Sequential Pattern Mining for Web Extraction Rule Generalization*

Chia-Hui Chang
Dept. of Computer Science and Information Engineering
National Central University, Chung-Li 320, Taiwan
chia@csie.ncu.edu.tw

ABSTRACT

Information extraction (IE) is an important problem for information integration with broad applications. It is an attractive application for machine learning. The core of this problem is to learn extraction rules from given input. 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 situation when multiple Web pages are available but each input Web page contains only one record (called singular page). To solve this problem, a hierarchical multiple string alignment approached is proposed to generate the extraction rules from multiple singular pages. In addition, the same method can be applied to IEPAD for finer feature extraction.

Keywords: information extraction, IEPAD, alignment, singular pages, encoding hierarchy

1. INTRODUCTION

Information integration systems often require a lot of efforts for manipulation among various data formats. Sometimes the data format are not even meant for application programs. For example, information collected from Web is usually expressed in HTML which is intended for display on Web browsers. Hence, there presents a special need for *wrappers* to extract relevant information from machine-generated Web pages. Contrast to *free-text* information extraction which roots from linguistic analysis [11], Web IE relies on structure identification marked by HTML tags (see [7] for a survey). The markups in Web pages together with the multiple tuples to be extracted contribute the so called semi-structured documents.

Previously, several research efforts have focused on wrapper generation for Web-based sources, for example, WIEN, Softmealy, and STALKER, etc. [12, 6, 8]. Basically, the wrapper induction systems generate a specialized extractor for each Web data source. Their work produce accurate extraction results, but the generation of the extractors still requires human-labeled/annotated Web pages as training examples to "tell" a wrapper induction program how to extract data from a given Web page by providing examples on how to partition a Web page, and how to group sub-strings into attributes and data records.

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

IEPAD (an acronym for information extraction based on pattern discovery) [2] is a novel IE system which attempts to eliminate the need of user-labeled examples. The user does not need to tell IEPAD what information to extract, but simply choose among patterns (discovered by IEPAD) to see if the pattern can extract the desired information. This amazing feature is based on the assumption that the input is a multiple-record Web page so that sequential pattern mining can be applied to discover the repeats. However, there are two major problems regarding this approach. First, IEPAD's assumption motivates the mining of all patterns that occur k-fold for relevant information with k-records ($k > 2$). If the input page contains many k-fold patterns, a lot of patterns will be discovered and make users hard to choose from. Or if the relevant information contains only one record, IEPAD fails. Second, IEPAD's approach solves only the extraction of the record boundary not the extraction of individual features. If one wants to extract finer information, post processing is required.

In this paper, we extend IEPAD to solve the above problems. The remainder of the paper is organized as follows. Section 2 presents background material on IE and IEPAD. We describe the system framework in Section 3. Section 4 presents the applications of our approach to multiple-record page extraction and one-record page (singular) extraction; and finally Section 5 concludes the paper.

2. BACKGROUND

IE from Semi-structured Data

Information Extraction (IE) is concerned with extracting the relevant data from a collection of documents. The goal is not necessarily to produce a general-purpose IE system, but to create tools that would allow users to build customized IE system quickly. A key component of any IE system is its set of extraction patterns (or extraction rules) that is used to extract information relevant to a particular extraction task. Therefore, research efforts have focused on the work to induce useful extraction patterns from training examples.

Information extraction from free text has been a subfield of Natural Language Processing (NLP) that is concerned with identifying predefined types of information from text [3]. Semi-structured data such as product description in predefined HTML templates is however an attractive application for machine learning [7]. IE can be useful in a variety of domains. Free text IE from MUC (Message Understanding Conference) series have focused on domains such as Latin American terrorism, joint ventures and company

management changes. Web information extraction, on the other hand, arises from the need for information integration on several applications such as comparison-shopping agents [4], job finding, etc.

There are three factors when designing an IE system. First, whether the training examples are annotated may influence the design of an IE system. Most machine learning based approaches rely on user-annotated training examples [9, 1, 12, 6, 8], very few systems generate extraction rules based on unlabeled text [10, 2]. Second, depending on the characteristics of the application domains, IE systems use extraction patterns based on one of he following approaches: context-based constraints, delimiter-based constraints, or a combination of both. For example, wrapper induction systems such as WIEN [12], Softmealy [6], Stalker [8] generate delimiter-based extraction rules, while some generate context-based rules [10, 2]. Finally, some IE systems may rely on background knowledge for pattern generalization. For example, RAPIER [1] imposes constraints based on the WordNet semantic classes. Softmealy [6] defines token classes such as word and nonword token classes.

The IEPAD System

IEPAD [2] is an IE system that does not require user-annotated training example. It applies several pattern discovery techniques including PAT-trees, multiple string alignment and pattern matching algorithms. The key idea of IEPAD is to discover and use patterns to extract data from target Web pages. A pattern in IEPAD is a subclass of regular expressions over an alphabet of tokens. Each pattern matches a set of strings. A pattern may contain options and alternatives. An example of a pattern is given below:

```
    <P><A><TEXT></A><BR>[<TEXT>]<BR><TEXT>
<BR><TEXT>                                    (1)
```

where <P>,
, <TEXT> etc. are tokens that match HTML tags `<p>`, `<br>`, and text strings, respectively. Options are denoted by [...]. In this example, the sixth token `<TEXT>` is optional. The following string matches this pattern:

```
    <p><a href="http://www.csie.ncu.edu.tw">
NCU</a><br>National Central University<br>
Chung-Li<br>Taiwan.                           (2)
```

The IEPAD extractor basically works as follows. Given a pattern and a Web page, the extractor translates the Web page into a token string and scans the token string to find all substrings that matches the pattern. It then outputs the substrings as data records. In this case, the extractor will output (3) given the example (2) and pattern (1). Removing the HTML tags, we obtain a data record with the four text strings in (2) as the attributes:

```
    ⟨"NCU","National Central University",
"Chung-Li","Taiwan"⟩.                         (3)
```

The IEPAD pattern discoverer reverses the task of the extractor. To discover extraction patterns, IEPAD relies on the use of an encoding scheme to abstract the input pages such that potential patterns can be discovered. For example, the above encoding scheme considers only tags that defines the structure of a document and encodes all other information between any two tags as a special token `<TEXT>`. IEPAD then discovers all patterns that occur k times in the token string by pattern mining techniques such as PAT tree and multiple string alignment. In other words, IEPAD presents rules which reflect the contextual structure of the data. This is why IEPAD also features the use of unlabeled data and in instead provides users with discovered patterns for selection. In this paper, we further devise a compromise approach for the user to specify the relevant information.

3. SYSTEM FRAMEWORK

In this section, we present the framework of our approach. In view of the problems incurred by the assumption of k-fold patterns, the system provides an interface for the user to mark the relevant information block that he/she is interested in. Meanwhile, to spare post processing for finer information extraction, the system offers a drill-down and roll-up operation to manipulate information slots. Detail description of these operations are given below:

- Enclosing relevant information block
  Users can mark a block and use the "Ctrl+C" hot key to assign the block as relevant information block. In comparison to previous work by WIEN, Stalker, Softmealy, etc. which require users to annotate the record boundary and the beginning and ending of each slot, IEPAD only requires users to mark the global scope of relevant information.

- Drilling down an information slot
  Drill-down operation allows users to navigate from current less detailed data to more detailed data. It is realized by an encoding hierarchy based on markup language syntax and general string custom. We borrowed this term from OLAP data cube operations [5].

- Specifying relevant information slots
  The result of the enclose and drill-down operations is presented by a spread sheet with multiple slots decomposed from the enclosed information block. Users can then specify relevant slots by mark them for extraction or further expansion.

Encoding Hierarchy

The drill-down operation in OLAP data cube is realized by stepping down a *concept hierarchy*. For example, we might define a concept hierarchy $year > quarter > month > day$ for *time*. The concept hierarchy for the drill-down operation in IEPAD is composed of a set of encoding schemes: markup-level encoding scheme > text-level encoding scheme > word-level encoding scheme, etc. The greater-than sign indicates that the left encoding is a higher level abstraction of the right one. For example, IEPAD introduced *block-tag encoding scheme* which concerns the structure of Web pages represented by HTML tags. In this encoding scheme, IEPAD **sees** only block-level tags where each block-level tag X is encoded as a tag

token <X>; any other text between two block-level tags is encoded as a special token <TEXT> [2]. In addition, *all-tag encoding scheme* [2] which concerns all HTML tags can be a lower level abstraction in the encoding hierarchy.

The encoding schemes not only translate the enclosed information block into a token string, but also enforce a natural segmentation of the data by the tokens they encodes. In other words, an encoded token string of length n can segment the data into n slots. We call the segments the *primitive data* of the tokens. The primitive data for each slot will further be encoded by the next level encoding scheme at drill-down operation. For example, the primitive data of the slots segmented by block-level encoding scheme can be encoded by all-tag encoding scheme at the drill-down operation. For other semi-structure documents such as XML, the encoding can be defined by the layer position of the tags in the parse tree of an XML page.

Both block-tag or all-tag encoding are markup-level encoding schemes and can be applied to any Web pages written in English or Chinese, etc. For further drill-down operations, we can establish text-level and word-level encoding schemes where some of them might depend on the constituents of individual languages. For example, text data (without markups) are made of paragraphs and sentences separated by control characters such as new-line (NL), carriage-return (CR), punctuation symbols such as period, comma, question mark, etc. In addition, we can consider word-level encoding schemes which concern the constituents of sentences – words separated by blank spaces, tabs, etc. Other symbols such as parenthesis, dollar signs ($), colons, and dashes etc. are also encoded at a proper level.

It is worth mentioning that at this level, language information may be included when defining these token classes. For example, we may define word token class to be separated by blank spaces for English, but this does not fit for character-based language such as Chinese and Japanese. Take another example, many free text IE systems require part-of-speech tagging, partial parsing and semantic interpretation which, however, are beyond the scope of this paper. In the context of semi-structured IE, we focus ourselves on simpler text data segmented from Web pages. The complete encoding hierarchy used in this paper is listed in Table 1 which shows a common knowledge for string composition. The system applies an encoding scheme by recognizing the delimiters which are not surrounded by quotation marks. From this hierarchy, it may feel like a gap between the text-level and word-level encoding schemes. There is no general encoding scheme to translate a long text string into a proper representation. In fact, text-level encoding is designed for long text strings to focus on the number of sentences. Word-level encoding, on the other hand, is designed for short text strings which usually require further processing such as email address, date, etc.

Record Identification

In this extension of IEPAD, the input can be one-record Web pages or multiple-record Web pages. We describe here how data are split into records when a user encloses an information block at one of the given p training pages. The enclosed block will first be translated into a token string by a proper encoding scheme either chosen by the system or specified by the user. Similar to [2], the user can specify the number of records in the enclosed block for data splitting. Let k be the number of records in the enclosed block. If k is greater than 1, then IEPAD constructs a PAT tree to discover the k-fold patterns as the *primitive record rule*, P_r. Otherwise, the encoded token string of the whole enclosed block is saved as the primitive record rule. Note that if k is not specified, IEPAD will conduct an exhaustive enumeration for all k from 1 to $n/2$ where n is the length of the encoded token string (given the minimum pattern length 2). As the number of patterns can be increasingly large, these patterns have to be validated by their regularity and density as defined in [2]. In addition, we add another test on primitive data in this extension as described next.

With the primitive record rule, the systems will perform pattern matching on each of the rest $p-1$ pages to find every occurrence of the primitive record rule in the encoded token strings. However, it's more than just finding the occurrences of a pattern P_r in the encoded token string of a Web page. The system will make a comparison between the primitive data of each matched tokens to ensure the correctness of the matching. In other words, the primitive record rule describes the contextual structure of the records, while the primitive data of each token describes what it really is to compensate the lost of information during the encoding process. The test on primitive data is enforced through out the paper so that the matching is correct not only for a exterior resemblance but also a interior resemblance.

The comparison of the primitive data for two tokens can be estimated literally or by their representation at the lower levels of the encoding hierarchy depending on the length. For long segments of primitive data, it is usually hard to compare them character by character. Therefore, we need some abstraction of such data for comparison. For example, for a TEXT token from block-tag encoding scheme, we would consider its encoded representation at all-tag encoding scheme and align respective token strings to find its edit distance (discussed in the next section). The length can be a helpful information too. We define $\frac{|x-y|}{max\{x,y\}}$ as the distance between two primitive data of length x and y. The length distance as well as the alignment distance constitute the evaluation for long segments.

Rule Generalization

Generalization is necessary when multiple records are given as training example. If there is only one record for training, the primitive record rule as well as the subsequent user-operation on the specified slots will be saved as the extraction rule. For example, the user may perform drill-down operation on a specific slot to enforce the segmentation of the primitive data until the desired slice of information can be extracted. Meanwhile, relevant information slots can be specified and given proper attribute names. The corresponding encoding scheme as well as the positions of the specified slots will then be record in the ex-

Category	Encoding scheme	Delimiters
Markup-level	Block-tag	<X>| X: block-level tags
	All-tag	<X>| X: text-level tags
Text-level	Paragraph	NewLine, CarriageReturn, Tab
	Sentence	Period, Question Mark, Exclamation Mark (followed by a blank space)
Word-level	Phrase	Colon, Comma, Semicolon, Bracket, Quotation Mark
	Word/Numeric	Blank
		Other delimiters: @, -, $, /, Period, etc.

Table 1: The Encoding Hierarchy for Web Documents

traction rule such that the extractors generated by IEPAD can perform the same encoding and extraction of the specified slots as recorded.

For multiple records identified from the data splitting procedure, we need a generalization over multiple instances. Let's say m token strings are discovered after data splitting. We will apply string alignment procedure to the m token strings to generalize the induced record rule. Multiple string alignment has been applied in IEPAD [2] to generalize the presentation of the critical common features and tolerate exceptions induced by missing attributes. The major problem in previous work is the decision of an alignment when multiple alignments have the same minimum *edit distance*. The edit distance between two strings are defined as the summation of the matching score between two aligned tokens. In previous work [2], the value for matching a token against a hyphen is 1, matching two same tokens charges no score, and a larger value 3, is given for matching two different tokens to avoid such alignments. With such matching scores, all of the following alignments for "*dtbt*" and "*dtbtbt*" will have the same edit distance 2.

$$\begin{array}{cccccc} d & t & b & t & b & t \\ d & t & b & t & - & - \end{array} \quad , \quad \begin{array}{cccccc} d & t & b & t & b & t \\ d & t & - & - & b & t \end{array}$$

In order to find a correct alignment, the match function needs to consider more than just facile tokens, especially on matching non-delimiter tokens (e.g. `TEXT` tokens) since such tokens actually represent some contents that have been abstracted. Therefore, the score of matching two tokens is defined by the alignment result of their primitive data as described in Section 3.2. With this new matching function, the system can make a better alignment and summarize the *signature representation* as a regular expression from the alignment of multiple token strings. For example, suppose we have the following alignment for three token strings "`dtbtbt`", "`dtbt`" and "`dtbat`". The signature representation will be expressed by "`dtb[a|-]t[b|-][t|-]`", which divides an information slot into 7 sub-slots.

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

The repetitive process of drilling down an information slot and aligning multiple instances can be considered as the decomposition of rule generalization process. We might conduct such processes for thousands of times without thinking. The decomposition of the process into several operations enables the program to simulate the work and discover the knowledge. As long as the matching score of two abstracted tokens are properly defined, the alignment will give a result similar to what we expect.

4. APPLICATIONS

In this section, we show two examples of this approach to solve the extraction of one multiple-record page and multiple singular pages, respectively.

Multiple-Record Page Extraction

The first application is about the extraction from Web pages containing multiple records. The difficult part is to discover the boundary for each record in the Web page. We have described above how IEPAD solves this problem by PAT-tree construction and multiple string alignment [2]. In this extension of IEPAD, the problem of multiple alignments and fine extractions are further overcome. First, the PAT-tree is constructed over the enclosed information block which excludes k-fold ($k > 2$) patterns in other part of the Web page. More importantly, the definition of matching score between two tokens has greatly improved the alignment result and reduced the number of alignments. Second, the extraction of fine information is addressed by the drill-down operation along the encoding hierarchy.

Figure 1 shows a snapshot of the interface after a serial of operations: enclosing relevant information block, drill-down on several information slots, and finally specifying relevant ones and save them as extraction rule. The top extraction pattern is expressed as `"<P><TEXT><BR>[<TEXT>]<BR><TEXT><BR><TEXT>"` and is accompanied by four patterns for each of the four `<TEXT>` tokens accordingly. For example, the first and the third patterns are `"<B><TEXT></B><TEXT>"` and `"<TEXT><FONT><TEXT></FONT><TEXT>"`,
respectively. Drill-down on the third slot of the pattern `"<TEXT><FONT><TEXT></FONT><TEXT>"`, we get pattern `",<TEXT>,<TEXT>"`. Through the upper-right window, the users can type in the attribute names and select the desired information blocks by clicking the check boxes above each block. For example in this figure, we have chosen slot 1 and 2 for "title", and "description", respectively.

The 'score" value can be extracted by sub-block 1 of block 3. The saved extraction rule can then be used to extract other Web pages fetched from the same Web site.

Multiple Singular Page Extraction

The second example regards the application to the extraction from multiple Web pages, each containing one record information. When the number of expected records k equals 1, IEPAD does not need the process of building PAT trees for record boundary discovery compared to one plural page extraction. Instead, the system matches the encoded token string (which is considered as the record pattern) against other training pages. Next, the system conducts multiple string alignment over the discovered records to summarize the record extraction rule. The following drill-down and slot-selection operations are similar to those for multiple-record page extraction. Figure 2 shows the alignment result of three singular pages.

It is worth mentioning that the training pages are not necessarily singular. We may choose one singular page for block enclosing, and take other plural pages for pattern matching. In this framework, as long as record pattern can be identified, pattern matching can be applied to other training page, either singular or plural, to recognize other training examples. In the ultimate situation, the enclosed block equals the whole page and a large number of slots may be segmented. Since this will increase the difficulties for slot-selection, it may need some kind of mechanisms to filter irrelevant slots. However, this problem is beyond this paper.

5. CONCLUSION

With the growth of the amount of online information, the availability of robust, flexible IE systems will become a stringent necessity. The application of pattern discovery based approach to IE can save a lot of efforts since no labeling is required. By taking the advantages of HTML tags and other delimiters, we can encode a page layer by layer and at the same time divide it into blocks. This is the basis of data segmentation and is shown to be successful in discovering the patterns for page template. Based on this idea, we have proposed a robust approach for IE and solve the problems for text token alignment and the filtering of useless blocks.

In this paper, we have extended IEPAD to handle a richer set of semi-structured documents. The proposed operations: enclosing relevant block, drilling down and specifying relevant information slots can reduce the user burden for annotation. With hierarchical encoding, IEPAD perform the same task of Stalker but with less user effort. The embedded hierarchical structure for the records in a Web is discovered automatically by recursive data splitting and string alignment techniques. Therefore, it can handle exceptions such as missing attributes and multiple attribute values. This approach is language independent, since it incorporates only generalization rule of markup language. Therefore, it is adapts to English, Chinese, etc. This approach can be applied to one-record or multiple record page extraction. We have not yet incorporated semantic generalization, however, it is easy to add such generalization into the encoding hierarchy. We believe the proposed framework has minimized the user burden required for learning extraction rule.

References

[1] M. Califf and R. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1997.

[2] 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 681–688, Hong-Kong, May 2–6 2001.

[3] J. Cowie and W. Lehnert. Information extraction. *cacm*, 39(1):80–90, Jan 1996.

[4] 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.

[5] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2001.

[6] 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.

[7] I. Muslea. Extraction patterns for information extraction tasks: A survey. In *Proceedings of AAAI'99: Workshop on Machine Learning for Information Extraction*, 1999.

[8] 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.

[9] E. Riloff. Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 811–816, 1993.

[10] E. Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044–1049, 1996.

[11] S. Soderland. Learning to extract text-based information from the world wide web. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 233–272, CA, USA, 1997.

[12] 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.

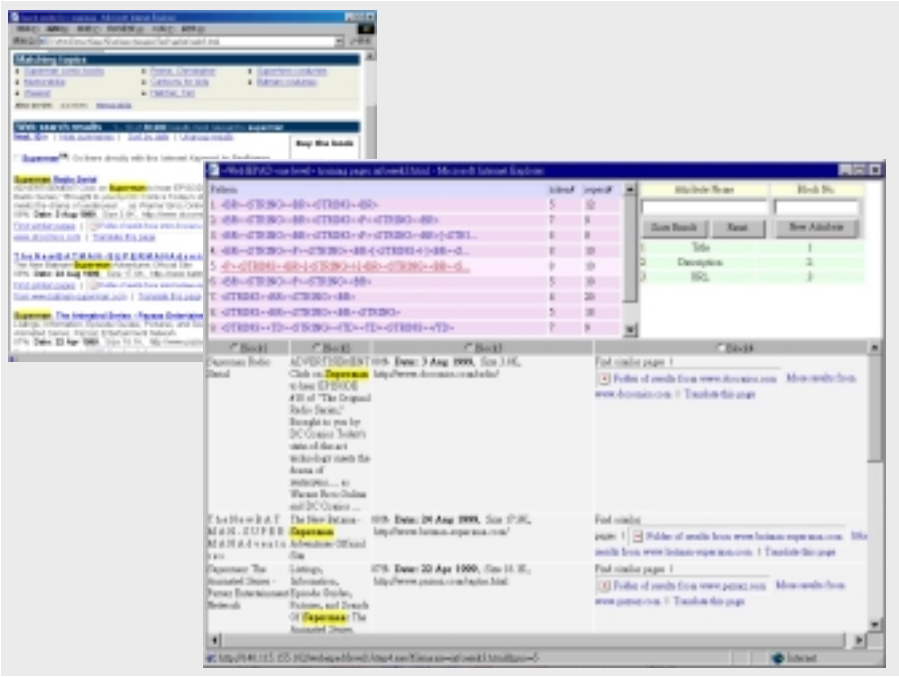

Figure 1: Multiple-record page extraction.

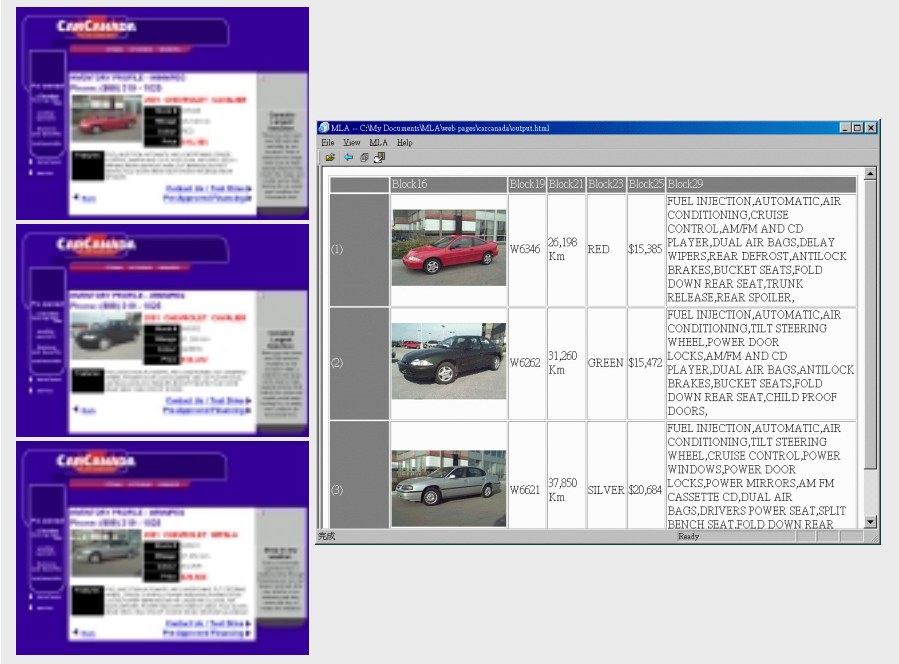

Figure 2: Alignment result for multiple singular pages.