Boosted Wrapper Induction

Dayne Freitag
Just Research
Pittsburg, PA, USA

Nicholas Kushmerick
Department of Computer Science
University College Dublin, Ireland

Presented By
Harsh Bapat

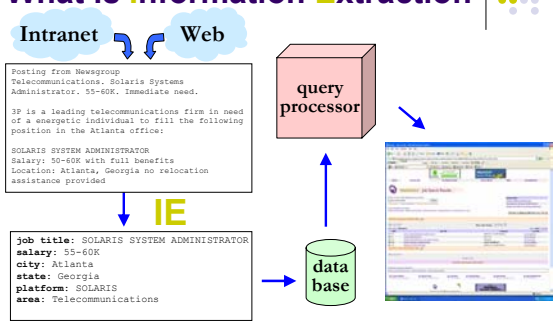
Agenda

- Background
- Wrapper Induction
- Boosted Wrapper Induction
- Experiments and Results
- Conclusions

What is Information Extraction

- Extracting facts about pre specified entities, relationships from documents
- Facts entered into database
- Facts are analyzed for different purposes

What is Information Extraction

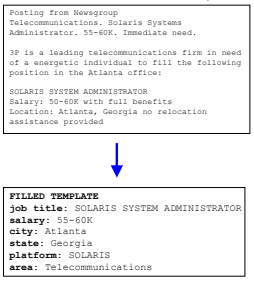


Approaches to IE

- Traditional task - Filling template slots from natural language text
- Wrapper Induction - Learning extraction procedures (wrappers) for highly structured text
  - Learn following patterns to extract a URL
    - ([< a href =“], [http]) – start of URL
    - ([.html], [”>]) – end of URL
  - Extracts http://abc.com/index.html from “.....<a href=“http://abc.com/index.html”>……”

The “traditional” IE task

- Input
  - Natural language documents (newspaper article, email message etc.)
  - Pre specified entities, templates
- Output
  - Specific substrings/parts of document which match the template.



Problems with traditional approach

- Writing IE systems by hand is difficult and error prone
  - Writing rules by hand is difficult
  - Tedious write-test-debug-rewrite cycle
- The machine learning alternative
  - Tell the machine ***what*** to extract
  - Learner will figure out ***how***

Agenda

- Background
- Wrapper Induction
- Boosted Wrapper Induction
- Experiments and Results
- Conclusions

Wrapper – Standard Definitions

- Program interfaces running of one program from other program
- Data which precedes or frames the main data
  - E.g. Headers in data transmission
- Databases – Who has permission to see/change the wrapped data

Wrapper – Standard Definitions

- Feature Subset Selection Problem
  - Selecting features relevant to target concept
- Wrapper Method
  - Retrain and Re-evaluate for different feature subsets
- Filter Method
  - Optimize simple performance criteria

Wrapper – IE definition

- Wrapper – Procedure for extracting structured information (entities, tuples) from information source
- Wrapper Induction – Automatically learning wrappers for highly structured text

Wrapper Example


```
<HTML>
<TITLE>Some country codes</TITLE>
<BODY>
<B>Some Country Codes</B><P>
<B>Congo</B><I>242</I><BR>
<B>Egypt</B><I>20</I><BR>
<B>Belize</B><I>501</I><BR>
<B>Spain</B><I>34</I><BR>
<HR><B>End</B></BODY></HTML>
```

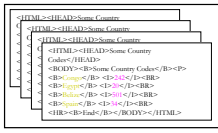
•Can use <B>, </B>, <I>, </I> for extraction
•Wrappers – Extraction patterns

```
procedure ExtractCountryCodes
    while there are more occurrences of <B>
        1. extract Country between <B> and </B>
        2. extract Code between <I> and </I>
```

Wrapper Induction

examples ⟺ hypothesis

Indian food is spicy.
Chinese food is spicy.
American food isn't spicy. ⟺ Asian food is spicy.



⟺ wrapper

Agenda

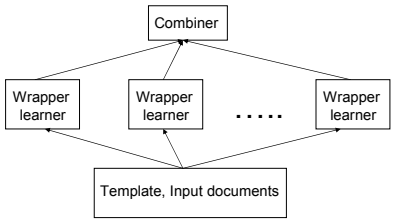
- Background
- Wrapper Induction
- Boosted Wrapper Induction
- Experiments and Results
- Conclusions

Boosted Wrapper Induction

- Wrapper Induction
  - Highly effective with rigidly structured text
- What about unstructured (natural language) text?
- Use Boosted Wrapper Induction
  - Performs IE in traditional (natural text) and wrapper (rigidly structured) domain

Boosted Wrapper Induction

- Use the Boosting trick



IE as Classification Problem

- Documents – sequence of tokens
- Fields – token subsequences
- IE task – identify field *boundaries*
- Boundaries – space between adjacent tokens
- Learn X() functions

$$X_{begin}(i) = \begin{cases} 1 & \text{if i begins a field} \\ 0 & \text{otherwise} \end{cases} \qquad X_{end}(i) = \begin{cases} 1 & \text{if i ends a field} \\ 0 & \text{otherwise} \end{cases}$$

given training examples { <i, X(i)> }

Wrapper Definition

- Pattern – sequence of tokens ([speaker :] or [dr.])
- Pattern matches token subsequence if tokens are identical
- Boundary detector or *detector*

`d = <p, s> where` `p - prefix pattern`
`s - suffix pattern`

- Detector `d=<p,s>` matches boundary `i` if
  - `p` matches tokens before `i` and `s` matches tokens after `i`

$$d(i) = \begin{cases} 1 & \text{if d matches i} \\ 0 & \text{otherwise} \end{cases}$$

Detector has confidence value C_d

Boundary Detector Example

Boundary Detector: [speaker:][dr.<Alpha>]

prefix *suffix*

Matches: "..... Speaker: Dr. Joe Konstan..."

Wrapper Definition

- Wrapper W is
- W = <F, A, H>
  - F = {F_1, F_2,..., F_t} F_i is detector
    - "Fore" detector identifies field-starting boundaries
  - A = {A_1, A_2,..., A_t} A_i is detector
    - "Aft" detector identifies field-end boundaries
  - H : [-infinity, +infinity] ⟶ [0, 1]
    - H(k) – probability that a field has length k

Wrapper Definition

- Every boundary i given "fore" and "aft" scores
  - $F(i) = \Sigma_k C_{F_k} F_k(i)$
  - $A(i) = \Sigma_k C_{A_k} A_k(i)$
- Wrapper classifies token <i, j> as -

$$W(i, j) = \begin{cases} 1 & \text{if } F(i)A(j)H(j-i) > t \\ 0 & \text{otherwise} \end{cases}$$

Wrapper Example

W:

F1=<[speaker:], [dr.<Alpha>]>

A1=<[dr.<Alpha>], [<Punc> University]>

Matches

Speaker: Dr. Joe Konstan , University of Minnesota, Twin-Cities.

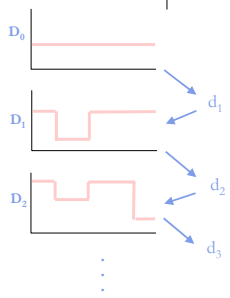
BWI Algorithm

- Learn "T" weak fore/aft detectors, use boosting
- Estimate H
  - Use frequency histogram H(k)

BWI – Boosting

- AdaBoost()
- $D_0(i)$ – uniform distribution over training examples
- For t = 1 to T
  - **Train**: use distribution D_t to learn weak hypothesis d_t
  - **Reweight**: modify distribution to emphasize examples missed by d_t

$D_{t+1}(i) = D_t(i)\exp(-C_{d_t} d_t(i)(2X(i) - 1)) / N_t$



BWI – Weak Learning Algorithm

- Start with null detector
- Greedily pick best prefix/suffix extension at each step
- Stop when no further extension improves accuracy

score(d) = sqrt(W_d^+) - sqrt(W_d^-)

$C_d = \frac{1}{2} \ln[(W_d^+ + \varepsilon) / (W_d^- + \varepsilon)]$

W_d^+ is total weight of correctly classified boundaries

W_d^- is total weight of incorrectly classified boundaries

Wildcards

- Special token matching one of a set of tokens
  - <Alph> - matches any token containing alphabetic characters
  - <ANum> - contains only alphanumeric characters
  - <Cap> - begins with an upper-case letter
  - <LC> - begins with a lower-case letter
  - <SChar> - any one-character token
  - <Num> - containing only digits
  - <Punc> - punctuation token
  - <*> - any token

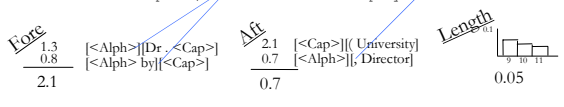
Boosted Wrapper Induction

Training	Extraction
Input: labeled documents	Input: Document, Wrapper, t
Fore = AdaBoost fore detectors Aft = AdaBoost aft detectors H = length histogram	Fore score F(i) = $\Sigma_k C_{F_k} F_k(i)$ Aft score A(i) = $\Sigma_k C_{A_k} A_k(i)$
Output: Wrapper W = <Fore, Aft, H>	Output: text fragment <i, j> {<i, j> | F(i)*A(j)*H(j-i) > t}

BWI Extraction Example

```
<0.26.4.95.09.31.03.bg02+@andrew.cmu.edu.0>
Type:     cmu.andrew.official.cmu-news
Topic:    Chem. Eng. Seminar
Dates:    2-May-95
Time:     10:45 AM
PostedBy: Bruce Gerson on 26-Apr-95 at 09:31 from andrew.cmu.edu
Abstract:

The Chemical Engineering Department will offer a seminar entitled
"Creating Value in the Chemical Industry," at 10:45 a.m., Tuesday, May 2
in Doherty Hall 1112.
The seminar will be given by Dr. R. J. (Bob) Pangborn, Director, Central
Research and Development, The Dow Chemical Company.
```



Confidence of "Dr. R. J. (Bob) Pangborn" = 2.1*0.7*0.05 = 0.074

Sample of learned detectors

SA-stime:

…Time: **2:00** – 3:30 PM #...

F1=<[time :], [<Num>]>
A1=<[ ], [- <Num> : <*> <Alph> #]>

CS-name:

…cgi-bin/facinfo?awb">**Alan#Biermann**</A><…

F1=<[<LC> <*> <*> <Punc> <*> <*> <ANum> " <Punc>], [<FName>]>
A1=<[# <LName>], [< <Punc> a <Punc> <Punc>]>

Agenda

- Background
- Wrapper Induction
- Boosted Wrapper Induction
- Experiments and Results
- Conclusions

Experiments

- 16 IE tasks from 8 document collections
  - Traditional domains - Seminar announcements, Reuters articles, Usenet job announcements
  - Wrapper domains – CS web pages, Zagats, LATimes, IAF web email, QS stock quote
- Evaluation methodology
  - Cross validation
    - Data set randomly partitioned many times
    - Learn wrapper using training set
    - Measure performance on testing set
- Performance evaluation metrics
  - Precision (P) – fraction of extracted fields that are correct
  - Recall (R)– fraction of actual fields correctly extracted
  - Harmonic mean F1 = 2 / (1/P + 1/R)

Experiments – Boosting

- Fixed look-ahead at L=3
- Varied T (rounds of boosting) from 1 to 500
- T_{peak} depends on difficulty of task
  - SA-stime achieves peak quickly
  - SA-speaker requires 500 rounds
- Wrappers tasks – few rounds required
  - Expected because highly formatted data

Experiments – Boosting

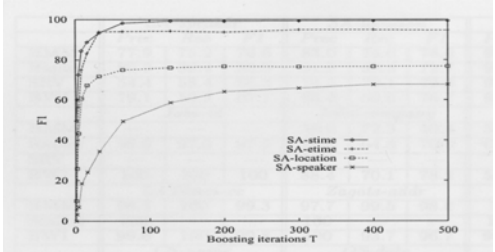


Figure 4: F1 performance on the *SA* tasks as a function of the number of rounds of boosting T.

Experiments – Look-Ahead

- Performance improves with increasing look-ahead
- Training time increases exponentially with look-ahead
- Deeper look-ahead required in some domains
  - IAF-altname requires 30-token “fore” boundary detector prefix

Experiments – Look-Ahead

L	*speaker*	*location*	*stime*	*etime*
1	7.0	40.1	27.7	7.4
2	51.9	76.6	95.0	84.9
3	67.7	76.7	99.4	94.6
4	69.3	75.5	99.6	93.9

Figure 5: F1 performance on the *SA* tasks as a function of look-ahead L.

Experiments – Wildcards

- Performance increases drastically in traditional IE tasks
- Additional improvement after adding task-specific lexical resources in form of wildcards
  - E.g. <FName> matches tokens in a list of common first names released by US Census Bureau

Experiments – Wildcards

wildcards	*speaker*	*location*	*stime*	*etime*
none	15.1	69.2	95.7	83.4
just <>*	49.4	73.5	99.3	95.0
default	67.7	76.7	99.4	94.6
lexical	73.5	—	—	—

Figure 6: F1 performance on the *SA* tasks as a function of various wildcard sets.

Comparison with Other Algorithms

- Other Algorithms
  - SRV(Freitag 1998), Rapier(Califf 1998)
  - HMM based algorithm
  - Stalker(Muslea *et al.* 2000) wrapper induction algorithm
- BWI has a bias towards higher precision
- BWI achieves higher precision while maintaining a good recall

Comparison with Other Algorithms

	SA-speaker			*SA-location*			*SA-stime*			*SA-etime*		
	Prec	*Rec*	*F1*	*Prec*	*Rec*	*F1*	*Prec*	*Rec*	*F1*	*Prec*	*Rec*	*F1*
HMM	77.9	75.2	76.6	83.0	74.6	78.6	98.5	98.5	98.5	45.7	97.0	62.1
Rapier	80.9	39.4	53.0	91.0	60.5	72.7	93.9	92.9	93.4	95.8	96.6	96.2
SRV	54.4	58.4	56.3	74.5	70.1	72.3	98.6	98.4	98.5	67.3	92.6	77.9
BWI	79.1	59.2	67.7	85.4	69.6	76.7	99.6	99.6	99.6	94.4	94.9	93.9
	Jobs-id			*Jobs-company*			*Jobs-title*			*Acq-acq*			*Acq-dlramt*		
HMM	—	—	—	38.6	72.3	50.4	53.2	63.0	57.7	32.8	29.2	30.9	49.3	63.5	55.5
Rapier	98.0	97.0	97.5	76.0	64.8	70.0	67.0	29.0	40.5	57.3	19.2	28.8	63.3	28.5	39.3
SRV	—	—	—	—	—	—	—	—	—	40.7	39.4	40.1	48.1	67.0	56.0
BWI	100	100	100	88.4	70.1	78.2	59.6	43.2	50.1	55.5	24.6	34.1	63.4	42.6	50.9
	IATimes-cc			*Zagats-addr*			*IAP-altname*			*IAP-ory*		
HMM	98.5	100	99.3	97.7	99.5	98.6	1.7	90.0	3.4	16.8	89.7	28.4
Stalker	100	—	—	100	—	—	100	—	—	48.0	—	—
BWI	99.6	100	99.8	100	93.7	96.7	90.9	43.5	58.8	77.5	45.9	57.7
	CS-name			*QS-date*			*QS-vol*					
HMM	41.3	65.0	50.5	36.3	100	53.3	18.4	96.2	30.9			
Stalker	—	—	—	0	—	—	0	—	—			
BWI	77.1	31.4	44.6	100	100	100	100	61.9	76.5			

Figure 7: BWI compared with four competing algorithms on sixteen tasks.

Conclusions

- IE – central to text-management applications
- BWI
  - Learns contextual patterns identifying beginning and end of relevant field
  - Uses AdaBoost, repeatedly reweights training examples
  - Subsequent patterns focus on missed examples

Conclusions

- BWI
  - Bias to high precision – learned patterns are highly accurate
  - Reasonable recall – dozens or hundreds of patterns suffice for broad coverage
- Performs traditional and wrapper domain tasks with comparable or superior performance

References

[1] *Boosted Wrapper Induction* - Freitag, Kushmerick

[2] *Wrapper Induction for Information Extraction* - Kushmerick, Weld, Doorenbos

Boosted Wrapper Induction

Comments by : Sameer Apte

Key Questions for the BWI algorithm

- What effect does no. of rounds of boosting T have on performance?
  - T = 1 to 500
  - No. of rounds for peak performance depends on difficulty of task
  - Easy tasks (*SA-stime*) quickly achieve peak performance, difficult ones (SA-speaker) take more rounds

Key Questions for the BWI algorithm

- What is the effect of look-ahead parameter L on performance?
  - Performance improves with increasing look-ahead L
  - Training time increases exponentially with look-ahead

Key Questions for the BWI algorithm

- How important are wildcards?
  - Drastic increase in performance on traditional IE problems
  - When lexical resources were added as wildcards, better performance achieved

Key Questions for the BWI algorithm

- How does BWI compare with other learning algorithms on the same tasks?
  - Extractors produced by BWI achieve higher precision (no. of correct extractions /total no. of extractions) than other learners
  - Also manage good recall (no. of correct extractions/total no. of fields actually present in the documents)

Boosted Wrapper Induction

Dayne Freitag
Just research
Pittsburgh,Pa,USA

Nicholas Kushmerick
Dept. of Computer Science
University College Dublin,Ireland

Presented by Harsh Bapat

Comments by Anand Sivaraman

Information Extraction

- Converting text (articles, web pages) into structured data objects
- For automatic processing
- Easy for highly structured data
  - (e.g.)Web pages generated by CGI scripts
- Problematic for conventional text

Wrappers

- Contextual Patterns
- Simple
- Accurate
- Formally,

W = <F,A,H>
F – "fore" detectors
A – "Aft" detectors
H(k) – probability that a field has length k

Wrapper Induction

- Hand coding wrappers – tedious
- Wrapper induction
  - Automatically learns wrappers
  - Highly accurate
  - Harder for complicated content /less rigidly-structured formatted content

Boosted Wrapper Induction

- Technique to extract from traditional (natural text) domain
- Learn simple wrappers
- Use Ada-Boosting
- Combine predictions from numerous extraction patterns