Automatic Information Extraction for Multiple Singular Web Pages

Chia-Hui Chang, Shih-Chien Kuo, Kuo-Yu Hwang, Tsung-Hsin Ho and Chih-Lung Lin

Dept. of Computer Science and Information Engineering
National Central University, Chung-Li 320, Taiwan
chia@csie.ncu.edu.tw, {bruce, want, windson, nono}@db.csie.ncu.edu.tw

Abstract. The World Wide Web is now undeniably the richest and most dense source of information, yet its structure makes it difficult to make use of that information in a systematic way. This paper extends a pattern discovery approach called IEPAD to the rapid generation of information extractors that can extract structured data from semi-structured Web documents. IEPAD is proposed to automate wrapper generation from a multiple-record Web page without user-labeled examples. In this paper, we consider another case when multiple Web pages are available but each input Web page contains only one record (called *singular* Web pages). To solve this case, a hierarchical multiple string alignment is proposed to allow wrapper induction for multiple singular Web pages.

1 Introduction

The problem of information extraction is to transform the contents of input documents into structured data. Unlike information retrieval, which concerns how to identify relevant documents from a collection, information extraction produces structured data ready for post-processing, which is crucial to many applications of text mining. Therefore, information extraction from Web pages is a crucial step enabling content mining and many other intelligent applications of the Web. Examples include meta-search engines [6], which organize search results from multiple search engines for users, and shopping agents [2], which compare prices of products from multiple Web merchants.

Information extraction has been studied for years, but mostly concentrated on free-text documents. In this case, linguistic knowledge such as lexicons and parsers can be useful. However, a huge volume of information on the Web is rendered in a *semi-structured* manner, where linguistic knowledge only provides limited hints. Another difference is that different Web sites have their own unique layout and format. Virtually no general "grammar rule" can describe all possible layouts and formats so that we can not have one extractor for all Web pages. As a result, each Web site may require a specific extractor, which makes it impractical to program extractors by hand.

In recent years, several research efforts have focused on learning the extraction rules to automate wrapper generation. For example, WIEN, Softmealy,

STALKER, and IEPAD, etc. [7, 4, 5, 1]. The former three are machine learning based approaches and rely on user-labeled training examples indicating what information to be extracted. The last one, on the other hand, uses a pattern mining approach to discover display template for data records. Therefore, it saves manual effort for labeling examples. Such a property is especially useful when dealing with Web pages containing too many variations such that a lot of user-labeled examples are required.

However, the input to IEPAD is limited to one Web page containing multiple records (called *plural pages*). When a Web page contains only one record (called *singular pages*), IEPAD fails to work since IEPAD relies on the discovery of regular and contiguous patterns to "estimate" the display template of these records. This multiple singular pages input is a contrast to the one plural page input to IEPAD, where each containing several records. In fact, IEPAD's approach is to solve the extraction of the record boundary not the extraction of individual features. In this paper, we consider the case when multiple Web pages are available but each input Web page contains only one record.

To extract meaningful information from multiple pages with similar structure, one thought is to align these multiple pages and sifting representing features for users. However, alignment of long pages is time consuming and error-prone. Therefore, we propose a hierarchical approach to solve the problem layer by layer. Such an approach is similar to the idea of building the parsing trees of these pages and then compare the them level by level. The main problem is how to correctly align multiple pages. If the alignment is done, what information should be extracted? In the next section, we describe a hierarchical multiple string alignment to show how data are divided for alignment. Section 3 describes the criteria for feature selection from the divided blocks and the logic of the extractor. Section 4 shows the preliminary experiment results to demonstrate the effect. Finally, section 5 concludes the paper.

2 Alignment of Multiple Singular Pages

In this section, we describe a hierarchical approach to align the input pages. The key idea is to divide each input page into a sequence of blocks and align these block sequences to understand the page structure for a Web site. The division is based on an abstraction/encoding function of the input pages. In this paper, we use a layered encoding function which utilize the HTML tags to translate each page into tag sequence and divide the page content at the same time. The alignment is based on both the markups in the HTML pages and the structure similarities between two blocks.

2.1 Layered Encoding

Layered encoding is a tag-based encoding scheme but considers only tags at the same layer. The so-called *layer* refers to the depth from the root in the HTML parse tree. For each tag, there is an associated layer. The layered encoding,

Layer_Encoding, transforms an information block to a token sequence with all tags of the smallest layer (or the outermost layer tags) and denotes any other text between two such tags as the <TEXT> token. For example,

Layer_Encoding("SOME TEXT<P><A>LINK</A></P>")
="<TEXT><P><TEXT></P>".

The encoding scheme used here is different from block-level encoding or text-level encoding used for IEPAD. Layered encoding does not need prior knowledge about the characteristics of individual tags, but depends on the layer positions of the tags in the parsing tree. Remember that the encoding scheme is not only used for data abstraction (such that the alignment program can recognize the structures), it is also a segmentation of the input text. For a token string with k tokens, there is a natural section of the page into k pieces. For m multiple singular pages, we will have m token strings for multiple string alignment.

2.2 Multiple String Alignment

Multiple string alignment is a technique used to find a general presentation of the critical common features for multiple strings. It has been successfully applied in IEPAD for constructing patterns which can tolerate missing attributes [1]. In this paper, we use multiple string alignment to find the general expression for the m token sequences.

Multiple string alignment is a generalization of *alignment* for two strings which can be solved in $O(s * t)$ by *dynamic programming* to obtain optimal *edit distance*, where s and t are string lengths. As an example of *two string alignment*, consider the alignment of two strings"*dtbtbt*" and "*dtbt*" shown below:

$$\begin{array}{c} d\ t\ b\ t\ b\ \ t \\ d\ t\ b\ t\ -\ - \end{array}$$

In this alignment, the first four tokens match their counterparts in the opposite string and the last two tokens are opposite two hyphens. The edit distance between two strings are defined as the summation of the matching score between two aligned tokens. For example, the value for matching two same tokens is zero, while matching a token against a hyphen charges 1; a value of 3 is given for matching two different tokens to avoid such alignment. However, such matching values can not distinguish the above alignment from the following alignment, because both alignments have the same edit distance 2.

$$\begin{array}{c} d\ t\ b\ \ t\ b\ t \\ d\ t\ -\ -\ b\ t \end{array}$$

In order to find which alignment is correct, the match function needs to consider more than just facile tokens, especially on matching two <TEXT> tokens since <TEXT> tokens actually represent some text contents that have been abstracted. There can be many ways to do the computations. For example, we can

represent a text string as a vector $(t_1, t_2, ..., t_n)$ of n tags and compute the cosine of the angle between two text vectors for the distance.

$$dis(\mathbf{t}, \mathbf{s}) = 1 - \frac{\sum_{i=1}^{n} t_i \cdot s_i}{\sqrt{\sum_{i=1}^{n} s_i}\sqrt{\sum_{i=1}^{n} t_i}}$$

Once the correct alignment for two token strings is given, we can extend the idea to multiple string alignment. The score or goodness of a multiple string alignment is not as easily generalized. A common used scores is the *sum of pairs* score which is the sum of the scores of pairwise edit distances. To avoid $O(n^m)$ complexity by dynamic programming for m strings of length n, an bounded-error approximation algorithm, called *center star*, is used such that the score of the alignment is no greater than twice the score of optimal alignment. The center star algorithm first computes the center string S_c which minimizes $\sum_{S_i \in S} D(S_c, S_i)$ over all strings in S. Then, align each string to the center string to construct multiple alignment [3].

From the alignment of multiple token strings, we can summarize the *signature representation* for them as a regular expression. For example, suppose we have constructed the alignment for three token strings "`dtbtbt`", "`dtbt`" and "`dtbat`" as follows:

$$\begin{array}{l} d\ t\ b - t\ b\ \ t \\ d\ t\ b - t - - \\ d\ t\ b\ a\ t - - \end{array}$$

The signature representation will be expressed by "`dtb[a|-]t[b|-][t|-]`", which divides each text information into 7 blocks. Among them, some blocks are of particular interests. Notice that token strings are composed of tag tokens and `<TEXT>` token; and only `<TEXT>` and some special tag tokens (which contain hyperlinks such as <A> and <IMG> tags) might contain useful information. Therefore, whenever there is a `<TEXT>` token or special tag tokens in the pattern, the corresponding information should be retained. To summarize, the process for block division and extraction proceeds as follows:

1. First, apply layered encoding for each of the m contents.
2. Second, align m token strings by center star algorithm.
3. Finally, extract those blocks denoted by `<TEXT>` and hyperlink tag tokens.

2.3 The Hierarchical Approach

From the $m \times n_1$ matrix of the multiple string alignment, we are especially interested in columns that are denoted by `<A>` tags and `<IMG>` that contain link information and `TEXT` tags that needs to be further divided. For each of the k `TEXT` columns, we apply the above procedure iteratively and get a result of $L_2 = n_{2,1} + n_{2,2} + \ldots + n_{2,k}$ blocks, where $n_{2,i}$ denotes the number of divided blocks for ith `TEXT` column. Then, for each of the m `TEXT` columns in the L_2 blocks, we can apply the block extraction procedure and get a result of $L_3 = n_{3,1} + n_{3,2} + \ldots + n_{3,m}$ subblocks, where $n_{3,i}$ denotes the number of divided blocks for the ith `TEXT` block. The same process can be applied level by level until no column can be divided anymore.

3 Feature Extraction

Through the hierarchical multiple string alignment approach, the input data can be divided as detailed as any tags can separate. Often this can divide as many blocks from hundreds to thousands. Therefore, the system should also decide what columns to be presented for users' selection. For example, if the texts of a column contain only string "`Author`", such a column might be the name describing the following column. If a good judgement is available, the system can always present users with information that might be useful and greatly reduces the number of blocks to be presented.

To extract useful information, a simple judgement is to examine the varieties among the blocks in a column. If all m blocks contains the same text string, it will be unnecessary to extract sun a column. On the other hand, if large varieties are presented among these blocks, the column might also be useless. After all, whether information is useful or not is quite objective. In this paper, we use edit distance to measure the dis-similarities between two text blocks, where the contents are compared literally byte by byte. A threshold is given for the decision of feature selection as shown in experiments. With the sifting process, the system presents only useful information for users and make this approach practical.

4 Experimental Results

Since most integration systems have to deal with CGI generated pages, we therefore focus ourselves on such data sources. The experiments here contain five commercial Web sites including Barnes&Noble, YahooKimo, NewWorld, E-Family, and DVDmall, each containing 50 pages. The average document size is 6KB∼57KB bytes for the data set. Comparing to the average number of 306∼1694 tags in a page, it's about 3.7% the page size. The number of columns segmented from a page is about 40∼393 for a depth of 20 levels.

4.1 Generalizing over Unseen Pages

Although the process for the pattern discovery is not like typical machine learning process, the goal is the same to generalize the extraction over unseen pages. Therefore, we have designed the experiments in a way similar to that used in machine learning. For each Web site, we randomly select m pages as the training pages and use remaining $N - m$ pages as validation set. For each Web page, the m training Web pages are referenced to get the alignment. To measure the correctness of the data alignment, we compute the ratio of correctly aligned blocks to the number of expected attributes. If any page has accuracy less than 1, such a page is added to the training pages such that $m + 1$ training pages is used as baseline for the extractor. As shown in Figure 1, the accuracy rate is pretty high for the first four data sources. Barnes&Noble has accuracy 85% for two training pages and increase to 90% for 5 training pages. The average accuracy is about 95% with 2 training example, and increased to 97% for two training examples.

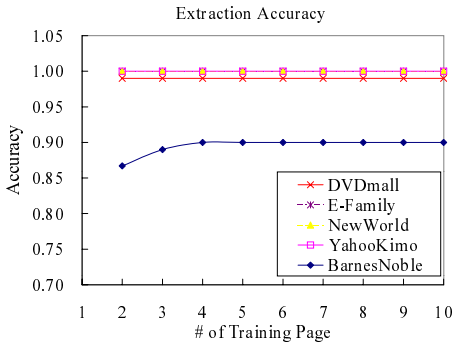

Fig. 1. The accuracy for different training example.

4.2 Feature Selection

As mentioned before, feature selection is designed to do the filtering of unnecessary information. The selection retains those columns with column variety greater than a threshold ϵ. With this mechanism, half the columns are filtered when a value of 0 is used for many Web site. The number of columns retained for Barnes&Noble is 67 as shown in Figure 2. As the threshold increased, more blocks are filtered. Suppose the number of desired attributes are 20 for this Web site, we can choose larger threshold ϵ to filter more columns. However, there is a risk that the desired blocks are filtered if the threshold is set too large.

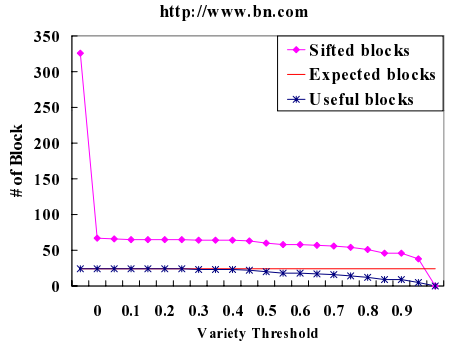

Fig. 2. The number of columns retained versus the threshold.

5 Conclusion and Future Work

In this paper, we presented a novel approach to the extraction of multiple one-record pages. The approach of hierarchical multiple string alignment to automatic feature extraction follows the track of IEPAD to save the efforts of labeling for machine-learning based approach. By taking the advantages of HTML tags, we can divide the Web content as detail as those tags can separate. Meanwhile, the variety measurement of strings in the same column is used to sift interesting features successfully. The result of experiments shows that the alignment can achieve 96% with two training pages. An average of 30 features are selected from 10KB Web pages, where most useful information are preserved.

Acknowledgement

This work is sponsored by National Science Council, Taiwan under grant NSC90-2213-E-008-042.

References

1. C.-H. Chang and S.-C. Lui. Iepad: Information extraction based on pattern discovery. In *Proceedings of the 10th International Conference on World Wide Web*, pages 223–231, Hong-Kong, 2001.
2. R.B. Doorenbos, O. Etzioni, and D.S. Weld. A scalable comparison-shopping agent for the world-wide web. In *Proceedings of the 1st International Conference on Autonomous Agents*, pages 39–48, NewYork, USA, 1997.
3. D. Gusfield. *Algorithms on Strings, Trees, and Sequences.* Cambridge, 1997.
4. C.-N. Hsu and M.-T. Dung. Generating finite-state transducers for semi-structured data extraction from the web. *Information Systems*, 23(8):521–538, 1998.
5. I. Muslea, S. Minton, and C. Knoblock. A hierarchical approach to wrapper induction. In *Proceedings of the 3rd International Conference on Autonomous Agents*, pages 190–197, Seattle, WA, 1999.
6. E. Selberg and O. Etzioni. Multi-engine search and comparison using the metacrawler. In *Proc. of the Fourth Intl. WWW Conference*, Boston, USA, 1995.
7. N. Kushmerick D. Weld and R. Doorenbos. Wrapper induction for information extraction. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 729–737, Japan, 1997.