An Efficient URL-Pattern Based Algorithm for Accurate Web Page Classification

CSE Seminar, 2016

Pawan Dubey

August 29, 2016

Manipal Institute of Technology

Preamble

Authors

Yiming Yang, Lei Zhang, Guiquan Liu, Enhong Chen at the University of Science and Technology of China

Motivation

- The paper proposes an efficient algorithm to classify web pages based on nothing but their URLs.
- Presented at the 12th International Conference on Fuzzy Systems and Knowledge Discovery, 2015

Colophon

- Made with $\LaTeX$ version 3.1415926. Presented with Beamer
- Source available at

  `http://github.com/pawandubey/seminar`

Introduction

Web Page Classification

- Assigning web pages to one of the predefined categories
- Difficult because of the amount of data
- Useful for contexual action

State of the Art

- Reading page text, hyperlinks etc
- Natural Language Processing techniques on the page text

Problems

- Too slow and inefficient
- Complicated models

URL Based Approach

URLs are the cheapest to access

- Speed boost
- Many relevant features due to SEO

Current Methods

- Use the same content-based techniques
- No incremental learning methods

Background on URL Patterns

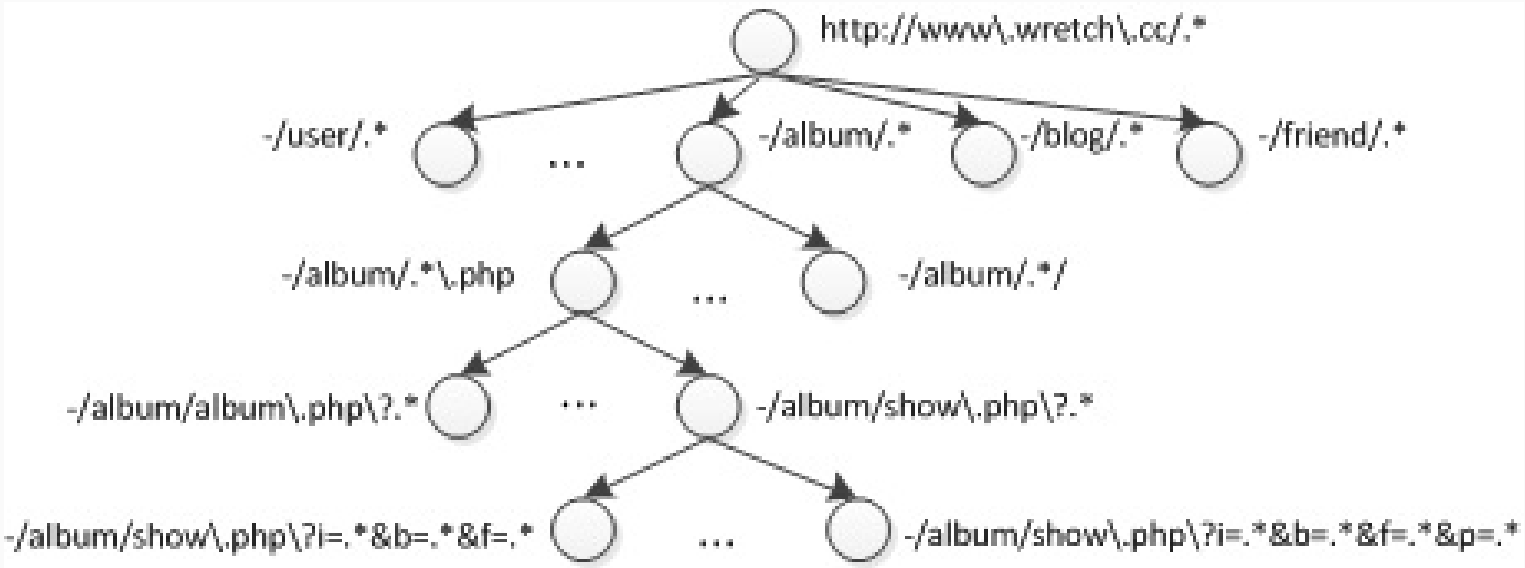
URL-Patterns

What are URL Patterns? A regular expression matching the URLs of a group of related pages.

- `http://manipal.edu/*` matches all pages related to Manipal University
- `http://manipal.edu/mit/*` matches all pages related to MIT, Manipal

Pattern Tree

A `Suffix Tree` like structure, instead for regexes.



Pattern Tree Node

Each node contains a key-value representation of the decomposed components of the URLs that matches the corresponding regex.

Example *https://google.com/search?q=regex*

- *Protocol* : HTTPS
- *Authority* : google.com
- *Path* : /search
- *q* : regex

The Algorithm

Pattern Tree Construction

Algorithm 1 pattern tree algorithm

Input: The set of URLs,S_{url}; The set of keys that have been processed K_{done};

Output: A root node for URLs in S_{url},N_{root};

1: Calculate entropy H(K) for keys that occur in over β percent of URLs of N_{root};
2: Select the new partition key K^* that minimize H(K);
3: $V \leftarrow$ salient values in K^*, trivial values as '*';
4: **if** all values in V are trivial values or size of $S_{url} < \tau$ **then**
5: **return** N_{root};
6: **end if**
7: **for all** v in V **do**
8: $S_v \leftarrow$ URLs whose value of K^* equals to v in S_{url};
9: $N_{new} \leftarrow$ pattern-tree construction(S_v,$K_{done} \cup K^*$);
10: Add N_{new} to children of N_{root};
11: **end for**
12: **return** N_{root};

Pattern Tree Construction

- For each key decomposed from the URL set, we determine entropy

Shanon Entropy

$$H(K) = \sum_{i=1}^{T} \frac{n_i}{N} log \frac{n_i}{N} \tag{1}$$

We choose the K^* which minimizes this value as the splitting variable.

Pattern Tree Construction

- We select all values for the key K^* and divide them as salient or trivial
- Division based on probablistic methods to find the decline of the frequency
- Recursively apply for all salient values of K^*
- Stop if either only trivial values left or number of URLs less than threshold

Problems

- Recursive - inefficient
- Not incremental

Improvement

- An incremental modification to the algorithm

Incremental Pattern Tree Construction

- Only reconstruct tree if the K^*_{new} is different from K^*_{old}
- Check if the new URLs match an existing node
- If yes, add to matching node and recursively update
- Create new nodes only if no match found for new URLs

Pros

- Only update a subtree of the whole tree

Pattern Generation

- Relationship between keys are determined
- Pattern is generated by topologically sorting the set of keys
- Final pattern set is constructed by looking at the patterns of leaf nodes

Classification

Binary

- Pattern for only one of the classes are matched with the URL
- Useful for tasks like sentiment analysis

MultiClass

- Patterns for all classes generated in advance
- Pattern weight w_{ij} determines class of URL
- w_{ij} is the number of URLs matching $pattern_j$ in $class_i$
- The longest matching pattern for the URL with each $class_i$ is selected as $candidate_i$

$$label_{url} = \max_{i \in class_m} (w_{candidate_i}) \tag{2}$$

Evaluation

Performance Measures

In terms of Information Retrieval, if the problem is to retrieve relevant documents from a dataset,

- Precision is the fraction of retrieved documents which are relevant
- Recall is the fraction of relevant documents that are retrieved
- F1 score is the harmonic mean of precision and recall

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{3}$$

- Running Time

Setup

- WebKB dataset
- Comparison with *Web Classification Using N-gram Based URL Features* by *R.Rajalakshmi*
- And *Rule Based Method*
- 90-10 Training-Test split

Functional Classification

To validate effectiveness and efficiency in classification

Results

Category	NBUF Method with SVM			NBUF Method with ME			UPCA		
	P	R	F1	P	R	F1	P	R	F1
student(1641)	79.0%	94.2%	85.9%	80.6%	79.7%	80.1%	80.8%	93.0%	**86.5%**
faculty(1124)	83.6%	54.5%	65.9%	66.1%	64.3%	65.2%	85.5%	63.4%	**72.8%**
course(930)	97.5%	77.0%	**86.0%**	86.6%	84.0%	85.3%	97.2%	70.0%	81.4%
project(504)	96.0%	45.3%	61.5%	70.8%	64.2%	**67.3%**	93.1%	50.9%	65.9%
average	89.03%	67.5%	74.8%	76.0%	73.1%	74.5%	89.2%	69.3%	**76.7%**

Take aways

- UPCA is about 10X faster than NBUF
- Achieves an improvement of 2% on F1
- Performs better on larger datasets

High Quality Page Identification

Prove effectiveness on a dataset with much noise.

Results

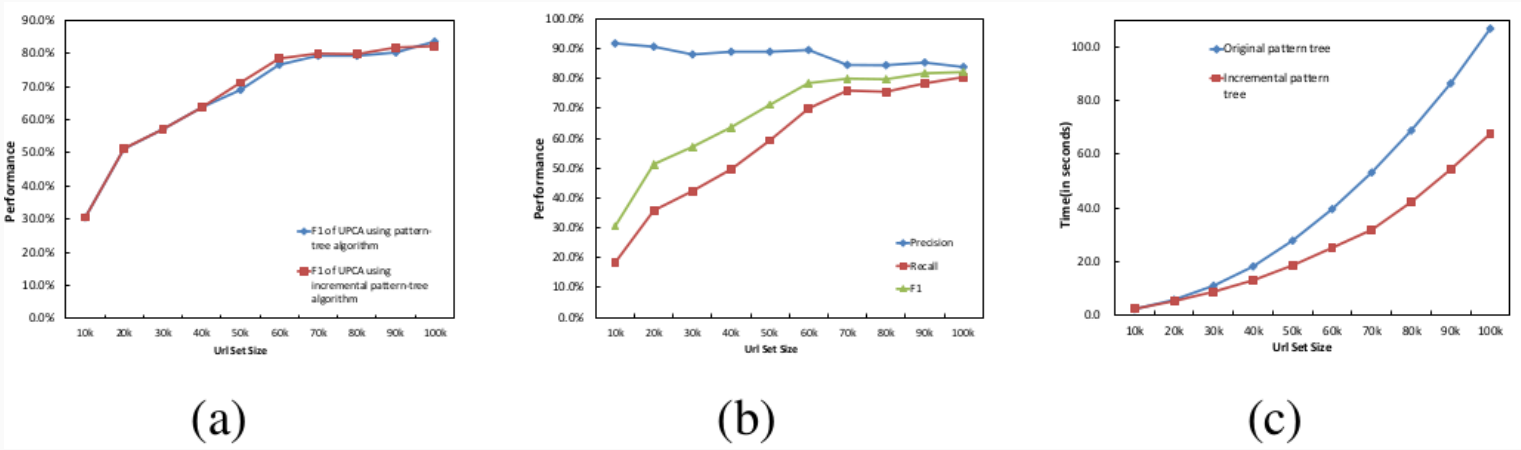
Hosts	Rule-Based Method			NBUF(SVM)			UPCA		
	P	R	F1	P	R	F1	P	R	F1
3g.ifeng.com	99.14%	47.5%	64.25%	96.15%	30.49%	46.30%	79.2%	82.1%	**80.6%**
m.dangdang.com	16.9%	53.6%	25.6%	-	-	-	17.8%	90.5%	**30.0%**
wap.ganji.com	58.7%	37.0%	45.5%	61.90%	13.54%	22.22%	64.4%	100%	**78.4%**
wap.lashou.com	88.0%	66.1%	75.7%	25.00%	3.87%	6.70%	89.9%	98.7%	**93.6%**

Take aways

- UPCA is clearly superior to the *Rule Based Method* approach

Conclusion

Performance Comparison



- UPCA provides a much more efficient way to classify web pages based on URLs
- The incremental algorithm achieves a performance gain on the recursive one
- A distributed version required for processing very large data sets

Resources and References

References

- Tao Lei, Rui Cai, Jiang-Ming Yang, Yan Ke, Xiaodong Fan, Lei Zhang. *A Pattern Tree-based Approach to Learning URL Normalization Rules.* In Proc. of the 19th International World Wide Web Conference (WWW 2010)
- Y. Lin, T. Zhu, X. Wang, J. Zhang, and A. Zhou. *Towards online review spam detection.* In WWW, pages 341–342. IW3C2, 2014.
- R. Rajalakshmi and C. Aravindan. *Web page classification using n-gram based url features.* In Advanced Computing (ICoAC), pages 15–21. IEEE, 2013.