

Applying Pattern Mining to Web Information Extraction

PS Web Information Extraction, SS 2005

"Proseminar Wissenschaftliches Arbeiten"

Jeremy Solarz (0330879)

April 15, 2005

1 Introduction

Due to the large information available over the internet the mechanisms for information extraction became highly important. The problem is to conform extraction techniques to match different types of sites. One approach uses training data to produce extraction rules by itself but it is very time consuming. Another one uses the ability of pattern mining which extracts patterns from web sites and doesn't need human intervention.

2 Structure of a Page

As we can see when we use a search engine the result is produced in a repetitive structure. If we ignore the content of the page and look only at its structure we see partly equality. We can use this parts as a template for the structure of the document.

If we use following example:

```
<html>
<head></head>
<body>
<table>
  <tr><td>Text1</td></tr>
  <tr><td>Text2</td></tr>
</table>
</body>
```

We transform the given example and replace the text with `Text(_)` and the tags with `Html(<tag>)`.

The resulting string shows many repeated patterns. And with this repetitions there come a new definiton. The so called "*maximal repeats*".

Definition The substring a occurs in a string S k times. For at least one pair i, j ($1 \leq i < j \leq k$) the p_{j-1} th token must be different for the p_{i-1} th token (called left maximal). And for another pair x, y ($1 \leq x < y \leq k$) the $p_x - |a|$ th token must be different for the $p_y - |a|$ th token (called right maximal).

3 Pattern discovery

We now want to get a specific level of the structure of the given string representing the web page. With level, we mean a part of the structure. We distinguish between block-level tags and text-level tags. These are further divided as shown in 1. Using this levels we get only the specified part of the whole page structure. For example getting the block-level tags results in:

```
"<table><tr><td>Text(_)</td></tr><tr><td>Text(_)</td></tr></table>"
```

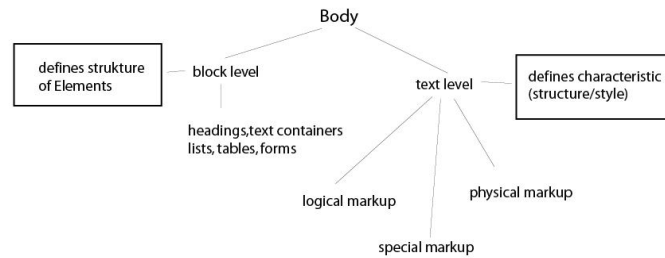


Figure 1: Structure levels [1]

4 Building the PAT Tree

Now we take the given substructure from before and encode it as a binary string. We do this by assuming a three bit value to each tag given in the substructure. Ends in: 000 010 100 110 101 011 010 100 110 101 011 001. The next step is the transformation of the bit string into a PAT-Tree.

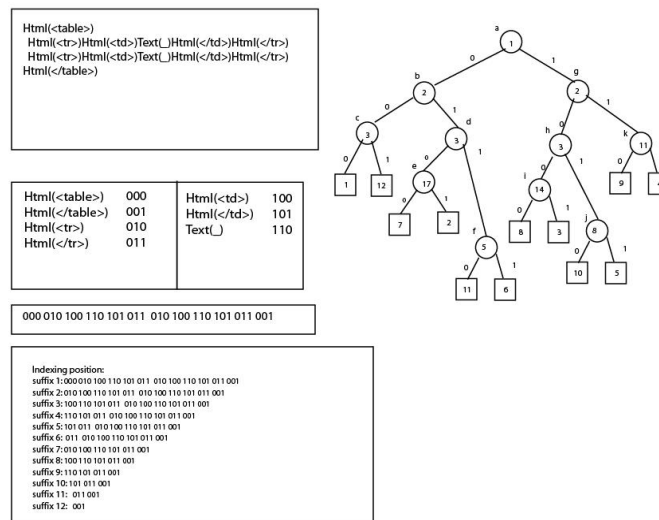


Figure 2: The PAT Tree [1]

First of all we build the possible suffixes. The first suffix is the previously built bit string. We now go to the next tag in the subpart of the page structure ("|<td>Text(_)</td></tr><tr><td>Text(_)</td></tr></table>") and build the bit string for it. This is our second suffix. The last step must be repeated until there
| |

are no remaining tags.

So we are ready to build our PAT-Tree. The number in the nodes is an index and the number over the edge is a bit value. For example if we have a Node 1 and the take the left branch (0) then at node 2 we take the right branch we get the prefix 01. If we want an specific suffix then we must build an unique prefix for it. If we want to get the count of repeats of a tag we only have to build an nonunique prefix.

5 Validate the max repeats

It somtimes happen that you get a big amount of maximal repeats. To get the best of them we should restrict the result by additional conditions.

Regularity We use the "standard deviation of the interval between two adjacent occurrences $(p_{i+1}-p_i)$, that is, the squnce of spacing between two adjacent occurrences $(p_2-p_1), (p_3-p_2), \dots, (p_k-p_{k-1})$ " [1]

Compactness value "of the density of a maximal repeat" [1].

The best way to use these attributes for the choice of patterns is to define a treshold. If a pattern is above this treshold it can be discarded.

6 Partitioning

If you have splitted results (because of a banner between the results) the compactness of the maximal repeats inhibits the adding of the pattern to the result. So we use the algorithm on them which leads to the real result.

7 Multiple String Aligment

The last step how we can improve our results is with mutiple string alignment. Given the following three strings

```
f g h i k j
f g h x k l
g g h i k -
```

we can generalize the extraction pattern as follows `[g|f]gh[i|X]k[j|l|-]`

8 Conclusion

The presented algorithm for information extraction is very useful if we operate on web sites that produce information in a repetitive structure. But normally we search for information not generated this way. So if you search for a specific term the shown algorithm isn't of much use.

References

- [1] Chia-Hui Chang, Shao-Chen Lui, Yen-Chin Wu, Applying Pattern Mining to Web Information Extraction. *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2001.
- [2] Gonnet, G.H.; Baeza-yates, R.A.; and Snider, T. 1992. *New Indices for Text: Pat Trees and Pat Arrays*. Information Retrieval: Data Structures and Algorithms, Prentice Hall.