Automatic Recognition of the Hits Line in Search Engine Result Page

Haibo Qiu[1,a], ZhongMin Qian[2,b],MoShu Qian[3,c]

[1]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics,Nanjing, 211100, China

[2] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 211100, China

[3] Research Institute of UAV, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China

[a]email:919264104@qq.com, [b]email: qzm_mail@nuaa.edu.cn

Keywords: Automatic Recognition; Hits line; Search Engine; Information Extraction; Decision tree

Abstract. When a search engine returns query results to users, it always returns the number of relevant documents (i.e., hits). The text line containing the hits number is called as hits line. The hits number can be used in several applications such as building meta-search engine, estimating the size and the relevance of search engines. Since the hits line is mixed with other text lines in the result page, it is difficult to automatically recognize and extract the line from the result page. To this end, decision tree techniques are employed together with a heuristic approach to build two filters to automatically identify the hits line. First, texts in result pages are automatically extracted in lines. Then four key features are identified and used to build a decision tree based on the learning sample search engines. Classification rules from the tree are built to serve as the first filter to recognize the extracted text lines. To reduce the mis-classification of the first filter, the second filter is constructed using a heuristic weighting approach. The experiment based on 100 search engines shows that the accuracy of 10-fold cross-validation is up to 95%.

Introduction

With the explosive development of Web, search engines have become the most popular Web tool for us in our daily life. We can use search engines to search documents, products, movies, music, images, blogs, etc., which are accessible through the internet. When a query is submitted to a search engine, it always returns a number that indicates how many results are relevant to the user query. We call the number as hits number. This hits number can be used in some applications. For example, building a large-scale meta-search engine [1] needs the number of documents to globally rank the documents returned by multiple underlying component search engines. The hits number can also be used to estimate the size of a search engine (i.e. the number of documents a search engine contains) and the relevance of search engine (i.e. how good a search engine is relevant to a given topic or category) [2]. The hits number is usually contained in a line visually at a specific place within the query result page, which we call as document hits line.

To able to extract the hits number, we need to first identify the hits line. However, the hits line is mixed with other result text lines in the result page, therefore it is difficult to automatically recognize and extract the line from the result page. Although a lot of work [7,9,10,11,13] have been done for extracting data from Web pages, there is rarely work to address this particular problem. To this end, we develop a two-step filtering approach to this problem. First, we identify several features from a number of real result pages, and use them to build a decision tree. We then use the tree to derive several classification rules as the first filter. To refine the results of the first filter, we design a second filter which uses weight-based heuristic approach. Our experiment indicates that this two-step filtering approach is highly effective in identifying the real hits line.

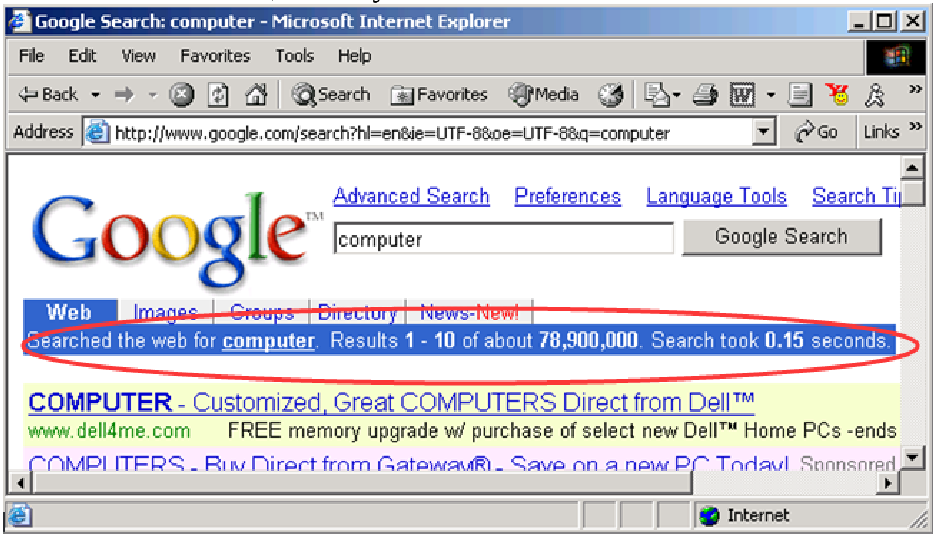
This paper has two major contributions. First, we propose a decision tree based learning approach to derive classification rules, which can be used to automatically identify the hits line. To resolve the failures of the decision three, we propose to use a second step to improve the effectiveness. Second, this approach can be also used to extract other special piece of data on the Web pages or even other type of text documents.

The rest of this paper is organized as follows. Section 2 describes our strategies for extracting HTML text lines. Section 3 presents the features used for building decision tree filter. Section 4 discusses the decision tree induction algorithm based on selected features. Section 5 discusses our method of constructing another filter using heuristics. Section 6 presents the experiment and Section 7 concludes the paper.

HTML Extraction

Visually, we can see that a whole HTML result page is displayed line by line through a browser. These kinds of lines are actually formatted in a HTML in a very complicated way. For example, a visual line in a browser might be composed of several <table> tags in the HTML. It is much harder to figure out organization of these <table> tags and extract them as a single line. One of the tasks of the HTML extraction is to organize and extract lines as showed visually in a browser.

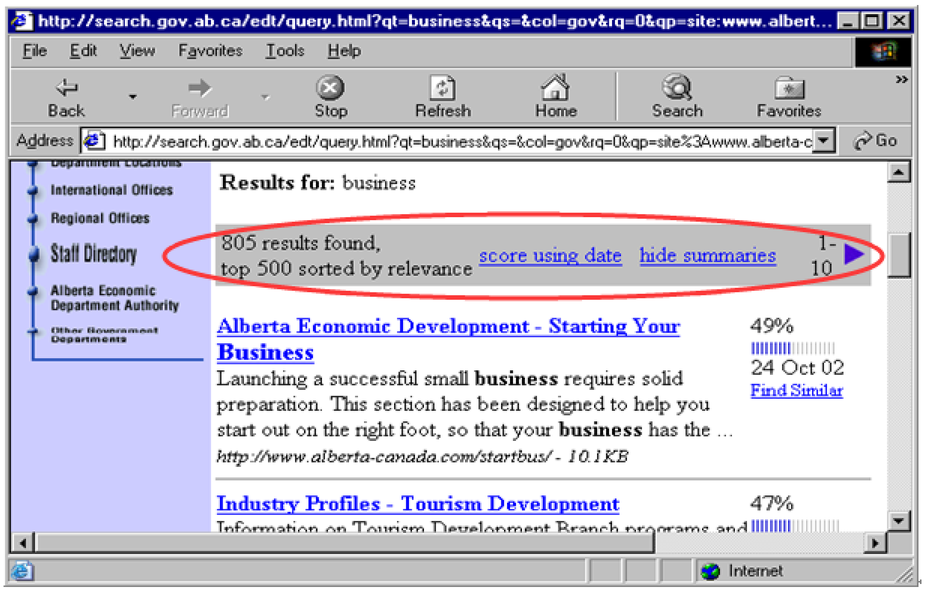
The documents hits line usually contains numeric numbers (integer), for example, in figure 1 and 2. If a line does not contain any numeric numbers, we can assume the line is not the hits line. Therefore, in the HTML extraction, we only consider the lines that contain numeric numbers.


Google Search: computer - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.google.com/search?hl=en&ie=UTF-8&oe=UTF-8&q=computer
Go Links
Advanced Search Preferences Language Tools Search Ti
computer
Google Search
Web Images Groups Directory News-New!
Internet

Searched the web for computer. Results 1 - 10 of about 78,900,000. Search took 0.15 seconds.

COMPUTER - Customized, Great COMPUTERS Direct from Dell™
www.dell4me.com FREE memory upgrade w/ purchase of select new Dell™ Home PCs -ends

Figure 1: the Example of document hits of Google


http://search.gov.ab.ca/edt/query.html?qt=business&qs=&col=gov&rq=0&qp=site:www.albert...
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address http://search.gov.ab.ca/edt/query.html?qt=business&qs=&col=gov&rq=0&qp=site%3Awww.alberta-c
Go
International Offices
Regional Offices
Staff Directory
Alberta Economic Department Authority
Internet

Results for: business

805 results found, top 500 sorted by relevance score using date hide summaries 1-10

Alberta Economic Development - Starting Your Business 49% 24 Oct 02 Find Similar
Launching a successful small business requires solid preparation. This section has been designed to help you start out on the right foot, so that your business has the ...
http://www.alberta-canada.com/startbus/ - 10.1KB

Industry Profiles - Tourism Development 47%

Figure 2: The Example of document hits line

However, a result page might contain many lines that have numeric numbers and these lines are actually the noisy data for recognizing the hits line. To reduce such noisy data, we should extract as fewer numeric numbers as possible from lines. So we use favor-hits-line extraction strategies to extract the numeric numbers. The favor-hits-line extraction strategy is to use some delimiters that often appear in the hits line to tokenize the lines to get the numeric numbers. Thus, even a line that has numeric numbers would not be considered if hits line delimiters are not used. For example, some lines that contain numeric numbers in a date format like 12/12/2013 or a time like 12:12:60 would not be considered as potential hits lines because the “/” and “:” are not the delimiters of hits lines. The extraction strategy would reduce the noisy data of hits line to some extent.

A result page might contain form widgets and scripts. It is not possible for the hits line to appear in them and they might become noisy data of hits lines. Therefore, they should not be extracted.

Feature Selection

Before learning, we should identify some features that can discriminate hits lines from others lines. We investigate the result pages of more than 100 search engines and find that a hits line usually might contain some key words(such as found, returned, matching and results), query term and “of” pattern(for example, 1-10 of 100) besides numeric numbers, as showed in figure 1 and 2. Based on this investigation, we define the following features for each line:

- The number of numeric numbers(*numOfNums*).
- Contain “*of*” pattern(*containOfPattern*)?
- Contain keywords(*containKeyWords*)?
- Contain query term(*containQueryTerm*)?

The number of numeric numbers has discrete values (1,2,3,4,6,…15). If a line has more than 15 numeric numbers, 15 is specified. If a line has “of” pattern, then the value of containOfPattern feature is 1, otherwise is 0. If a line contain one or more keywords, then the value of containKeyWords is 1, otherwise is 0. If a line contains query term, then the value of containQueryTerm is 1, otherwise is 0.

For the class, we just have two categories: one is hits line, the other is non-hits line.

Decision Tree Induction

We collect 100 search engines, respectively submit three queries to each search engines to get the result pages, extract all lines that contain numeric numbers, and then analyze the lines to get their features. All these lines with features are dumped to training database. So far, each line in the training database can be considered as a training example.

The decision tree induction algorithm is based on the ID3 [3]. In the algorithm, all features are categorical, i.e, discrete-valued. We use Gini function to select the best splitting feature. However, our decision tree induction algorithm, as showed in table 1, has some special differences from the ID3 [3]. It is the special circumstances of the topic that decide the differences. As we can see, in a result page, only one or two lines are the hits lines, while others are not, which counts for a huge majority. Our goal of the tree induction is to learning as many hits line formats as possible to classify new hits lines. We do not want any hits line is missing in the tree induction because of the majority voting rule. Therefore, the majority voting rule is not applicable to this circumstance. And our experiment proves that if we use the majority rule the prediction accuracy is very low. Thus, we revise the ID3 to adapt it to our decision tree induction. As showed in figure 3, if attribute-list is empty, we first check if there exists a hits line class; if it is, then we label the leaf node with the hits line class, otherwise, non-hits line is labeled. Another revise to the ID3 is when the samples for a given value ai of test-attribute is empty we label the leaf node with the non-hits line class. With these two revise, the classification rules are as in table 2.

However, the first revise leads to more non-hits line examples misclassified as hits line class. The result is that we may have a number of hits lines in a result page after applying the classification rules. Actually we just need one or two lines to be hits lines. To solve this side-effect of tree induction, we apply some heuristic approach as the last filter to get the real hits line.

```
Generate_Decision_Tree(samples, attribute_list)
{   create a node N;
      if samples are all of the same class C, then
          return N as a leaf node labeled with the class C;
      if attribute_list is empty then
      {     if there exists a hits line in samples, then
                return N as a leaf node labeled with the hits line class;
          else
                return N as a leaf node labeled with no-hits line class;
      }
      select test-attribute, the attribute among attribute_list with the highest impurity;
      label node N with the test-attribute;
      new-attriubte-list = attribute_list – test-attribute;  //remove the test-attribute.
      for each known value $a_i$ of test-attribute
            grow a branch from node N for the condition test-attribute = $a_i$ ;
            let $s_i$ be the set of samples in samples for which test-attribute = $a_i$ ;
            if $s_i$ is empty then
                  attach a leaf labeled with the no hits line class;
           else
                  attach the node returned by Generate_Decision_Tree($s_i$, new-attribute-list);
}
```

Table 1: Decision Tree Induction Algorithm

```
containKeyWords=0 Λ containOfPattern=1Λ numOfNums=4->true
containKeyWords=1 Λ containOfPattern=1 Λ numOfNums=2->true
containKeyWords=1 Λ containOfPattern=1 Λ numOfNums=12->true
containKeyWords=0 Λ containOfPattern=0 Λ numOfNums=1 Λ containQueryTerms=1->true
containKeyWords=0 Λ containOfPattern=1 Λ numOfNums=3 Λ containQueryTerms=0->true
containKeyWords=1 Λ containOfPattern=0 Λ numOfNums=1->true
containKeyWords=1 Λ containOfPattern=0 Λ numOfNums=2->true
containKeyWords=1 Λ containOfPattern=0 Λ numOfNums=3->true
containKeyWords=1 Λ containOfPattern=0 Λ numOfNums=4 Λ containQueryTerms=0->true
containKeyWords=1 Λ containOfPattern=0 Λ numOfNums=9 Λ containQueryTerms=1->true
containKeyWords=1 Λ containOfPattern=1 Λ numOfNums=3->true
containKeyWords=1 Λ containOfPattern=1 Λ numOfNums=4->true
```

Table 2: Classification Rules

Heuristic Approach

The heuristic approach is to filter the misclassified non-hits lines of the classification rules. To do this, we do some investigations on the result pages and the results of using classification rules. We find that the real hits line usually has more different keywords than non-hits lines and is located in first place or top 7 lines. We also find that some search engines [4,5,6] return the categories hits as well as the document hits in a result page, and the categories hits are always displayed before the document hits. The categories hits are still kept in the result of classification rules, for example, in table 3. In this case, the first heuristics can not apply to them. After the study of such results, we find that if we give "of" pattern more weights and choose the biggest number of lines with "of" pattern, the real hits line would have more weights than others. With these observations, we develop a filter that is based on counting weight of each line.

Weight = numOfDifferentKeyWords + containOfPattern(n+ isMaxNumber*m)+ isFirstLine*

The numOfDifferentKeyWords denotes how many different key words the line has; If the line contain “of” pattern, then containOfPattern is 1, otherwise 0. If the line contains the biggest number among the lines with “of” pattern, then isMaxNumber is 1, otherwise 0. If the line is the first line in the result of classification rules, isFirstLine is 1, otherwise 0. The n and m in our experiment is 4 and 3 respectively.

open directory categories (1-5 of 12)
home: cooking (1172 matches)
recreation: outdoors: camping: cooking (48)
more...] open directory sites (1-20 of 3077) ← *the real hits line*

Table 3: The Example of Containing Categories Hits and Document Hits After Applying Classification Rules

Experiment

As we mentioned in Section 4, we collect 100 search engines that covers different fields, such as government, art, agriculture, entertainment, university and general search engine. We manually submit three different queries to each search engine to get the first result page of each query. We use hothouseobjects library together with our algorithms to extract lines that contain numeric numbers from the result pages. And then we dump all the lines with the features needed in tree induction and heuristic approach into training examples database.

To do 10-fold cross validation [8], we divide the 100 search engines into 10 groups, and each group has 10 search engines. Then every time we use 9 groups for learning and 1 group for testing.

At first, we use the standard decision tree induction program C5.0 [12] to do 10-fold cross validation. The average accuracy is about 50% because of the majority rule in the tree induction. It causes that some real hits lines are misclassified as non-hits lines. This is totally avoided in our tree induction algorithm.

Then, we use our two filters to recognize the hits lines of the same data. First, we use 9 groups for learning to build the decision tree, and then use the classification rules of the tree to filter the lines of 1 group. After the classification rules filter is finished, we use heuristic filter to the intermediate result. The final results would be the predicted hits lines of the group. The experimental result is showed in table 4.

Experiment	Accuracy	Reasons
1	100%	
2	100%	
3	90%	no rules for the new site hits line
4	100%	
5	100%	
6	90%	no rules for the new site hits line
7	100%	
8	90%	One query result is misclassified
9	90%	no rules for the new site hits line
10	90%	One query result is misclassified
Average	95%	

Table 4: The Experimental Result of 10-fold Cross Validation with 2 filters

Each site has three result pages corresponding to three queries. In the evaluation of accuracy, if the hits line for one query is not correctly recognized, then we assume the hits line of the site is not correctly recognized. For example, in Table 1, the experiment 8 and 10 show that only one query result is not correctly identified, so the hits line of the site is considered not to be correctly recognized. Therefore, the accuracy is 90% out of 10 sites. There exist three cases in which a new special hits line format is met in testing, but it is not met in learning. Therefore, no rules can be applied to these new special cases. However, because of our tree induction algorithm, some other non-hits lines come into the result. The main reason that the experiments 3, 6 and 9 have 90% accuracy is that new special hits line format is met. If we have enough examples, the average accuracy would be 98%.

Conclusion

To recognize the document hits line in the search engine query result page, we use two filters to filter the lines containing numeric numbers: Classification rules filter and Heuristic filter. Through the two filters, the system can automatically recognize the document hits line with a very high accuracy of 95%.

The approaches can be adapted to other purposes of mining Web pages, for example, extracting the next page URLs in the result page or some specific information such as address, telephone, and so on.

Acknowledgement

In this paper, the research was sponsored by the Nature Science Foundation of Jiangsu Province (Project No. SBK2014042586)

References

[1] Wu Xiaolan, Wang Qi. Overviews on Meta-search Engine Researches [J]. LIBRARY AND INFORMATION SERVICE, 2009 53(9) 46-49

[2] Panagiotis G. Ipeirotis, Luis Gravano, and Mehran Sahami. Probe, count, and classify: Categorizing hidden web databases. SIGMOD Conference, 2001.

[3] Jiawei Han, Micheline Kamber, Data Mining Concepts and Techniques(Third Edition), Morgan Kaufmann Publishers, 2012.

[4] http://dmoz.org/

[5] http://www.yahoo.com

[6] http://www.google.com

[7] Can Lin, Zhang Qian, Xiaofeng Meng, and Wenyin Liu. Postal address detection from web documents. WIRI Conference, pages 40–45, 2005.

[8] Sholom M. Weiss, Casimir A. Kulikowski, Computer Systems that Lean Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems, Morgan Kaufmann Publishers,1991

[9] Hongkun Zhao, Weiyi Meng, Zonghuan Wu, Vijay Raghavan, and Clement T. Yu. Fully automatic wrapper generation for search engines. WWW Conference, pages 66–75, 2011.

[10] ZHAO Xin-xin [1] SUO Hong-guang [2] LIU Yu-shu. Web Content Information Extraction Method Based on Tag Window[J]. APPLICATION RESEARCH OF COMPUTERS 2007, 24(3) 144-146

[11] CHANG Hong-yao ZHU Zheng-yu CHEN Ye ZHANG Peng ZENG Li-fang. Content extraction technique for web pages based on HTML-tags[J]. COMPUTER ENGINEERING AND DESIGN. 2010, 31(24) 5187-5191

[12] C5.0, http://www.rulequest.com/

[13] Jer Lang Hong, Eugene Siew, Simon Egerton. DTM- Extracting Data Records from Search Engine Results Page using TreeMatching Algorithm. SOCPAR Conference, Dec. 2009, pages 149-154.