

A Hierarchical Approach to Wrapper Induction

STALKER

1. Introduction

<Short introduction of myself>

2. The purpose of Stalker

Today we have big information everywhere around us – a lot of different information. The question is now, how we can explore the data and compare against different sources.

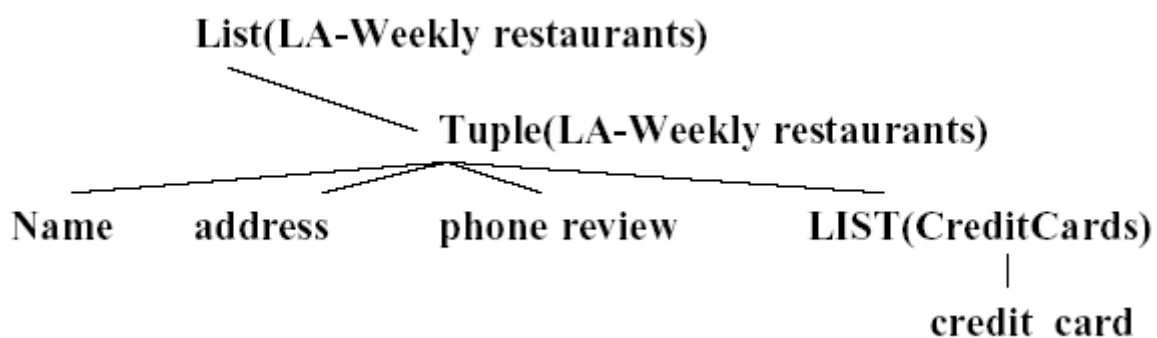
The problem of extraction data is that the syntax is quite well described, but the semantic is not.

A small example of what we want to get out from a source could maybe a webpage that that provides some information about restaurants.

3. How can we describe sources and its content?

```
1: <p> Name: <b> Yala </b><p> Cuisine: Thai <p><i>
2: 4000 Colfax, Phoenix, AZ 85258 (602) 508-1570
3: </i> <br> <i>
4: 523 Vernon, Las Vegas, NV 89104 (702) 578-2293
5: </i> <br> <i>
6: 403 Pico, LA, CA 90007 (213) 798-0008
7: </i>
```

Given the example above we want to get out the Name of the restaurant, so we use the *EC* description. This description is a hierarchical tree that splits up every page in k-tupel. Each node is tupelo with either a list L or a leaf.



Given this notation and an extraction rule, each element can be extracted easily.

A Hierarchical Approach to Wrapper Induction

STALKER

Example how this notation works:

R1 = *SkipTo()* That means skip each character until you find a tag

**1: <p> Name: **

R2 = *SkipTo()* Describes the end of the restaurant name

So we call R1 and R2 end rules. The problem is that R1 and R2 can be occurring more than once so we melt R1 and R2 to R3 or R4.

R3 = *SkipTo(Name) SkipTo()* - ignore everything until you find a landmark called "Name" and ignore everything until you find a tag

and

R4 = *SkipTo(Name Symbol HtmlTag)* ignore everything until you find a 3 token landmark -> Name – sign followed by an HTML tag

This grammar extraction is similar a turing machine – quite simple.

To finish this grammar extraction process we define a R6.

R6 = *SkipTo(AllCaps) NextLandmark(Number)*

- ignore everything until you find an AllCaps word was found and check if next landmark is a number.

R5 and R6 are just variants of skipTo's.

In the few minutes I will now explain the meaning

4. Extraction Rules as Finite Automata

Given the example above, each SkipTo() defines a landmark – if you define more landmarks, you will get landmark automata. One special landmark is the linear landmark – that is defined by a sequence of tokens and wildcards.

The definition of a landmark automata is, that they are nondeterministic automata within a transition $S_i \rightarrow S_j (i \neq j)$ with a label $l_{i,j}$.

A Hierarchical Approach to Wrapper Induction

STALKER

This defines defines the Simple Landmark Automata SLG.

5. Learning extraction rules

Learning means, that the user marks up a point of interest –this could be an area code in source or something else. This is a manual approach – that means user interaction is required.

E1: 513 Pico, Venice, Phone: 1-800-555-1515

E2: 90 Colfax, Palms , Phone: (818) 508-1570

E3: 523 1st St., LA , Phone: 1-888-578-2293

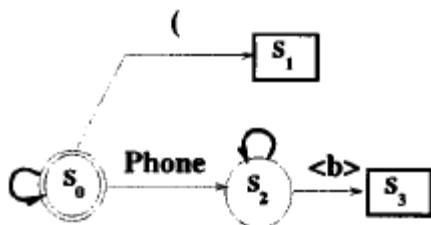
E4: 403 Vernon, Watts , Phone: (310) 798-0008

Stalker now generates a LA –

R1 -> R1 ::= SkipTo() - for E2 and E4 its ok, but for the other source this rule fails

In the second step we just inspect the failed E1 and E3 and generate a new LA –
R2 -> R2 ::= SkipTo(Phone) SkipTo()

this picture describes the rule R1 and R2



Stalker tries to take just the positive examples and not the failed – It is a typical sequential algorithm that tries to figure out the disjunctive. For this purpose an initial Set of candidates is prepared. This set will is refined in every iteration step until the best candidates are found.

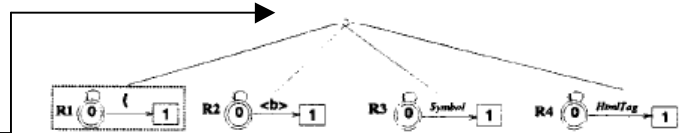
A Hierarchical Approach to Wrapper Induction STALKER

```

STALKER( Examples )
- let RetVal be an empty SLG
- WHILE Examples ≠ ∅
  - aDisjunct = LearnDisjunct(Examples)
  - remove all examples covered by aDisjunct
  - add aDisjunct to RetVal
- return RetVal
  
```

```

LearnDisjunct( Examples )
- Terminals = Wildcards ∪ GetTokens(Examples)
- Candidates = GetInitialCandidates( Examples )
- WHILE Candidates ≠ ∅ DO
  - let D = BestDisjunct(Candidates)
  - IF D is a perfect disjunct THEN return D
  - FOR EACH t ∈ Terminals DO
    Candidates = Candidates ∪ Refine(D, t)
  - remove D from Candidates
- return best disjunct
  
```



Every initial candidate is a 2 state landmark automata within a transition $S \rightarrow S_i$ that is labeled by a landmark automata or ends within a wildcard or Prefix(p)

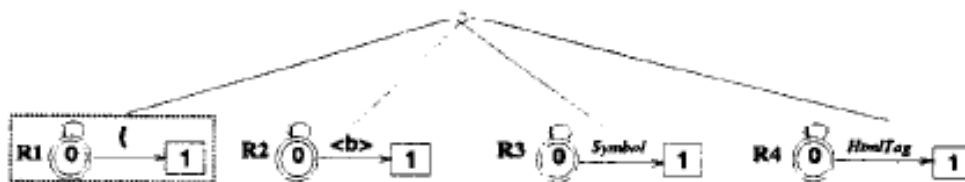
The refine function tries to find a better disjunctive by using a landmark refinement and adds new states to the automaton.

6. Example of Rule Induction

Lets have closer look to the restaurant example.

We want extract the postal code from the E1, E2, E3 and E4 -> after the first iteration, LearnDisjunct creates the first initial candidates.

R1 + R3
R2 + R4 – with wildcards



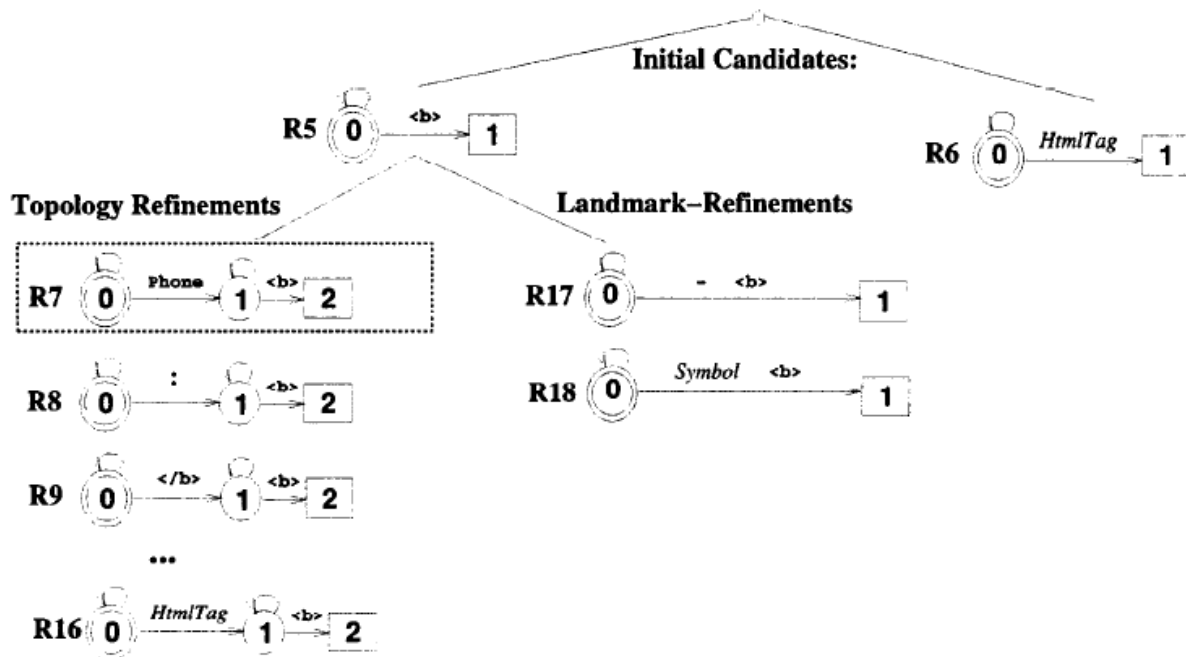
LearnDisjunct – returns R1 – a perfect disjunct and the first iteration ends.

In the second iteration – LearnDisjunct is invoked with the uncovered training examples E1 and E2 and computes the set of refining terminals.

{Phone : , . HtmlTag Word Symbol }

A Hierarchical Approach to Wrapper Induction STALKER

Then LearnDisjunct generates the initial candidates rules R5 and R6.



As both candidates accept the same false positives LearnDisjunct randomly selects the rule to be refined first -> R5. By refining R5 STALKER creates the topology refinement R7, R8 until R16 together with the landmark refinement R17 and R18.

No we can see, that R7 is perfect disjunctive – it covers all examples E1 and E3.

7. My personal conclusion of Stalker

In my opinion it is a good algorithm to extract data from different websites, because STALKER does not care about missing tags and not well formed code.

On the other hand it could be really exhausting to mark each value that I am interested in and compute the rules.

It is quite a good idea – because in my opinion it concentrates on the reduce to the maximum principle. The only think what I have missed in my paper is the explanation of how the user is involved. It is said that the user has to markup – but what kind of tool was used is not clear.

The algorithm is quite stable for non dynamic pages, because if values are often changed you have to do all the work again.