Learning Deep Web Crawling with Diverse Features

Lu Jiang, Zhaohui Wu, Qinghua Zheng and Jun Liu
School of Electronic and Information Engineering
Xi’an Jiaotong University
Xi’an, P.R. China
roadjiang@yahoo.com, wzh@stu.xjtu.edu.cn, qzheng@mail.xjtu.edu.cn, liukeen@mail.xjtu.edu.cn

Abstract—The key to Deep Web crawling is to submit promising keywords to query form and retrieve Deep Web content efficiently. To select keywords, existing methods make a decision based on keywords’ statistic information deriving from TF and DF in local acquired records, thus work well only in textual databases providing full text search interfaces, whereas not well in structured databases of multi-attribute or field-restricted search interfaces. This paper proposes a novel Deep Web crawling method. Keywords are encoded as a tuple by its linguistic, statistic and HTML features so that a harvest rate evaluation model can be learned from the issued keywords for the un-issued in future. The method breaks through the assumption of plain-text search made by existing methods. Experimental results show that the method outperforms the state of the art methods.

Keywords—Hidden Web; Deep Web surfacing; machine learning

I. INTRODUCTION

Deep Web refers to World Wide Web content that is not part of the surface Web, which is directly indexed by search engines. Studies [1] showed that Deep Web content is particularly important. However, to obtain such content is challenging and has been acknowledged as a significant gap in the coverage of search engines [2]. To surface it, a crawler can pre-compute the submissions for Deep Web forms and exhaustively index the response results off-line, just as other static HTML pages. The approach enables leveraging the existing search engine infrastructure hence adopted by most of crawlers, such as HiWE (Hidden Web Exposer) [3], Hidden Web crawler [4] and Google’s Deep Web crawler [2].

There are two main challenges in Deep Web surfacing. First in order to find the entry of the Deep Web database, the crawler has to detect query forms iteratively among millions of potential forms on the internet. Second given the query form to a Deep Web database, the crawler should automatically generate promising keywords and submit them to the database. In this paper, we concentrate on the second challenge i.e. how a crawler can automatically generate promising queries so that it can carry out efficient surfacing within affordable cost. The challenge has been studied by a couple of researchers such as Barbosa [5], Ntoulas [4], Jayant [2] and Liu [6]. In these methods, candidate query keywords are generated from the obtained records, and then their expected harvest rates, i.e. the capability to obtain new records, are calculated according to their local statistics, such as DF (Document Frequency) and TF (Term Frequency). The one with the maximum expected harvest rate will be selected for the next query. Their basic idea is the same while the difference is that they prefer different strategies to derive the expected harvest rate of each query candidate.

Existing methods work well in textual databases for plain-text documents where full text search interfaces are provided, whereas not well in structured databases of multi-attribute or field-restricted search interfaces. For the full text search, all of the words in every stored document are indexed [7] thus each word can be issued to query. The statistics deriving from TF and DF might reflect words’ promise to retrieve new content in this circumstance. However, this will not hold for the structured database since its response pages are relatively structured and of more HTML or XML information, leading to that only the statistic information based on TF and DF can hardly discriminate the promising words from the unpromising ones. Text content in HTML or XML pages is usually surrounded by tags, which serve as indicating the semantics of their values, especially for the XML response pages. To select query words for Deep Web submission in HTML pages, features such as keywords’ location, tag name, POS (Part Of Speech) may be significant. In Section III, we developed diversified features for encoding candidate query keywords.

To sum up, in this paper, we present a novel method for Deep Web crawling. Each word obtained from response pages is encoded as a tuple representing its linguistic, statistic and HTML features. The issued keywords are used to train a model using machine learning algorithm which will be used to evaluate the harvest rates for un-issued keywords. The method breaks through the assumption of plain-text search made by existing methods. The experimental results on real world Deep Web sites show that this method outperforms existing ones.

II. RELATED WORK

The state-of-the-art Deep Web crawling approaches generate new keywords by analyzing statistics of current records returned from previous queries. Barbosa L. et al. first introduced the ideas, and presented a query selection method which generated the next query using the most frequent keywords in the previous records [5]. However, queries with the most frequent keywords in hand do not ensure that more new records are returned from the Deep Web database. Ntoulas A. et al. proposed a greedy query selection method based on the expected harvest rate [4]. In the method,
candidate query keywords are generated from the obtained records, and then their harvest rates are calculated. The one with the maximum expected harvest rate will be selected for the next query. Jayant M. et al. improved the keyword selection algorithm by ranking keywords with respect to their TFIDF (Term Frequency Inverse Document Frequency) [2]. The newly introduced information of TF boost the method in terms of applicability since the crawler can get better understanding of the keywords. Liu J. et al. extended the Ntoulas’s method to entire form by introducing a novel concept MEP (Minimum Executable Pattern). In the method, a MEP set should be build and then promising keywords are selected by joint harvest rate of the keyword and its corresponding pattern. By selecting among multiple MEPs, the crawler achieves better results [6].

III. LEARNING CRAWLING WITH FEATURES

A. Overall Crawling Algororithm

To surfacing Deep Web in a practical way, the key issue is how to estimate the harvest rate given the current acquired records. To solve the problem, we conceive an adaptive algorithm for Deep Web crawling, shown in fig.1. The basic idea of the algorithm is to estimate harvest rates for unseen keywords generalizing from those previously experienced queries.

1: Select the keyword with maximum harvest rate from candidate set;
2: If termination condition is satisfied Exit;
   Else generate and issue a query to the database using the selected keyword;
3: Download and parse response pages then compute the harvest rate of the current query;
4: Renew the training set and the candidate set;
5: If error > s, retrain the learning model;
6: Estimate the harvest rate for each keyword in candidate set; Goto Step 1;

Figure 1. Adaptive algorithm for Deep Web crawling

To initialize the algorithm, the crawler has to add some seed keywords to the candidate set. Usually these words can be obtained from web pages of query forms. In step 2, the crawler hires the slide window termination policy [6] to terminate the crawling process.

The parse process in step 3 is to extract and segment all keywords appearing in new records and compute the harvest rate of the current query. The information is used to update the training and candidate set. The training set \( Tr \) is the set of tuple \((q, hr)\) of all issued queries where \( q \) is a query and \( hr \) is its harvest rate. Given the current query, candidate set \( C \) is the set of available keywords to be issued. Elements in \( Tr \) or \( C \) are encoded in Vector Space Model and main difference between them lies the \( hr \) in \( Tr \) is seen while in \( C \) is not.

The learning model will be updated when the accumulated square error reaches to the threshold \( s \) in step 5:

\[
\text{error} = \sum_{j=1}^{t} (hr_j - \hat{hr}_j)^2 , \quad (1)
\]

in which, \( hr_j \) is the harvest rate of \( j^{th} \) query whereas \( \hat{hr}_j \) is the estimated harvest rate by the learning model at \( j-1^{th} \) query. \( t \) is the query number when the latest learning model is trained and \( i \) is the current query number. Step 6 attempts to estimate the harvest rate for query-harvest rate tuples in candidate set by generalizing from those in training set.

B. Features of Keywords

This section focuses on how to encode a query-harvest rate pair \((q, hr)\), i.e., the feature representation procedure. To encode it, we present the three types of features, including linguistic feature, statistical feature and HTML feature.

Linguistic features.

POS (Part of Speech): This feature represents the part of speech of keywords, including noun, verb, adjective, adverb, numeral, interjection and etc. We divide these POSes into several groups and members of the same group are assigned the same feature value.

Length: This feature represents the length of keywords, i.e. number of characters in a keyword. Generally speaking, keyword whose length is less than 6 may be more rewarding. In the experiments, we find that when processing Deep web of Chinese or Japanese, the best length is 2 or 3. For the English Deep web, the length feature is trivial for reward.

Language: This feature represents the language that keywords fall into. It takes effect when the target Deep Web is multilingual. The feature can be recognized by the coding region of the code set. For example Chinese character coding region of Unicode is \([u4E00, u9FA0]\). Different languages are signed different values.

Statistical features.

TF: Term frequency of a keyword in the acquired records. It was firstly used by Barbosa L. in [5] for the query selection, based on the assumption that keywords of higher TF will be more rewarding.

DF: Document frequency of a keyword in the acquired records. Ntoulas A.’s approach [4] was mainly on the basis of this feature.

TFIDF: It [8] has been a popular Information Retrieval measure for balancing the word’s importance on page with its overall importance. Google’s Deep Web crawler introduced this measure for its seeds and candidate keywords selection [2].

RIDF (Residual IDF): The value is computed as

\[
\text{RIDF} = -\log \frac{DF}{|D|} + \log(1 - e^{-\frac{TP}{|D|}}) \quad (2)
\]

where \( D \) is the collection of the all records reside in the Deep Web Database. RIDF tends to highlight technical terminology, names, and good keywords for information retrieval which tend to exhibit nonrandom distributions over documents [9]. The paper first brings RIDF to the measure for Deep Web keywords selection.

HTML Feature.
The HTML format not only affects the visual appearance of the text and the document layout, but also plays an important role in indicating the semantics of the presented data.

**TAG**: This feature consists of the HTML tags and attributes information of a keyword. The feature may imply the semantic information of a keyword leading to decent capability of distinguishing unpromising words.

**Location**: It represents the location information of the keyword’s node in the DOM tree derived from the HTML document, which can be simply evaluated by depth of the node.

**Markedness**: The markedness of a keyword determines how much the word is highlighted comparing to the remaining text of the HTML document. The value $x$ can be computed as

$$x = (F \cdot \Delta f + b + o + u + c) \cdot (1 - z)$$

where $b$, $o$, $u$ and $z$ have the value 1 when the keyword is bold, oblique, underlined or striked-out respectively and 0 otherwise. Similarly, $c$ indicates whether the text element has a color deviated from the document default. $\Delta f = f - fd$ where $f$ is the font size of the element and $fd$ is the default font size for the document in points. The constant $F$ defines the relation between the text size and its markedness [10].

### IV. EXPERIMENTS

In experiments, we encoded a query-harvest rate pair using the features of POS, length, language, RIDF, tag, location and markedness. Tag values in response HTML pages were segmented into words and then extracted for their features mentioned above. Segmented English word was stemmed using the standard Porter suffix-stripping algorithm [11] and Chinese or Japanese words with a length more than 4 were discarded. The kNN [12] algorithm was employed to train a learning model. Obviously, it is not possible to compare the method with all methods mentioned above. However, we integrated the methods of [2], [4], [5], [6] as the Probing method and preserved their main ideas, in which a promising keyword is selected by its harvest rate calculated according to its DF and TF.

Three Deep Web sites in different scales and domains were chosen as the experimental sites. In the case of IMDB (Internet Movie Database) site (http://www.imdb.com), as its large scale, the crawler only restricted itself to crawling movies released theatrically to accelerate the experiment. The total number of which is approximately 438,000. Queries to the site were applied to the pattern “Names”. The second Deep Web site Siyuan Music Station (http://music.xjtu.edu.cn) as a medium scale multilingual website, maintains approximately 160,000 songs. In experiment the crawler first looked for singers and then retrieved all songs of them. Regarding Siyuan Video on Demand System (http://vod.xjtu.edu.cn), which is a small scale Deep Web site, collects approximately 1,200 Chinese and English movies. In experiment, the crawler targeted the search textbox “Movie Name” to discover movies.

#### A. Performance Comparision

Our interest is to discover their records as many as possible with the affordable cost. However, to keep our experiment and its analysis simple, we assume that the cost for every query is constant. In order to evaluate performance of our method, we conducted two experiments and displayed the results in fig.2 and fig.3 in which the y-axis denotes the database coverage, while the x-axis represents the query number. Fig.2 shows the experimental results on IMDB where AL and Probing curves represent the results of crawling method based on machine learning and probing respectively. The result shows that AL method is more efficient than Probing, holding a lead of up to 20% coverage. The experimental results on Siyuan Music Station are plotted in fig.3, which also indicates the AL method is more efficient than Probing method. For example, the AL method only issues 80 queries to retrieve almost 50% of the documents, while the Probing method requires more than 200 queries for the same coverage.

![Figure 2. Experiment on IMDB](image1)

![Figure 3. Experiment on Siyuan Music Station](image2)

#### B. Learning Policy Comparison

To compare the performance of learning models using different policies, we conducted another experiment. Fig. 4
illustrates the results of four types of learning policies, i.e. kNN, Naïve Bayes, SVM and MLP (Multilayer Perceptron).

It turns out that the performance of kNN, Naïve Bayes and SVM is more or less the same, but the MLP performs the worst, especially in initial period. This seems to be because, with the small samples in initial period, MLP failed in convergence. But as the number of samples grows, the algorithm converges and achieves a better performance. By comparing learning policies on different sites, we found that the performance of Bayes was unstable. And both kNN and SVM excelled on small training samples. Nevertheless SVM requires much more execution time than kNN. As a result, we prefer to employ kNN in the experiments.

As shown in fig.4, although the four curves are slightly different, however 10% higher than the Probing curve. We analyzed their query logs and found the percent of common keywords selected by different policies is about 82%. Consequently, the efficient performance of AL method is relatively independent of the machine learning policy it adopts. We believe the outstanding performance benefits from the overall method.

![Diagram](image)

Figure 4. Comparison between different learning models on Siyuan VOD

V. CONCLUSION AND FUTURE WORK

The paper presents a novel machine learning Deep Web crawling approach. The method enables a crawler to utilize diversiform features of keywords and exploit the information to derive promising queries. Experimental evaluation on real Deep Web sites shows that the method has a great potential over previous methods. It works well for both the structured database and the database for plain-text. In general, it retrieves their 80% of the total records by issuing hundreds of queries.

However, so far we have only discussed the method for genetic box. It is interesting and promising to apply the method to a whole query form. We will focus on the issue in our future work.

ACKNOWLEDGMENT

The research was supported by the National High-Tech R&D Program of China under Grant No.2008AA01Z131, the National Science Foundation of China under Grant Nos.60825202, 60803079, 60633020, the National Key Technologies R&D Program of China under Grant Nos. 2006BAK11B02, 2006BAJ07B06, the Program for New Century Excellent Talents in University of China under Grant No.NECT-08-0433.

REFERENCES