Research Journal of Applied Sciences, Engineering and Technology 5(4): 1278-1283, 2013
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: June 28, 2012 Accepted: August 08, 2012 Published: February 01, 2013

A Feature-Weighted Instance-Based Learner for Deep Web Search Interface Identification

[1]Hong Wang, [1]Qingsong Xu, [2]Youyang Chen and [2]Jinsong Lan
[1]Department of Mathematics,
[2]Department of Information Science and Engineering, Central South University, China

Abstract: Determining whether a site has a search interface is a crucial priority for further research of deep web databases. This study first reviews the current approaches employed in search interface identification for deep web databases. Then, a novel identification scheme using hybrid features and a feature-weighted instance-based learner is put forward. Experiment results show that the proposed scheme is satisfactory in terms of classification accuracy and our feature-weighted instance-based learner gives better results than classical algorithms such as C4.5, random forest and KNN.

Keywords: Deep web mining, instance-based learning, search interface identification

INTRODUCTION

The deep web (also called the invisible web, hidden web) (Bergman, 2001) refers to the web content that is behind HTML forms and web services search interfaces which are hidden from traditional web crawlers. In most cases, the only way to surface the deep web information is by formulating various queries on such search interfaces. Applications such as deep web sampling and deep web categorization are based on the analysis of these interfaces. Since the majority of search interfaces on the deep web are HTML forms, to automatically integrate them and retrieve their contents, the perquisite is to identify whether these forms are searchable.

In this study, we address the above-mentioned identification problem using a pre-query approach. We first carefully examine form features and create a hybrid feature set to characterize the HTML forms. Instead of using tree classifiers in previous researches (Cope *et al*., 2003; Barbosa and Freire, 2005; Ye *et al*., 2009), we try a different instance-based one for search forms classification. Based on the K* algorithm proposed in Cleary and Trigg (1995), we put forward a Feature-Weighted Instance-Based Learning (FWIBL) algorithm. FWIBL achieves a satisfactory classification accuracy and gains better results than the original K* algorithm and classical classification algorithms such as KNN, C4.5 and random forest on real datasets.

This study has the following contributions:

- We propose to model an HTML form using hybrid features, namely features from structural form controls and semantic textual contents within the form involved. This combination of structural and semantic features has proved to be successful.
- We propose a novel instance-based learning algorithm for search for identification purpose. Our approach is based on K* but we use a feature-weighted blending parameter to compute the sphere of influence (the effective number of instances) in K*.
- We conducted tests of the improved algorithm with classical classification algorithms such as KNN, C4.5 and random forest on real data sets. Experimental results have shown that the current approach is more effective in differentiating search forms from non-search forms in the deep web

LITRATURE REVIEW

Finding interfaces or entry points (mainly HTML forms) to search online databases is crucial for further information extraction and integration of the deep web. If the scale of the deep web under investigation is small, we could manually judge and collect the search interfaces as was done in Palmieri Lage *et al*. (2004) and Chang *et al*. (2004) though it might be a little tedious. But this method will not work for the real deep web, which is estimated to be 500 times larger than the surface web (Bergman, 2001). Search interface identification process has to be automatic. In a machine learning environment, a binary classifier is needed to differentiate searchable forms from non-searchable ones.

Two major approaches are currently employed to construct the binary classifier: pre-query and post-query approaches. The former method uses a form classifier to judge an HTML file according to the features within

Corresponding Author: Hong Wang, Department of Mathematics, Central South University, China

the form itself while the latter one identifies the form search ability by means of submitting queries to the web databases through HTML forms and decision is made based on the returned result pages.

Cope *et al*. (2003) proposed a pre-query approach that employs automatically generated features to describe textual forms and uses the C4.5 decision tree learning algorithm to classify them. This method was further developed by Barbosa and Freire (2005, 2007) but they used far less features than Cope *et al*. (2003) while achieved a higher precision.

Bergholz and Childlovskii (2003) gave an example of the post-query approach. They constructed a form analyzer using different heuristics such as the length of the input field, the presence or absence of a password protection to identify whether a form is queryable. They then applied the query prober to manage the automatic form filling and decided the usefulness of the form according to results of probing.

Ye *et al*. (2009) extended the random forest algorithm by applying a weighted feature selection during the building of individual classifiers. They also compared their algorithm with Bayes, C4.5, SVM and traditional random forest approaches.

Shestakov (2009) suggested to divide all forms into 2 groups (u-forms and s-forms) based on the number of visible form controls and demonstrated that such separation improves the system accuracy. They implemented 2 binary classifiers: the first is for forms with one or 2 visible fields of select or text types (u-forms) and the second is for the forms with more than 2 visible fields (s-forms). Both classifiers determine whether a form is searchable or non-searchable using on a decision tree algorithm.

The above-mentioned previous approaches have demonstrated the power of tree classification algorithms. In this study, we want to transfer to instance-based learning and give it an opportunity to test its strength in deep web search interface identification.

Before discussing how to classify an HTML form, we have to extract the features within the form. Thus, we will discuss form feature extraction first.

Fig. 1: Arxiv search interface-a simple search form

Different from previous papers focusing on structural features only, we take a hybrid approach which exploits both structural and semantic information within the form.

METHODOLOGY

Feature extraction: The search interface identification problem can be stated formally in the following way: given a set of all web pages W_A, find $W_S \subset W_A$ which satisfies that W_S contains searchable interfaces. As stated in Cope *et al*. (2003), HTML forms constitute of a large majority of search interfaces on the deep b, thus we only discuss HTML form-based W_S in the following.

To differentiate W_S from W_A-W_S (non-search forms) automatically using a machine learning approach, we should first train the classifier with sample datasets which gives a good characterization of the real word HTML forms.

According to Raggett *et al*. (1997), an HTML is a section of an HTML document which begins with a <*form*> tag and ends with a </*form*> tag. A search HTML form containing normal content, markup, controls (such as checkboxes, radio buttons and menus) and labels on those controls. Users generally issue a search query by modifying the controls (entering text, click checkboxes and/or selecting menu items) in the form and then submit the query to an agent for further processing. There are numerous search forms on the web and a typical search form on www.arxiv.org and its HTML source code are shown in Fig. 1 and 2, respectively.

As shown in Fig. 2, the arxiv search form contains a pair of FORM tags, 2 INPUT elements (one type is "text" and the other is "submit") and one SELECT

```
<form id="search" method="post" action="/search">
<div class="search-for">Search or Article-id</div>
<div class="links">(<a href="/help">Help</a> | <a href="/find">Advanced search</a>)</div>
<input type="text" name="query" size="24" maxlength="64" />
 
<select name="searchtype">
<option value="all" selected="selected">All papers</option>
<option value="ti">Titles</option>
<option value="au">Authors</option>
<option value="abs">Abstracts</option>
<option value="ft">Full text</option>
<option value="help">Help pages</option>
</select>
<input type="submit" value="Go!" /><br />
</form>
```

Fig. 2: Arxiv search interface-HTML source code

Fig. 3: Arxiv login interface-a non-search form

element. FORM attributes such as “method” and “action” is considered in that they are very important features (Ye *et al*., 2009).

As is shown in Cope *et al*. (2003), features such as numbers of “text” INPUT elements and SELECT elements are also good indicators of the form search ability. These features are extracted from the form element or control structures and will be called “structural features” in our approach. One may notice that the word ”SEARCH” occurs five times within the form body and such semantically ”positive” words help a human judge in determining the form search ability. Generally, LABEL name, FORM action URL and textual contents within the form are of much semantical importance and they become “semantical features” in our extracted feature set. The arxiv search form also contains a DIV tag, which is more frequently used outside a form for style purpose and will be ignored.

A non-search form differ from a search from either structurally or semantically or both. Take an example again from the Arxiv.org. Its login interface is a non-search form (Fig. 3). From its HTML source code (Fig. 4), we can see that the “password” INPUT element makes Fig. 4 different from Fig. 2 structurally and the ”negative” words such as “login”, “password” makes these 2 forms different semantically.

Due to the above considerations, we will model an HTML form using structural features such as number of the labels, absence or presence of a password INPUT element within a form. Moreover, we put a special emphasis on semantic features which are extracted from label or control names, FORM action URL and all the other textual contents within the form.

An improved instance-based learner: Instance based (memory-based or case-based) learning algorithms classify a new test instance by comparing it with instances previously seen in the training process, which have been stored in memory or databases (Aha *et al*., 1991). Its underlying assumption is that similar instances should be in similar classes. And the core of such learning algorithms is how to define the similarity between 2 instances and how to make a prediction based on such similarities.

K*(KStar) algorithm proposed in Cleary and Trigg (1995) is a typical example of instance-based learning. For the *i*-th feature, its transformation probability from instance *a* to instance *b* can be expressed as *p (i) (b|a)*. And the transformation probability from instance *a* into instance *b p (b|a)* is calculated using the following formula:

$$p(b \mid a) = \prod_{i=1}^{m} p(i)(b \mid a), \tag{1}$$

where, *m* is the number of features in the instances.

In computing *p (i) (b|a)*, the K* algorithm assumed that all features are of equal importance and used an identical blend parameter to control the "sphere of

```
<form name="user-login" action="http://arxiv.org/user/login" method="post">
<h2>If you're already registered:</h2>
<input type="hidden" name="dest" value="http://arxiv.org/user" />
Enter your <b>username or e-mail address</b>,<br />
<input name="username" type="text" size="28" maxlength="64" /><br />
and <b>password.</b>
<br />
<input type="password" name="password" size="28" maxlength="128" />
(Forgot your password?
<a href="/user/lost_password">Try this</a>.)
<br />
 <br />
<input type="checkbox" name="remember" value="1" />
Have my browser remember who I am (<a href="/user/about_perm_cookie">explain</a>)
<br /> <br />
<input type="submit" name="submit-login" value="Login" />
</form>
```

Fig. 4: Arxiv login interface-HTML source code

influence" for all features. However, in real classifications, different features actually play different roles and thus cannot be treated equally.

In this study, we consider the differences of features' importance in computing transformation probabilities between instances and thus improve the original K* algorithm by introducing a feature-weighted blend parameter.

In our approach, the importance of features is measured by the information gain statistic. The information gain of a given attribute x_i with respect to the class attribute C is the reduction in uncertainty about the value of C when we know the value of x_i and is computed using the following formula:

$$I_i = I(C; x_i) = H(C) - H(C \mid x_i), \tag{2}$$

where, $H(C)$ denotes the entropy of C and $H(C \mid x_i)$ denotes conditional entropy of the value of Y if the value of x_i is known.

The feature-weighted blend parameter $B(i)$ for the i-th feature is obtained by normalizing its information gain value I_i according to following formula:

$$B(i) = \begin{cases} MAX, & I_i = I_{min}; \\ MAX \times (I_{max} - I_i)/(I_{max} - I_{min}), & I_{min} < I_i < I_{max}; \\ 0, & I_i = I_{max}. \end{cases} \tag{3}$$

where, I_{max}, I_{min} are the maximum and minimum values among information gain values of all features respectively, and the upper limit MAX(0≤MAX≤100) of the normalization is the original blend parameter in the K* algorithm. With this normalization, small values are assigned to important features and large values to unimportant ones.

Our purposed feature-weighted instance-based learning algorithm is as follows:

Algorithm 1: A Feature-Weighted Instance-Based (FWIB) Learning Algorithm:

1. Let T be the training set, instance $v(x, c) \in T$ and the number of instances is N
2. for each test instance $u = (x', c')$ do
3. for each feature i in the feature set
4. calculate $p(i)(u|v)$ from v to u using $B[i]$ obtained in (3)
5. end for
6. Compute the transformation probability $p(u|v)$ according to (1)
7. Calculate class probability $c' = \arg\max_t \sum_{k=1}^{N} p(b \mid a) I(t = c_i)$, where I is an indicator function
8. end for

EXPERIMENTAL RESULTS

To measure the effectiveness of our feature-weighted instance-based learning approach, we performed experiments on 2 datasets. The datasets used in this study were taken from real world web pages that contain HTML forms. One was created by collecting sites contained in the largest web directory www.DOMZ.org. The other was extracted from sites collected by Yahoo Chinese web Directory (site.yahoo.com.cn). The HTML forms in both datasets were crawled during April 2011 and June 2011 using our web crawlers written in Python. The former data set mainly contains 897 English HTML forms while the latter mainly contains 526 Chinese ones. The data distribution for each set varies, ranging from general sites, academic sites, recreation sites to social sites and science sites. It is by far the largest bilingual data set used in similar research. Information of the 2 datasets is detailed in Table 1.

We evaluated our algorithm with 3 different feature evaluators, namely, information gain evaluator (default), x^2 static evaluator and gain ratio evaluator to calculate feature importance values. In addition, we tested a random method: generate $B(i)$ for all features from $[0, MAX]$ randomly. One may notice that if let $B(i) = MAX$ for all features, our algorithm is reduced to the original K* algorithm and we also tested our algorithm in this case. Results for DMOZ Dataset and Yahoo Dataset are shown in Fig. 5 and 6, respectively.

From Fig. 5 and 6, we can see that generally, FWIBL with information gain evaluator gains the highest classification precision and performs

Table 1: Two datasets used in the experiment

	Search forms	Non-search forms	Number of features
DMOZ	451	446	18
Yahoo	246	280	18

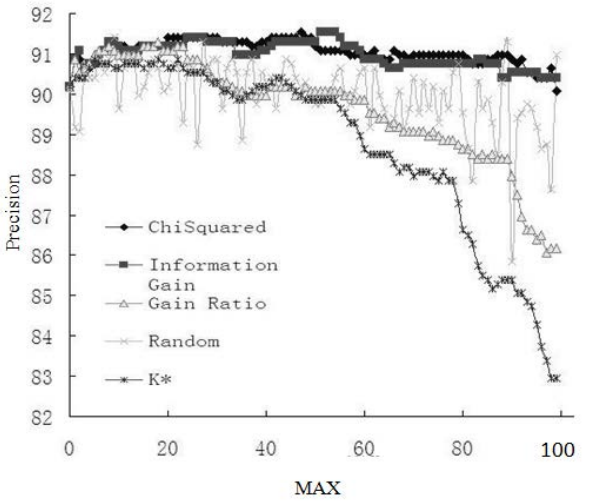

Fig. 5: DMOZ dataset-FWIBL with 4 feature evaluators

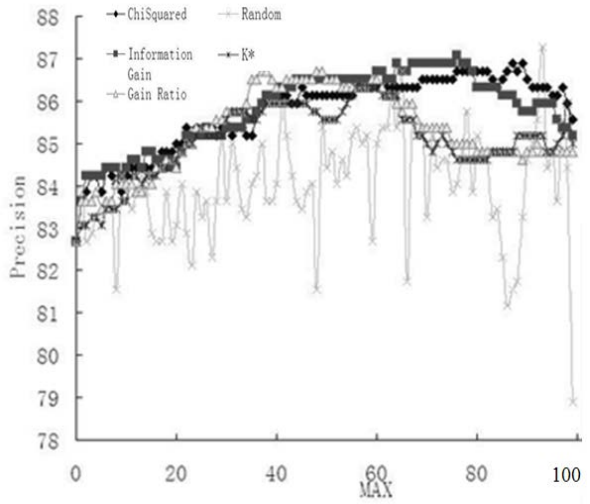

Fig. 6: Yahoo data set-FWIBL with 4 feature evaluators

Table 2: Classification accuracy for DMOZ and yahoo datasets

	DMOZ dataset	Yahoo dataset
C4.5	90.4124	85.5513
Random forest	89.2976	83.0798
IBK	88.9632	82.8897
K*(b = 20)	90.6354	84.6007
K*(b = best)	90.8584	86.3117
FWIBL (MAX = 60)	91.1928	86.6920
FWIBL (MAX = best)	91.5273	87.0722

well across a wide range of MAX values. FWIBL with the other 2 evaluators also perform better than the random chosen method and the K* algorithm. This can be attributed to the fact that different HTML form features carry different weights in the classifying a search form from a non-search form.

Besides, to compare FWIBL with other popular machining learning algorithms, we also experimented with C4.5, Random Forest and IBK(KNN) models using their default settings with 10-fold cross-validation in the machining learning tool kit Weka (Hall *et al*., 2009). The fractions of correct classification of various algorithms are presented in Table 2. The best result(s) for both datasets are highlighted in bold face. Results were also obtained for K* and FWIBL which uses a default blending value and which gave best accuracy.

As can be seen from Table 2, compared with C4.5, Random Forest, KNN,K*, FWIBL gave the best results for both of the two datasets.

In our experiments, we also find all algorithms perform better using DMOZ Dataset than Yahoo Dataset, though both datasets use the same feature set. The reason why this happens is still unclear. Perhaps it lies in the stylistic difference in HTML compiling habits between English and Chinese users or the linguistic difference in the 2 extracted semantic-related features between the 2 languages.

CONCLUSION AND FUTURE WORK

In this study, we propose to model the HTML form using a hybrid of structural and semantic features. Our results have shown that this combination of structural and semantical features is successful. In order to classify search forms from non-search ones in the deep web, we put forward the FWIBL learning algorithm. We conducted tests of the FWIBL algorithm on DMOZ and Yahoo Datasets and also compared its performance with popular classification algorithms such as KNN, C4.5 and random forest on the same datasets. Experimental results have shown that the current approach is more effective in finding the HTML search forms in the deep web.

Our experiments have also revealed that HTML search forms may have language-specific features, which may lies in users' habits in compiling HTML files or the semantic differences between the languages involved. Many other features in HTML files, such as URL, textual contents outside the HTML form, the header information of the web page may also help in the classification process. However, how to extract and exploit such features needs further investigation. In the future, we will try to do more experiments to figure out the differences in search forms in English and Chinese deep webs and focus on how to exploit features outside the HTML form but still with the same HTML file.

ACKNOWLEDGMENT

This study was supported in part by the National Natural Science Foundation of China (No. 90820302 & No. 10771217).

REFERENCES

Aha, D., D. Kibler and M. Albert, 1991. Instance-based learning algorithms. Mach. Learn., 6(1): 37-66.

Barbosa, L. and J. Freire, 2005. Searching for hidden-web databases. Proceedings of 8th International Workshop on the Web and Databases (WebDB 2005), Baltimore, Maryland, 5: 1-6.

Barbosa, L. and J. Freire, 2007. Combining classifiers to identify online databases. Proceedings of the 16th International Conference on World Wide Web, pp: 431-440.

Bergholz, A. and B. Childlovskii, 2003. Crawling for domain-specific hidden web resources. WISE Proceedings of the Fourth International Conference on Web Information Systems Engineering, pp: 125-133.

Bergman, M., 2001. The deep web: Surfacing hidden value. J. Elec. Publish., 7(1): 01-07.

Chang, K., B. He, C. Li, M. Patel and Z. Zhang, 2004. Structured databases on the web: Observations and implications. ACM SIGMOD Record, 33(3): 61-70.

Cleary, J. and L. Trigg, 1995. K*: An instance-based learner using an entropic distance measure. Proceedings of the 12th International Conference on Machine Learning, pp: 108.

Cope, J. and N. Craswell and D. Hawking, 2003. Automated discovery of search interfaces on the web. Proceedings of the 14th Australasian Database Conference, 17: 181-189.

Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. Witten, 2009. The weka data mining software: An update. ACM SIGKDD Exp. Newsletter, 11(1): 10-18.

Palmieri Lage, J., A. da Silva, P. Golgher and A. Laender, 2004. Automatic generation of agents for collecting hidden web pages for data extraction. Data Knowl. Eng., 49(2): 177-196.

Raggett, D., A. Le Hors and I. Jacobs, 1997. Html 4.0 specification. World Wide Web Consortium Technical Report, REC-html40.

Shestakov, D., 2009. On building a search interface discovery system. Proceedings of the 2nd International Conference on Resource Discovery, pp: 81-93.

Ye, Y., H. Li, X. Deng and J. Huang, 2009. Feature weighting random forest for detection of hidden web search interfaces. Comput. Linguist. Chinese Lang. Proc., 13(4): 387-404.