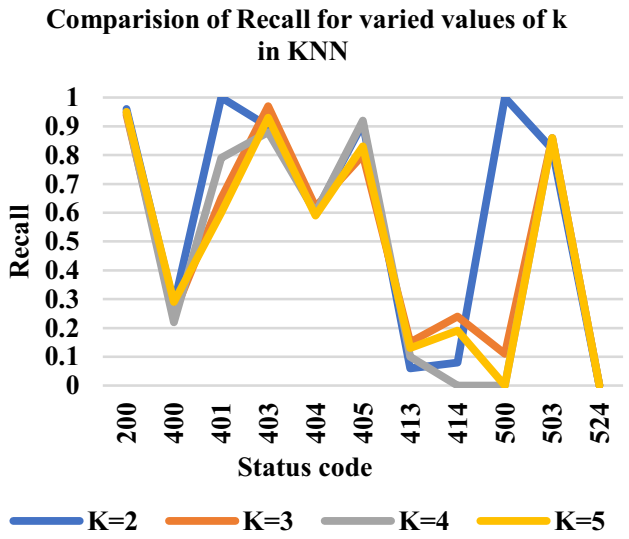

Fig. 6 Comparison of recall for varied values of K in KNN

The following figures show the experimental results in graphical forms. Figures 5 and 6 show a comparison of precision and recall in KNN for $k=2$–5. In Figs. 12, 16 and 18, the ratio is shown in decimal notation, i.e., 20/100 is 0.2 (Table 11).

Figures 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, and 18 conclude that for the proposed approach, KNN has performed better than SVM. Figure 19 shows that ICHW has a high harvest rate in contrast to its pioneer contemporaries. The values of the status code as shown in Figs. 5, 6, 7, 8, 9, 10, 11, 12, and 13 show that system has correctly classified the forms as well as submit them. The values of forms for submissions are retrieved from the bounded values of form during parsing. Results are also shown for the ratio of testing and training data. For this approach for $k=5$ at 40% of testing, data gave promising results. On being compared with a focused crawler (FC), form-focused crawler (FFC), and

Table 11 Comparison of accuracy for KNN and SVM

Accuracy					
KNN					SVM
Percentage of testing data	$K=2$	$K=3$	$K=4$	$K=5$	
20%	0.89	0.88	0.89	0.9	0.89
30%	0.88	0.89	0.88	0.92	0.90
40%	0.90	0.80	0.90	0.92	0.90
50%	0.90	0.89	0.90	0.91	0.90

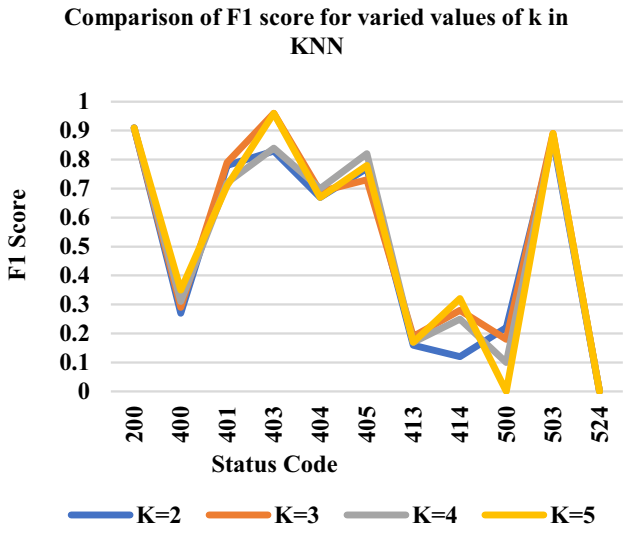

Fig. 7 Comparison of F1 score for varied values of *K* in KNN

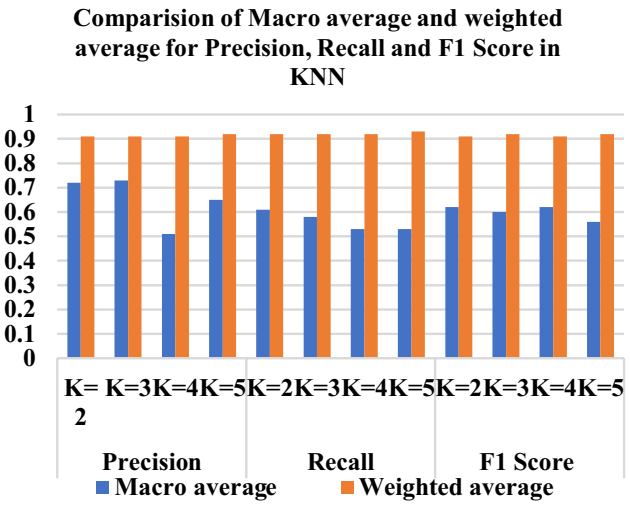

Fig. 8 Comparison of macro-average and weighted average for precision, recall and F1 in KNN

Enhanced form crawler (EEFC). ICHW has a more than 10% high harvest rate than EEFC. There exist only a few crawlers that implement both pre-query and post-query approaches, ICHW also worked on both techniques. The rejection rules and stopping criteria have impacted the harvest rate of the crawler. In the training phase, the space complexity of KNN is $O(n*d)$, while in the testing phase it is $O(n*k*d)$. n represents a number of data points, d represents a number of features and k represents the number of nearest neighbors considered. For SVM, complexity is $O(n^3)$.

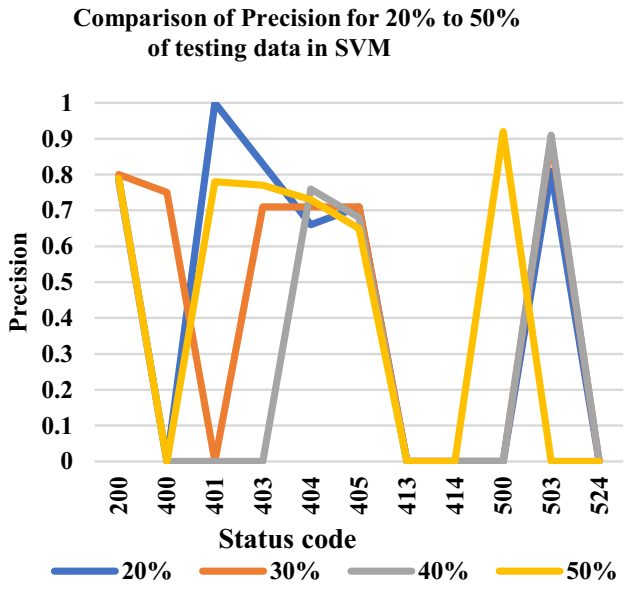

Fig. 9 Comparison of precision for 20–50% of testing in SVM

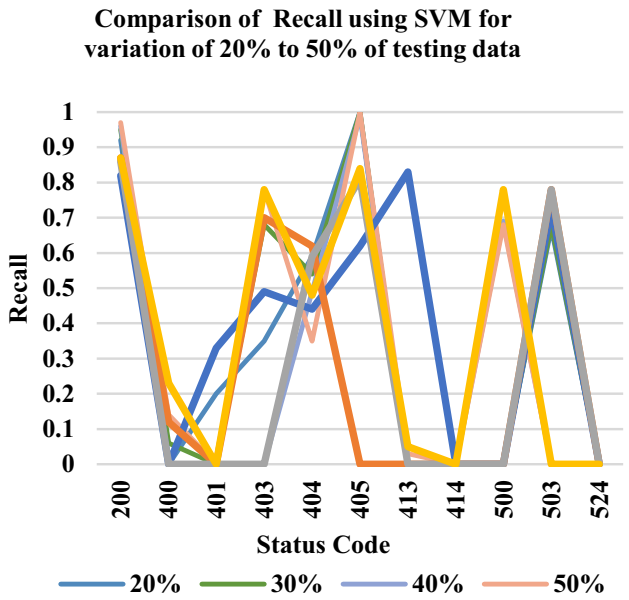

Fig. 10 Comparison of recall for varied values of *K* in KNN

ICHW as an approach for atmospheric emission

Suppose the user has a goal to find a property with a good air quality index. Given f (Amritsar, Punjab), (Ludhiana, Punjab), (Jalandhar, Punjab) be the three cities for which search is targeted. Instead of using three different crawling nodes, the crawling is implemented as three different threads for each tuple. Let us assign C1 = (Amritsar, Punjab), C2 = (Ludhiana, Punjab), and C3 = (Jalandhar, Punjab). The location-based subdivisions of the cities are taken as

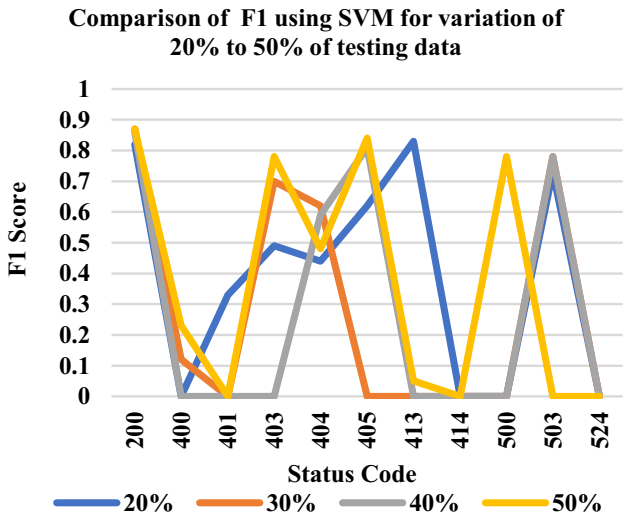

Fig. 11 Comparison of F1 using SVM for variation of 20–50% of testing data

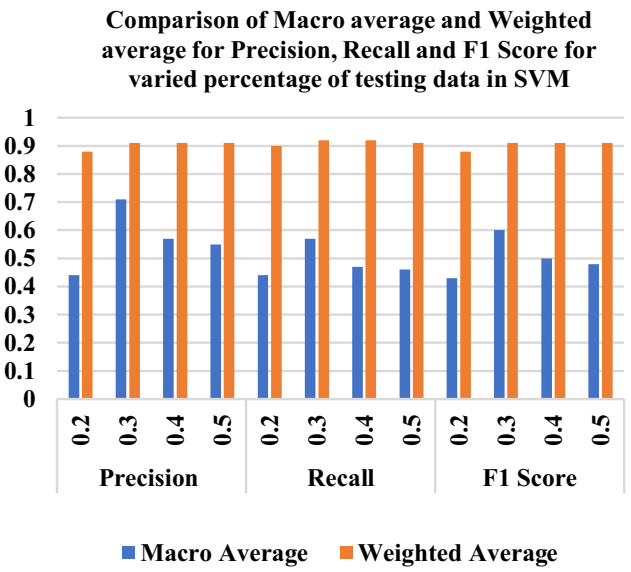

Fig. 12 Comparison of macro-average and weighted average for precision, recall and F1 score for varied percentage of testing data in SVM

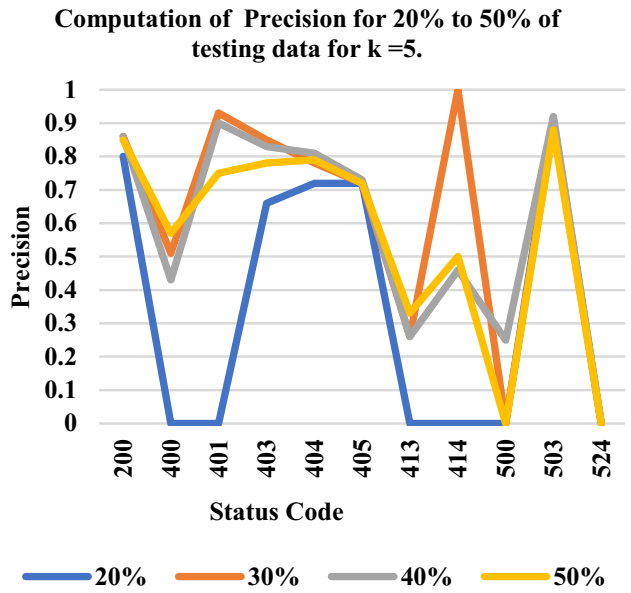

Fig. 13 Computation of precision for 20–50% of testing data for $K=5$

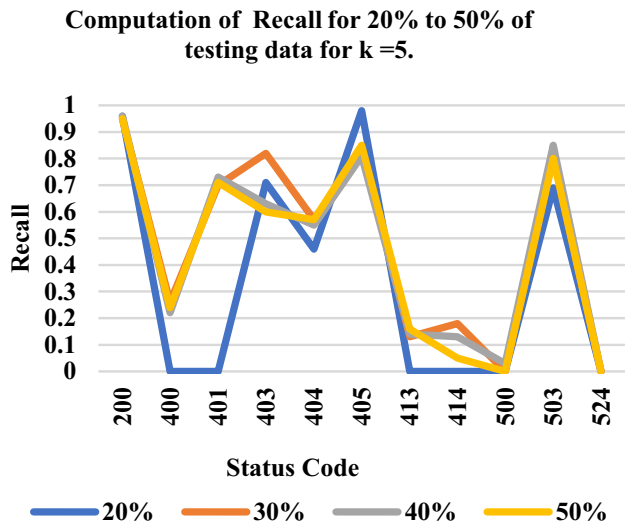

Fig. 14 Computation of recall for 20–50% of testing data for $k=5$

the administrative divisions. Amritsar and Jalandhar have five administrative divisions whereas Ludhiana has seven administrative divisions. Location-based crawling is done on these administrative divisions. Crawled data are combined for average pollution in each city. The goal is to find PM 10 and PM 2.5 values in administrative divisions. The crawler will crawl and parse the data from the real estate website, and combine this with location-aware crawling. The traversing of the crawler is controlled using rejection rules.

The results will be useful for making the right investment in a property based on qualitative, relevant and empirical data. In addition, suppose if a user is already living in any of the above-mentioned cities, crawling using this web crawler will help find similar properties and set a good value on their own. User can also search for fair deals. Due to space constraints, the results regarding the submission of the form regarding each feature are not presented, moreover, most of the URLs belongs to the dynamic databases. Data are combined from both real estate and pollution URLs, by implementing the expectation maximum clustering technique using the Gaussian mixture model. Data normalization is performed using MAX–MIN normalization. In this case,

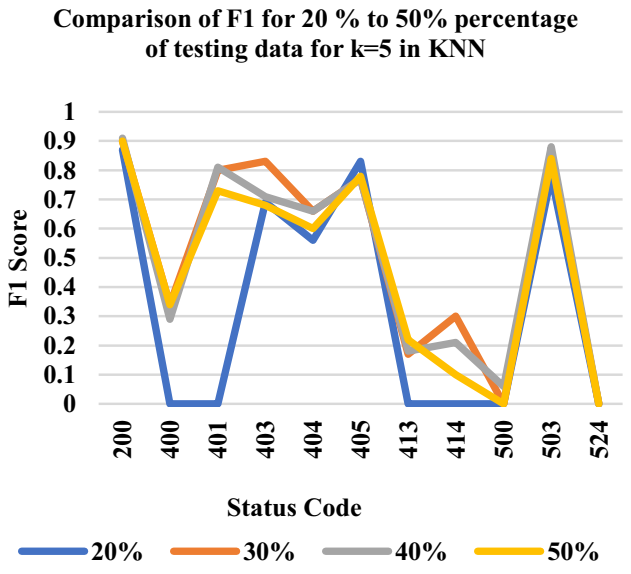

Fig. 15 Comparison of F1 score for 20–50% of testing data

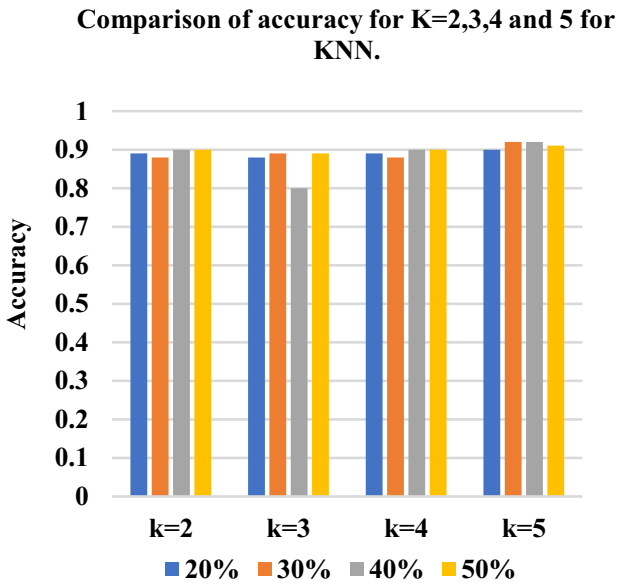

Fig. 17 Comparison of accuracy for $K = 2$, 3, 4 and 5 for KNN

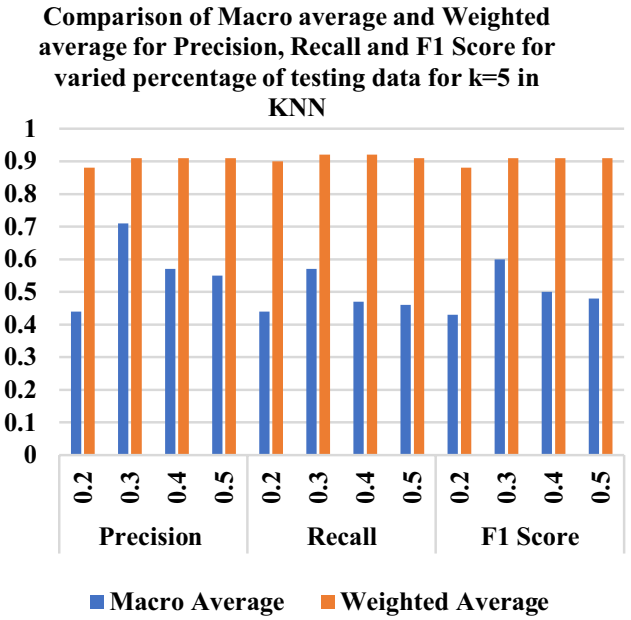

Fig. 16 Comparison of macro-average, weighted average for precision, recall and F1 score in SVM

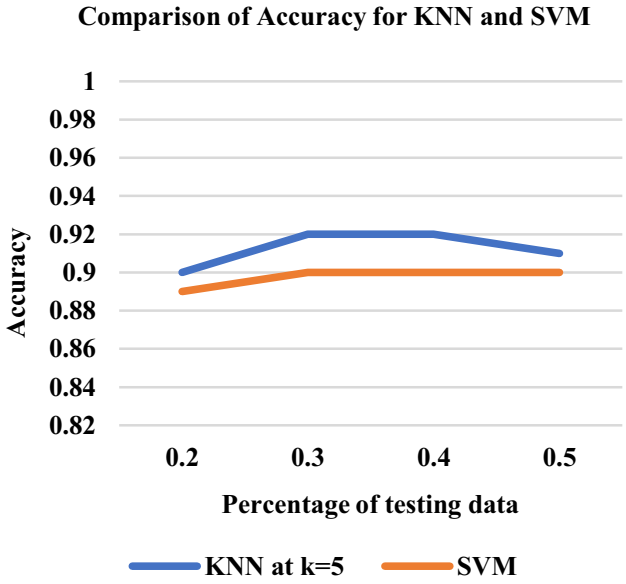

Fig. 18 Comparison of accuracy for KNN and SVM

the expectation–maximization algorithm is implemented to find parameters. The parameters are defined as: M denotes the sample of data points, μ is the Gaussian distribution, Σ covariance, u is defined as input vector, 'I' denotes possible curves, 'i' denotes data points, C is the Gaussian curve, w_{ij} is the weighting factor of a feature vector, π denotes Gaussian weight, ∂ is the standard deviation and m is a number of data points in dataset. Derivation of likelihood is as follows: let θ be the random variable with binary values

$$\theta = P(I) \tag{27}$$

$$I - \theta = P(0) \tag{28}$$

The likelihood is defined as

$$l(\theta) = \theta^{n_1}(I-\theta)^{n_0} \tag{29}$$

Taking derivative on both sides of Eq. (29):

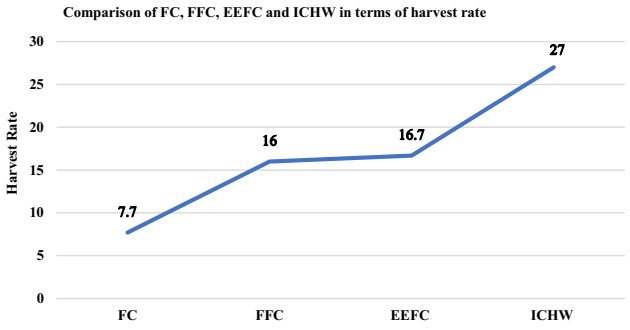

Fig. 19 Comparison of FC, FFC, EEFC and ICHW in terms of harvest rate

$$\frac{\partial^2 l(\theta)}{\partial(\theta)} = n_1\theta^{n-1}(I-\theta)^{n_0} - n_0\theta^{n_1}(I-\theta)^{n_0-1} \tag{30}$$

$$= \theta^{n-1}(I-\theta)^{n_0-1}\left(n_1(I-\theta) - n_0\theta\right) \tag{31}$$

$$= \theta^{n-1}(I-\theta)^{n_0-1}(n_1\left(n_1+n_0\right)\theta \tag{32}$$

If $\theta = 0$, or $\theta = 1$:

$\theta = \frac{n_1}{n_0+n_1}$.

Let M data samples be denoted as M_1, M_2, M_3,…..,M_n, the maximum likelihood for the Gaussian model is derived as

$$\log l\,(\mu,\sigma) = \sum_{i=1}^{m}\left(\frac{1}{\sqrt{2\Pi}}e\frac{-(x-\mu)^2}{2\sigma^2}\right) \tag{33}$$

$$= C + \sum_{i=1}^{m} - \log l - \frac{(x^{(i)}-\mu)^2}{2\sigma} \tag{34}$$

$$\frac{\partial \log l(\mu,\sigma)}{\partial\mu} = \frac{1}{\sigma^2}\sum_{i=1}^{m}(x^i-\mu) \tag{35}$$

$$= \sum_{i=1}^{m}\frac{1}{\sigma} - \frac{(x^{(i)}-\mu)^2}{\sigma^3} \tag{36}$$

$$\sigma^2\mathrm{Ml} = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu_{\mathrm{Ml}})^2 \tag{37}$$

Now estimation maximization for the Gaussian model is derived as follows. Suppose Y is multinomial distribution,

$$P(Y=k;\theta) = \mu_k \tag{38}$$

$$T \sim N\left(\mu_k, \sum k\right) \tag{39}$$

$$p\left(x=k,T;\theta,\mu,\sum\right) = \theta_k\frac{1}{(2\pi)^{\frac{n}{2}}\left|\sum k\right|^{\frac{1}{2}}}e\frac{-1}{2}\left(z-\mu_k\right)^{T\sum_K^{-1}(z-\mu_k)} \tag{40}$$

Expectation calculation:

$$p\left(x|z;\theta,\mu,\ \sum\right) = \prod_{i=1}^{m}p\left(x^{(i)}\middle|z^i;\theta,\ \mu,\ \sum\right) \tag{41}$$

Maximization calculation:

$$\max_{\theta,\mu,\sum}\sum_{i=1}^{m}\sum_{i=1}^{m}\sum_{k=1}^{k}q(x^{(i)}=k)\log\left(\theta_k\mathcal{N}(z^{(i)};\mu_k\right) \tag{42}$$

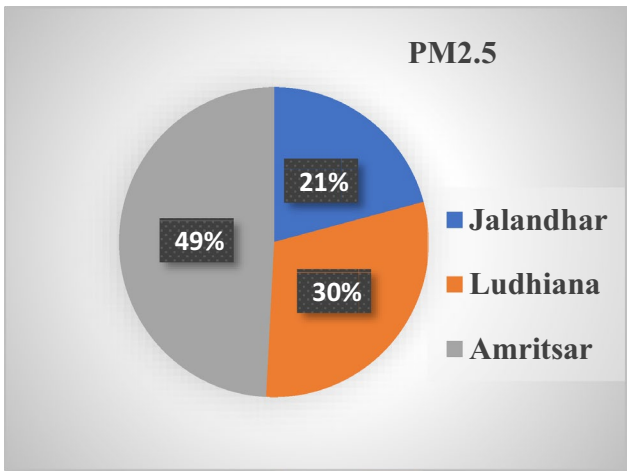

Fig. 20 Comparison of PM 2.5 in cities of Punjab

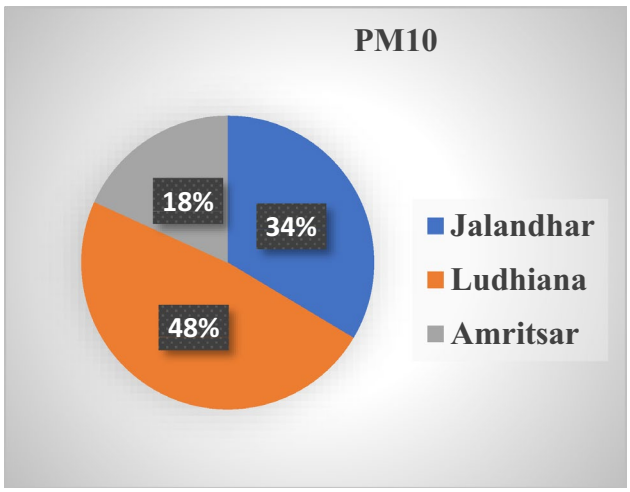

Fig. 21 Comparison of PM 10 in cities of Punjab

Table 12 Comparison of running time and number of searchable forms

Domain	Running time of ACHE	Running time of ICHW without rejection rules	Running time of ICHW	Searchable form ACHE	Searchable form ICHW
Property	Not included	7 h 12 m	6 h 39 m	Not included	3809
Book	8 h 21 m	7 h 21 m	6 h 58 m	599	4589
Flight	7 h 59 m	6 h 18 m	7 h 52 m	1705	2843
Music	7 h 59 m	7 h 00 m	6 h 58 m	776	1447
Premier	Not included	6 h 35 m	6 h 01 m	Not included	668
Product	7 h 50 m	7 h 28 m	7 h 48 m	386	1999
Pollution	Not included	7 H 26 M	6 H 20 M	Not included	2002

After applying the above-discussed technique, clusters of regions are formed according to the air quality index in Amritsar, Jalandhar and Ludhiana (Figs. 20, 21).

The further analysis could be made on reason of low air quality index. Due to space constraints, tabular form of data is not presented, and the above figures have shown the computed results of air quality in the three cities.

Comparative advantages

The proposed technique is one of its kind works that associate real estate data and air quality index to find property in smart cities of Punjab. The crawler can be trained to be used for any other search terms. The results have shown that the proposed approach has a high harvest rate as compared to existing techniques. This approach is scalable in terms of the growing size of the web, and it is extensible as any third-party component for example indexer can be added. The ranking is a function of both out links and term weighting, due to which chances of term bias is less. The crawler successfully saves itself from the crawler traps due to efficient stopping criteria's and accurately classify more status codes than [41]. The F1 measure of the proposed technique is higher than [42], as this technique has also implemented text clustering. Another advantage of this technique is that it works with both GET and Post methods. In this way, the crawler can have a high number of URLs for analysis and indexing. Table 12 compares the running time and number of searchable forms of adaptive crawler for hidden web entries (ACHE) and ICHW. There exists no technique as perfect that it can stop a crawler to fall into the spider traps. Therefore, the intelligent rules of rejection are designed to prevent the crawler from falling into infinite crawling loops. This technique outperforms the web crawler presented in [43]. On comparing accuracy and recall, in testing phase crawler in [43] has accuracy 81.06% and precision 84.62%, while both performance measures have reached above 95% in technique. Figures 9 and 10 show the comparison of precision and recall delivered by the proposed crawler. Harvest rate our proposed system is more than [44] and [45], but their technique has also implemented indexing. Indexing is part of our future work.

The above table shows the running time of ICHW is comparatively less than ACHE. In addition, the number of searchable forms founds are more than ACHE. A goal of a crawler is to find maximum searchable forms in minimum visits, so the number of searchable forms without rejection rules are not included.

Conclusion

ICHW crawler simultaneously works with all stages in hidden web crawling. The dual objective is fulfilled by efficiently searching the hidden web sources, and then minimizing the visit and saving the crawler resources by proposing rejection rules. This study shows the successful implementation of a web crawler for combining real estate data and pollution data in smart cities of Punjab. The crawler is effective in both applications. By implementing path learning and similarity, the crawler can correctly judge the form-based data. The intelligent stopping criteria are introduced to minimize the unproductive crawling. The retrieved hidden web page classification is addressed by considering URL filtration beyond just <form> tags. A knowledge base of suitable values extracted is created to accurately fill the forms. Experiments results show not only the harvest rate of ICHW is appreciable, but it is also able to accurately crawl PM 10 and PM 2.5 related data. Future work includes working with a larger data set, a large number of classes, advanced stopping criteria, and the use of geospatial data for air quality index and a user interface of the crawler.

References

1. Kobayashi M, Takeda K (2000) Information retrieval on the Web. ACM Comput Surv 32(2):144–173
2. Wu M, Lee C (2020) A study on natural language processing classified news. pp. 244–247
3. Chakrabarti S (2003) Crawling the web. In: Mining the web, pp 17–43
4. Kaur S, Geetha G (2007) Advances in web crawlers, vol. 10, pp. 1–22
5. Li Y, Wang Y, Du J (2013) E-FFC: an enhanced form-focused crawler for domain-specific deep web databases. J Intell Inform Syst 40(1):159–184
6. Wu Z et al (2003) Towards automatic incorporation of search engines into a large-scale metasearch engine. In: Proceedings - IEEE/WIC International Conference on Web Intelligence, WI 2003, pp 658–661
7. Madhavan J et al (2007) Web-scale data integration: you can only afford to pay as you go. Cidr 7:342–350
8. Cope J, Craswell N, Hawking D (2003) Automated discovery of search interfaces on the web BT. In: Fourteenth Australasian Database Conference (ADC2003), vol. 17, pp. 181–189
9. Barbosa L, Freire J (2005) Searching for hidden-web databases. Proc WebDB 5:1–6
10. Hicks C, Scheffer M, Ngu AHH, ShengQZ (2012) Discovery and cataloging of deep web sources. In: Proceedings of the 2012 IEEE 13th International Conference on Information Reuse and Integration, IRI 2012, pp. 224–230
11. Raghavan S, Garcia-molina H (2001) Crawling the hidden web. In: 27th VLDB Conference, Roma, Italy, pp 1–10
12. Perkowitz M, Doorenbos RB, Etzioni O, Weld DS (1997) Learning to understand information on the internet: an example-based approach. J Intell Inform Syst 8(2):133–153
13. Liddle S, Embley D, Scott D, Yau SH (2003) Extracting data behind web forms. Lect Notes Comput Sci 2784:402–413
14. He Y, Xin D, Ganti V, Rajaraman S, Shah N (2013) Crawling deep web entity pages. Web Search Data Min, 355–364
15. Furche T, Gottlob G, Grasso G, Schallhart C, Sellers A (2013) OXPath: a language for scalable data extraction, automation, and crawling on the deep web. VLDB J 22(1):47–72
16. Álvarez M, Raposo J, Pan A, Cacheda F, Bellas F, Carneiro V (2007) Crawling the content hidden behind web forms. In: Lecture notes in computer science, pp. 322–333
17. Dragut EC. Deep web query interface understanding and integration
18. Khare R, An Y, Song I-Y (2010) Understanding deep web search interfaces. Survey 39(1):33–40
19. Jamali M, Sayyadi H, Hariri BB, Abolhassani H (2006) A method for focused crawling using combination of link structure and content similarity. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings), WI'06, pp. 753–756
20. Chakrabarti S, Van Den Berg M, Dom B (1999) Focused crawling: a new approach to topic-specific web resource discovery. Comput Netw 31(11):1623–1640
21. Barbosa L, Freire J (2007) An adaptive crawler for locating hiddenwebentry points. In: Proceedings of the 16th international conference on World Wide Web—WWW '07, p. 441
22. Najork M, Wiener JL (2001) Breadth-first search crawling yields high-quality pages. In: Proceedings of the 10th International Conference on World Wide Web, WWW 2001, pp. 114–118
23. Korf RE, Schultze P (2005) Large-scale parallel breadth-first search. In: Proceedings of the National Conference on Artificial Intelligence, vol. 3, pp. 1380–1385
24. Peshave M, Dezhgosha K (2005) How search engines work and a web crawler application
25. Wang W, Chen X, Zou Y, Wang H, Dai Z (2010) A focused crawler based on naive Bayes classifier. In: 3rd International Symposium on Intelligent Information Technology and Security Informatics, IITSI 2010, pp 517–521
26. Zheng X, Zhou T, Yu Z, Chen D (2008) URL rule based focused crawlers. In: IEEE International Conference on e-Business Engineering, ICEBE'08—Workshops: AiR'08, EM2I'08, SOAIC'08, SOKM'08, BIMA'08, DKEEE'08, pp. 147–154
27. Barbosa L, Freire J. Searching for hidden-web databases
28. Barbosa L, Freire J (2007) An adaptive crawler for locating hidden web entry points. In: 16th International World Wide Web Conference, WWW2007, pp. 441–450
29. Madhavan J et al (2007) Web-scale data integration: you can only afford to pay as you go. In: Proceedings of CIDR, pp. 342–350
30. Zhao F, Zhou J, Nie C, Huang H, Jin H (2016) SmartCrawler: a two-stage crawler for efficiently harvesting deep-web interfaces. IEEE Trans Serv Comput 9(4):608–620
31. Jou C (2019) Schema extraction for deep web query interfaces using heuristics rules. Inf Syst Front 21(1):163–174
32. Tsikrika T, Moumtzidou A, Vrochidis S, Kompatsiaris I (2016) Focussed crawling of environmental web resources based on the combination of multimedia evidence. Multimed Tools Appl 75(3):1563–1587
33. Helfenstein A, Tammela P (2016) Data and text mining analyzing user-generated online content for drug discovery: development and use of Med-crawler, pp. 0–5
34. Dong H, Hussain FK (2011) Focused crawling for automatic service discovery, annotation, and classification in industrial digital ecosystems. IEEE Trans Ind Electron 58(6):2106–2116
35. Lopez-Aparicio S, Grythe H, Vogt M, Pierce M, Vallejo I (2018) Webcrawling and machine learning as a new approach for the spatial distribution of atmospheric emissions. PLoS ONE 13(7):1–15
36. Chang KC-C, He B, Li C, Patel M, Zhang Z (2004) Structured databases on the web. ACM SIGMOD Rec 33(3):61
37. Kaur S, Geetha G (2020) SIMHAR—smart distributed web crawler for the hidden web using SIM+Hash and Redis server. IEEE Access 8:117582–117592
38. Teixeira PM (2018) Relevance ranking for predicting web search results, vol 1
39. Valkanas G, Ntoulas A (2011) Rank-aware crawling of hidden web sites. In: WebDB, pp 1–6
40. Haveliwala TH, Gionis A, Klein D, Indyk P (2002) Evaluating strategies for similarity search on the web. In: Proceedings of the 11th International Conference on World Wide Web, WWW '02, pp 432–442
41. Heydon A, Najork M (1999) Mercator: a scalable, extensible web crawler. World Wide Web 2(4):219–229
42. Sangaiah AK, Fakhry AE, Abdel-Basset M, El-henawy I (2019) Arabic text clustering using improved clustering algorithms with dimensionality reduction. Clust Comput 22:4535–4549
43. Hien NLH, Tien TQ, Van Hieu N (2020) Web crawler: design and implementation for extracting article-like contents. Cybern Phys 9(3):144–151

44. Schedlbauer J, Raptis G, Ludwig B (2021) Medical informatics labor market analysis using web crawling, web scraping, and text mining. Int J Med Inform 150:104453
45. Sharma A, Shrivastava V, Singh H (2021) Experimental performance analysis of web crawlers using single and multi-threaded web crawling and indexing algorithm for the application of smart web contents. Mater Today: Proc 37(2):1403–1408