Automatic Classification of Researcher Web Pages

Jiaxuan Wang	Yifei Li	Zidong Wang	Jin Zhang
Computer Science	Electrical Engineering	Information	Computer Science
University of Michigan	University of Michigan	University of Michigan	University of Michigan
jiaxuan@umich.edu	liyifei@umich.edu	wzidong@umich.edu	jinatum@umich.edu

Abstract

Here goes for the abstract

1. Project Description

With booming of web pages on the Internet, many people and organizations want to build customized websites to express ideas. Among them, some are professional researcher web pages which display achievements and progress of researchers. It is important to automatically distinguish between a researcher web page and a non-researcher web page because it will help academic search engines to filter out those irrelevant pages.

This project serves to answer the question of how researchers are different from non-researchers. To achieve that end, we need a way to model people's website writing. There are several options available. One natural attempt would be tracking researchers' activities on social media and compare researchers' usage of diction and syntax to those of an average person. However, since many researchers are not active social media users, this characterization is biased. A better approach would be to classify researchers by their personal websites because of the prevalent uses of them in the researcher community.

For the project purpose, we will use tf-idf as one of weighting scheme to represent each web page and plan to apply machine learning techniques such as Random Forest, Naive Bayes, and Neural Network for feature selection and classification. We plan to compare those different methods on corpus crawled from 4 distinct universities.

Before rolling out our own plan, it would be helpful to observe what people have done in the industry. Google scholar and Microsoft Academic Search both provide simple ways to broadly search for academic paper and literature. User can search many articles and abstracts from academic publishers, universities, and other websites. However as we know, there is no such tool that can help identify the difference between researcher and non-researcher web pages. Our tool will take data from both websites, perform texture analysis on them.

2. Related Work

(Qi and Davison, 2009) surveys the literature of web page classification. They point out that web page classification in general is usually formulated as a supervised learning problem. Depending on different purposes, web page classification can be further broken down into subject classification (predicting subject/topic of a web page), functional classification (distinguishing categories such as "researcher web page", "personal homepage", and "course page"), sentiment classification (mining opinion/attitude), genre classification (separating comedy from tragedy), and search engine spam

classification. There are also other ways to dissect a classification problem. For example, classification can be grouped into multi-class versus binary, hard assignment versus soft (distributional) assignment, single label versus multi-label, flat (all categories parallel) versus hierarchical. In their work, Web classification can be used to construct web directories (eg. Yahoo web directory), improve quality of search result (disambiguation and topic sensitive page rank), help QA systems, and improve the performance of domain specific search engine (classifier used to decide relevancy).

Although web classification is closely related to text classification, there are some major differences. First of all, a web page is written in HTML which gives structural information that text does not have. Secondly, a web page is more link driven than text. These difference makes web page classification an interesting problem in its own right. Because of the differences between text classification and web page classification, the features describing an web page should be carefully chosen. (Qi and Davison, 2009) points out that there are mainly two categories of features, on page features and the neighbouring features. On page feature includes textual content and tags. They are noisy so that bag of words feature may not work. To address the noisiness, one can use N-gram shingles (at the cost of exponential space). To utilize HTML tags, well tuned linear combination of title, headings, meta-data and main text are proven to be useful. Also, K nearest neighbor can be used for weighing tag importance. However, one problem in using tags is that their function as representational tools is bigger than their semantic importance (which is what we care about). Other methods of on page feature includes document summary (which is shown to improve accuracy by around 10% as compared to content based classifiers), URL based classification (not very accurate but increase time/space efficiency), and visual analysis. One paper that is particularly related to our research problem is (Asirvatham and Ravi, 2001) which focused on classifying research pages based on image. The premise is that researcher pages have more synthetic images compared to background mixed pages. The second kind of feature extraction focuses on extracting features on neighboring pages. The justification for this approach is that features on the target page may be misleading. For example, some web page contain large amount of images or flash object but little textual content which makes it hard for classifiers to infer about its function or topic. The solution is to use information of related pages. There are in general two types of assumptions made. The weak assumption is neighboring pages share some common features and the strong assumption is neighboring pages are likely in the same category. The strong assumption is unsuitable to function classification as personal web pages are likely linked to some hub pages instead to other personal pages.

Since web pages have text and multimedia data, they can be viewed as structured, semi-structured or unstructured. The performance of a classifier highly depends on the feature set that we choose. The presence of redundant or irrelevant attributes may lead to misclassification . Besides, with the size of feature set growing, the time it takes to model the data increases. In order to address the problem of feature selection, (Mangai and Kumar, 2012) proposes a novel feature selection method based on the Ward's minimum variance measure. In the method, they first clusters the feature set into different groups based on Ward's algorithm. Then within each group, they selects the feature that have the largest information gain in predicting the class label. Their experiment shows that this method reduces significantly the dimension of the feature even compare to PCA. Although the size of the feature set shrinks a lot, the classification accuracy is not affected much and very close to the result of classification using the full features and the reduced feature set after applying PCA.

(Riboni, 2002) conducts various experiments on a corpus of 8000 documents belonging to 10 Yahoo categories, using Kernel Perceptron and Naive Bayes classifiers. Author analyzes the peculiarities and tried to exploit them for representing web pages in order to improve categorization accuracy. Author introduces a new method for representing linked pages using local information that makes hypertext categorization feasible for real-time applications. She uses the content of the A element in the HTML structure which is used to link the current page to another page. The words in the description are close

to the subject of the linked pages. She uses this description for representing the linked pages without having to download them.This representation method is surely less powerful than other hypertextual ones (Chakrabarti et al., 1998; Furnkranz, 1999; Joachims et al., 2001; Cristianini et al., 2000) but can be used for real-time categorization and is feasible for every set of web pages. Author uses Composite Kernels for combining the usual representation of web pages base on local words with the simple hypertextual. Experimental results show that, in spite of the mediocre performance of the simple hypertextual representation, assigning a value near to 0.2 to the weight improves classification accuracy, confirming the usefulness of the combination of standard and hypertextual representations. (Shen et al., 2004) proposes a new Web-page classification algorithm based on Web summarization for improving the accuracy. They first give empirical evidence that ideal Web-page summaries generated by human editors can indeed improve the performance of Web-page classification algorithms. They further propose a new Web summarization-based classification algorithm and evaluate it along with several other state-of-the-art text summarization algorithms on the LookSmart Web directory. Experimental results show that the summarization-based classification algorithm achieves an approximately 8.8% improvement as compared to pure-text-based classification algorithm.
They introduce an ensemble classifier using the improved summarization algorithm and show that it achieves about 12.9% improvement over pure-text based methods. The new classifier combines 4 different weak classifiers to gain better classification accuracy. The four methods are:

- Adapted Luhn's Summarization Method

In this extraction-based method, every sentence is assigned with a significance factor, and the sentences with the highest significance factor are selected to form the summary. The summary of a single page will be formed by the sentences with the highest scores.

- Latent Semantic Analysis (LSA)

In LSA, concepts are represented by one of the singular vectors where the magnitude of the corresponding singular value indicates the importance degree of this pattern within the document. Any sentence containing this word combination pattern will be projected along this singular vector. The sentence that best represents this pattern will have the largest index value with this vector.

- Content Body Identification by Page Layout Analysis

In order to utilize the structure information of Web pages, author employs a simplified version of the Function-Based Object Model (FOM). They utilize the model to identify the position of sentence within web page, and assign different weight to sentences based on if they are in the content body or in unrelated tags, such as navigation bar.

- Supervised Summarization

A supervised learning based on feature selection and Naive Bayesian model is also used here to construct the ensemble classifier. The features are: 1. position of sentence 2. length of sentence 3. word distribution 4. similarity between sentence and title 5. cosine similarity between sentence and all text in the page 6. cosine similarity between sentence and meta-data in the page 7. the number of occurrence of words from sentence in special word set 8. average font size in the sentence. NB model trains a summarizer based on the 8 features, and each sentence will then be assigned a score. Author sums up the above 4 summarizers to ensemble a new, strong summarizer. The performance is improved by 12.9% when equal weights are given to the 4 summarizers.

3. Data Description

3.1 Data Collection

We have crawled researchers' web pages from four universities - Stanford University, Massachusetts Institute Of Technology (MIT), University of California Berkeley and University of Michigan

(UMich). We are also planning on using researcher web pages crawled from Microsoft academic search. To balance the proportion of researcher web page from that of the non-researcher web page as to better match the real world situation, we are also crawling data from the deep web (specific method are still under debates).

We used the library Scrapy(scrapy.org) in Python to crawl web page data from the universities listed above. Starting URL of crawlers are set to the faculty search pages, and the crawling depth are set to 2 or 3 so that the crawler would stop at a reasonable point to avoid over crawling. We defined "researcher page" as researcher personal homepage along with lab homepages. All other pages are counted as non-researcher web pages.

Up to date, we have crawled about 2160 pages with 1023 pages annotated. 259 of them are labeled as "researcher pages", and the rest 764 are labeled as "normal pages". Sample "researcher pages" have URLs like "http://history.berkeley.edu/people/richard-m-abrams". Sample "normal pages" have URLs like "http://townsendcenter.berkeley.edu/deadlines". All the web pages are stored in JSON format, with keys including "label", "name", "url" and "description". "label" is defined as a binary value to indicate whether a page is a "researcher page". "name" refers to the title of the page. Partial web pages have empty value corresponding to the "name" key because they do not have title in the HTML. "url" identifies the http address of the web page. "description" corresponds to the content of the web page, which is the content between <body> tag in the HTML file.

3.2 Annotating Method

After crawling pages from different websites, we need annotate them manually. First of all, we define a web page that meets the following criteria as a researcher web page:

- It describes a certain reseacher, including his/her research interest, contact information, research group, students, and any related information
- It contains link to the researcher's publication

Researcher group pages are not counted as researcher web pages. The official web pages like http://www.csail.mit.edu/user/1324 will only be counted as researcher web pages as long as they contain research interest and publication links

In order to speed up the annotating process, we write a python script that opens each URL address we stored and accepts input from the users, which indicating the label of the content of the corresponding URL. The pseudo-code are as followed:

```
Open the file storing the crawled web pages;
        for each pages:
                if it is not annotated:
                        open the page in web browser;
                        wait for input from the user:
                                if it is a researcher web page, input 1
                                if it is a not, input 0
                                if user input other character, ask the user to re-input it.
                                if user input 'q', store the annotated data and quit.
```

Figure 1 shows an example of the interface. We can see from the example that, if we input characters rather than "1" , "0" or "q", we will be asked to re-input the command. If we input "q", we will exit the program and then the annotated data will be stored.

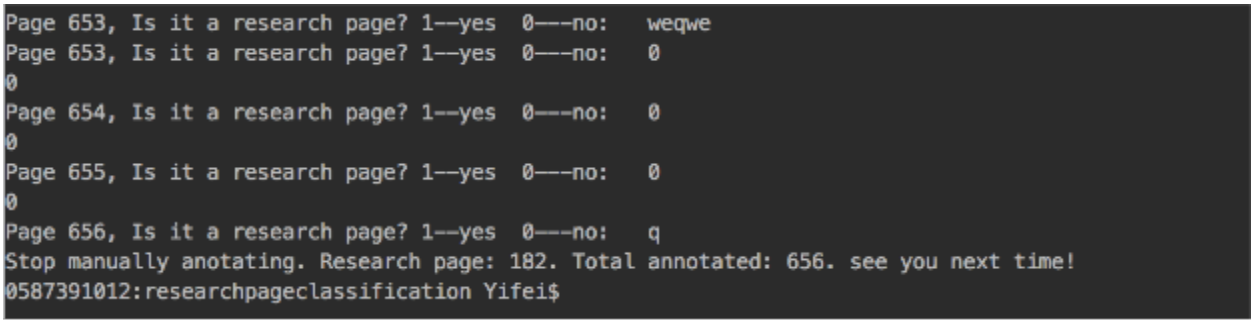

Figure 1. Illustration of annotation method. We first input "weqwe", it is not recognized, so we have to re-input "0". Then we input "q", the program exit and store the annotated data.

Since it is difficult to define a researcher web page accurately, bias will be introduced during the annotating process. Aiming at solving this problem, each page will be annotated 4 times by our group member. Only pages with agreement from at least three people will be selected as researchers web page or non-researcher web pages. Currently most pages are only annotated by one person. So we are going to let other team members to do the annotation for our data set within this week and calculate the agreement.

3.3 Example of Crawled Pages

Figure 2 is a sample page for researcher web page.

Figure 2. Illustration of a researcher web page

The data corresponding to researcher web page stored in JSON is:


```
{
  u'description': [u'<body id="genesis-1c">\n\n ⋯ <body>'],
  u'label': 'researcher',
  u'name': [u'Grossman, Pam'],
  u'url': u'https://ed.stanford.edu/faculty/pamg'
}
```


Figure 3 is a sample page for non-research web page.

Figure 3. Illustration of a non-researcher web page

The data corresponding to non-researcher web page stored in JSON is:


```
{
  u'description':[u'<body><!-- <img alt="full screen background image"
src="images/site/Pepsized_Blur_15.jpg" id="full-screen-background-image" /><!-- CONTAINER
→<div class="container" id="container">··· <body>'],
  u'label': 'nonresearcher',
  u'name': [u'Jiye, Huang'],
  u'url': u'http://jiyehuang.com/'
}
```


3.4 Statistics of Data Set

Table 1 shows the statistics of crawled web pages up to date. These web pages will form the basic training set and test set. We aim to crawling more pages from the non-academic website and annotate them as the project progresses. Currently we will use these crawled pages to help us develop the classification algorithm framework.

Total Crawled	2160
Total Annotated	1023
Researcher Web Page	259
Non Researcher Web Page	764

Table 1. Statistics for crawled data.

4. Method description

Data Preprocess

1) HTML TAG Extraction (on-page content extraction)

Using information from tags can strengthen a classifier's performance. However, only some of the tags are useful for web page classification as other tags are used for visual display. After reviewing previous works, we select the following tags to extract:

Group 1	These tags are representative of a page. They usually contain words marked as part of titles or headings.	<title> <h1> <h2> ..... <h6>
Group 2	This group of tags are used for emphasizing part of texts and distinguish them from regular texts to show their importance	 <b> <i>
Group 3	This tag is used for linking the current page to others	<a href>
Group 4	This tag is used to mark the beginning and end of a text paragraph in web page	<p>
Group 5	Researchers have proved that these tags sometimes contains important concepts	
Group 6	This tag contains meta data information	<meta>

In order to extract these useful information, we utilize BeautifulSoup to parse html files and extract useful information. **Beautiful Soup** is a Python package for parsing HTML and XML documents (including malformed markup, i.e. non-closed tags). It creates a parse tree for pages that can be used to extract data from HTML, which is useful for web scraping.

2) Extract URL information

URL contains rich information regarding web pages. For example, when looking at "http://www.rle.mit.edu/people/directory/elfar-adalsteinsson/", we can guess that it is a web page describing the person "elfar-adalsteinsson". Besides, since the postfix is "edu", this web

page belongs to an academic institution. So in terms of URL, we can first extract its postfix information. Besides postfix, the directory information is also a good signal for classification.

Therefore, the complete process for data preprocessing is shown as follows:

1. For each web page, use BeautifulSoup to extract all the texts and related information within each groups described in table 2, then preprocess them (tokenize, remove stop words) and store them in a dictionary.
2. Besides html tags, also extract postfix(edu/org/com……) and directory (people / directory/)
3. Build inverted index for each feature groups, and remove the words that appears less than n times. These form the initial feature space for the model.

3) Neighboring page

In addition to on page features, it would be helpful to include paragraphs of neighboring pages. In this project, we utilize the information on pages that are at most 2 steps away from the target page. Depending on the topology, there are mainly 5 kinds of neighboring pages: parent, child, spouse, grandparent, and grandchild (currently, we only consider child and grandchild relations as they don't explicitly require a web graph to be built).

The information extracted from each neighboring page is also different depending on their types.

neighbor types	**information extracted**
parent	The paragraph linking to the target page
child	The <h1> <h2> <h3> <h4> <h6> and title tag text
spouse	The summary of the page
grandparent	The link text to the parent page
grandchild	The <h1> <h2> <h3> <h4> <h6> and title tag text

Feature Selection

The initial feature set is usually very large and contains redundant information as well as noise. These redundancy may affect the performance of classifier. Therefore dimensionality reduction is essential. Here, we consider a variety of method for feature reduction. Namely, they are PCA, kmeans, hierarchical clustering with information gain.

Classifier Training

An essential task in any machine learning task is to select a hypothesis space. Here we consider using the following classifiers: linear SVM, rbf SVM, Multinomial Naive Bayes, Decision tree, Adaboost, Random Forest, linear regression, and polynomial regression, K Nearest Neighbor, and Neural Network. A detailed comparison of each method will be included in the result section.

Evaluation Methodology

Since the dataset is unevenly split between researcher and non researcher websites, accuracy is not a good metric to use. Instead, we use mean average precision to evaluate our result, which is essentially the area under precision recall curve.

5. Experiments Results

Currently unavailable

References

[1] http://academic.research.microsoft.com/
[2] https://scholar.google.com/
[3] http://scrapy.org/
[4] Asirvatham, A.P. and Ravi, K,K, 2001. Web page classification based on document structure. Awarded second prize in National Level Student Paper Contest conducted by IEEE India Council.
[5] Dou Shen, Zheng Chen, Qiang Yang, Hua-Jun Zeng, Benyu Zhang, Yuchang Lu, and Wei-Ying Ma. 2004. Web-page classification through summarization. In SIGIR, pages 242–249.
[6] Daniele Riboni, "Feature Selection for Web Page Classification", D.S.I., Universita degli Studi di Milano, Italy, dr548986@silab.dsi.unimi.it
[7] J. Alamelu Mangai, V. Santhosh Kumar, S. Appavu alias Balamurugan. 2012. A Novel Feature Selection Framework for Automatic Web Page Classification. International Journal of Automation and Computing. 9(4), August 2012, 442-448, DOI:10.1007/s11633-012-0665-x
[8] Johannes Furnkranz, 1999. Exploiting structural information for text classification on the WWW. Proceedings of the 3rd Sym-posium on Intelligent Data Analysis (IDA-99), Springer-Verlag, Amsterdam, Netherlands.
[9] Kamal Nigam, Andrew Kachites Mccallum, Sebastia Thrun, Tom Mitchell. 2009. Text Classification from Labeled and Unlabeled Documents using EM. Machine Learning, , 1-34()

[10] Nello Cristianini, John Shawe-Taylor.2000. An introduction to Support Vector Machines (and other kernel-based learning methods), Cambridge University Press,
[11] Soumen Chakrabarti, Byron Dom and Piotr Indyk. 1998. Enhanced hypertext cat-egorization using hyperlinks. Proceedings of SIGMOD-98, ACM International Conference on Management of Data, ACM Press, New York, US, pp. 307-318.
[12] Thorsten Joachims, Nello Cristianini and John Shawe-Taylor. 2001. Composite kernels for hypertext categorisation. Proceedings of ICML-01, 18th International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, US,, pp. 250–257.
[13] Xiaoguang Qi and Brian D.Davison. 2009. Web page classification: Features and algorithms. ACM Comput. Surv. 41, 2, Article 12 (February 2009), 31 pages DOI = 10.1145/1459352.1459357 http://doi.acm.org/10.1145/ 1459352.1459357