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

With booming of web pages on the Internet, it is important to automatically distinguish between a researcher web page and a non-researcher one because it will help academic search engines to filter out irrelevant results. In this project, we first build 7 classifiers using at most 8 group of features. We also propose a novel two-step classifier. Experimental results show that title and paragraph tags are useful in identifying researcher pages while meta tags are less informative. Regarding classifiers, ensemble learners achieve good accuracy on test data. while the two-step classifier performs moderately.

1. Project Description

Digital library systems like Google scholar and Microsoft Academic Search are interested in retrieving scientific research publications. One way to do that is through tracking researchers' personal web pages. Since it is infeasible to crawl information from the entire Internet, digital libraries try to collect only relevant web pages to minimize network bandwidth and hardware usage. A key component for this type of focused crawling is a classification module that identifies potentially useful documents.

This project aims at building such a researcher web page classifier. Our goal is to help digital library systems to efficiently filter out non-researcher web pages and to enrich their database. To achieve that, we first crawl web pages from top universities and use TF-IDF as the weighting scheme to represent each page. Then, we train 7 classical classifiers to compare their performances. We also propose a two step classification framework which achieves result comparable to these classifiers.

In the following sections, we review related works, describe data collection methods, introduce classification pipeline, and analyse the result.

2. Related Work

Web page classification is extensively studied in the literature. In this section, we first define the problem and then review known techniques for feature construction and feature selection.

(Qi and Davison, 2009) surveys the literature of web page classification. Depending on different purposes, web page classification can be broken down into subject classification, functional classification , sentiment classification , genre classification , and search engine spam classification. Qi points out that there are mainly two categories of features used in this context, namely on page feature and neighbour page feature.

On page feature includes textual content and tags of the current page. They are noisy so that vanilla Bag of Words may not work. One can mitigate this problem using N-gram shingles at the cost of exponential space. Other methods of prepossessing on page feature include document summary , URL based classification, and visual analysis. We use all of these methods except visual analysis in this project as it will require knowledge in computer vision. The second kind of feature is neighbor page

feature. The intuition for including it is that on page feature can be misleading. The solution is to use information of related pages. There are two types of assumptions made in using neighbor page feature. The weak one state that neighboring pages share common features with the page of interest and the strong one asserts neighboring pages are usually in the same category. Here, the second assumption does not hold as researcher pages frequently link to hub pages instead of to other researcher pages.

With raw features constructed, the next step is feature selection. The performance of a classifier highly depends on the feature set that we choose. Redundant and irrelevant attributes can lead to not only misclassification, but also increased data modeling time as the feature set grows. The following are known techniques that tackle these problems. (Mangai and Kumar, 2012) proposes a feature selection method based on the Ward's minimum variance. Here, they first cluster the feature set into different groups based on Ward's algorithm. Then within each group, they select the feature that have the largest information gain in predicting a class label. Their experiment shows that this method significantly reduces the dimension of the feature without sacrificing accuracy. (Chen et al., 2009) proposes a novel relevance measure, discriminating power measure (DPM), for feature space reduction. Their result shows that DPM reduces input dimensionality from 10427 to 200 with zero rejection rate and with less than 5% decline in test accuracy. In our work, we use DPM to reduce the dimension of feature space (refer to section 4 in detail).

Other interesting methods for web page classification include an anchor tag based classification (Riboni, 2002) and a summarization based classification (Shen et al., 2004). (Shen et al., 2004) proves that web page summaries generated by human editors can improve the performance of web page classification. They propose an automatic web summarization-based classification algorithm that achieves good result. They also introduce an ensemble classifier using the improved summarization algorithm and show that it gains 12.9% accuracy improvement over pure-text based methods.

3. Data Description

3.1 Data Collection

We crawled researcher web pages from Stanford University, Massachusetts Institute Of Technology, University of Pennsylvania, and University of California Berkeley. To introduce noise in our dataset to better match the real world scenario, we randomly crawled web pages from Yahoo and CNN.

The crawlers are implemented using a Python library Scrapy(scrapy.org) and the seed pages are set to be faculty pages. For Yahoo and CNN, starting URLs are set to be the front pages.

Our original dataset contains 5054 web pages, which are all manually annotated by one of our group-mates with subsets annotated by all of us. Of those pages, 1020 are classified as researcher web pages. To balance the final dataset, we randomly added 1020 non-researcher web pages, so it ended up with 50-50 percent for each category. We take 1500 pages for training and the remaining for testing. Table 1 summarizes the statistics of our final data set. All the web pages are stored in JSON format, with keys including "plabel", "rlabel", "head", "url" and "body". "plabel" is a binary value indicating whether a page is a personal page. "rlabel" is a binary value indicating if a page is a researcher page. "head" refers to the content in the <head> tag (May not present in all pages). "url" identifies the address of the web page. "body" corresponds to the content in the <body> tag of the HTML file.

	Total	Personal	Non Personal	Researcher	Non Researcher
Training Data Set	1500	922	578	750	750
Test Data Set	540	335	205	270	270

Table1. Statistics for our data set

3.2 Annotation Method

We use a hierarchical approach for annotation. We first determine whether a page is a personal page (Figure 2), and then check whether it is a researcher page (Figure 1).

One effective way to quickly obtain an upper bound for classifiers is to use manual annotation agreement. Here all of our team members annotate 100 randomly selected pages from our dataset. A web page annotation is considered to be agreed if at least 3 of us annotated it as such. In this way, we reached 98% agreement on personal page and 93% agreement on researcher page.

Checking whether a page is a personal page is easy compared with checking whether a page is a researcher page because the definition of personal pages is well understood. Correspondingly we reached a higher agreement on personal page annotation. The definition of a researcher page is less clear. A page would be labeled as a research page if and only if it fulfills all of the 3 requirements:

- It is labeled as personal web page
- It describes a certain researcher, including his/her research interest, contact information, research group, students, and any other related information
- It contains links to the researcher's publication or list out the researcher's publication on the current page

In order to speed up the annotation process, we wrote a python script that opens every URL and records input from a user as the label for the corresponding page.

Figure 1. A researcher web page

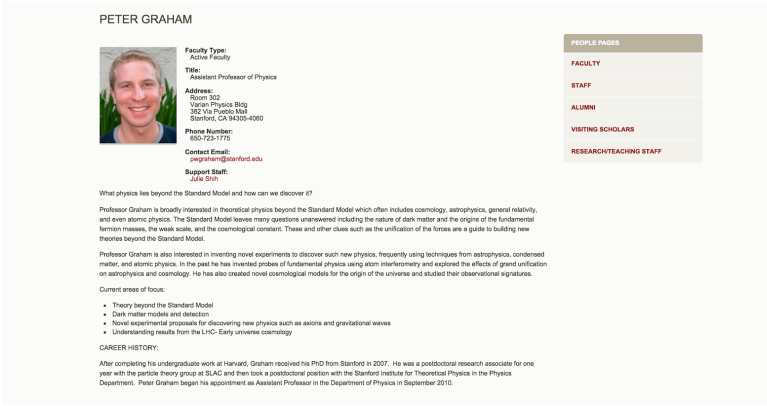


Figure 2. A non-researcher web page

4. Method Description

We implemented four different methods for researcher classification: URL feature, on page feature, neighbor page feature, and 2 step classification. The first three are baseline methods that are inspired by related works and the fourth is an original method that we proposed.

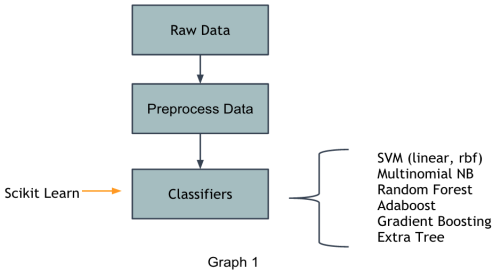
In this section, we introduce the pipelines for classification and look at a feature selection method to reduce dimension.

4.1 Pipeline Overview

The common setup

The work flow of any classification task can be summarized by Graph 1. We first clean the raw data

to a form that is convenient to use and then feed it to a classifier to categorize the documents. For all of our methods, we use the same set of classifiers indicated in the figure, which includes Support Vector Machine with linear and RBF kernel (a standard baseline for most classification problems), Multinomial Naive Bayes (a baseline that performs well on text classification), Adaboost (an ensemble learner that is resistant to overfitting), Gradient Boosting (an ensemble learner that performs well on a variety of tasks), Random Forest (an ensemble learner with little correlation between trees), and Extra Tree (a more randomized Random Forest). They are implemented using a Python library Scikit Learn. For common preprocessing, we tokenize, remove stop words, and stem for each feature group.



Graph 1

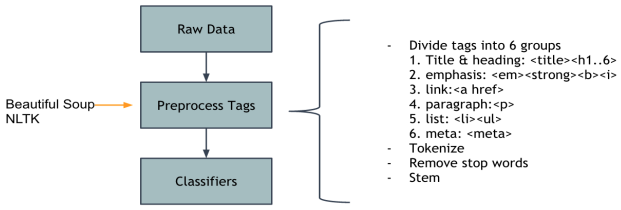
Pipeline: URL feature

The first pipeline classifies web pages solely based on their URLs. URL contains rich information regarding web pages. For example, when looking at www.rle.mit.edu/people/directory/elfar-adalsteinsson/, we can guess that it is a web page describing the person "elfar-adalsteinsson". Since the post-fix is "edu", we know that this web page belongs to an academic institution, which suggest that the person might be a researcher. In the data prepossessing step, we break an URL down into components including path, fragment, query parameter, and host name. Then for each of these parts, we learn a TF-IDF representation of all URLs and feed them to the classifiers.

Graph 2

Pipeline: on page feature

As discussed in related work, there are 2 types of features. We exploit them both. On page feature should be intuitive to understand. We clean our data by dividing HTML tags into semantically related groups. The following table shed some light on the intuition.



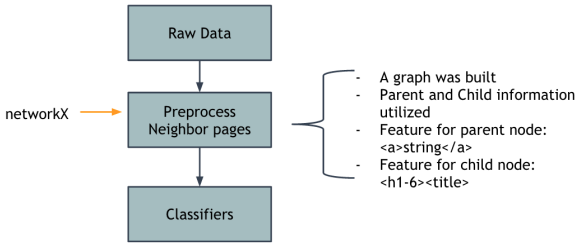
Graph 3

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

Table 2. Explanation of 6 tag groups.

After common prepossessing for each feature group, the learned TF-IDF vectors are fed to the classifiers.

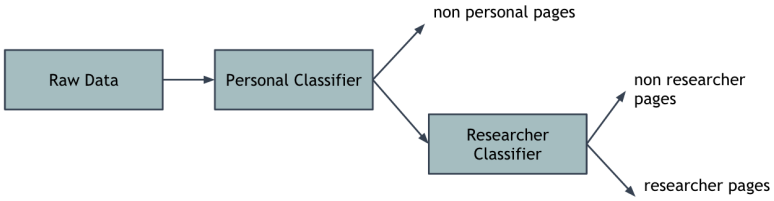
Pipeline: neighbor page feature



Graph 4

Web can be viewed as a graph structure with each node being a web page. It is natural to assume that the best classifier should incorporate this connection information. Indeed, common sense support this conjecture. If the link from the parent page explicitly state that the current page is a researcher page, the classification task becomes trivial. Similarly, if the child page is a publication page, it is very likely that the current page is a researcher homepage. With that intuition, we build a graph on the 5054 raw web pages using networkX. Then we extract the parent and child feature for each page. The feature used for the parent page is the string within the anchor tag pointing to the target page because we believe it is very informative about the target page. The feature used for the child page is the h1 to h6 and the title tag of the child page because a summary of the child page should work well in identifying its parent. Then the same prepossessing steps are applied to form a TF-IDF representation of the page to feed to the classifiers.

Pipeline: 2 step classification (Original Contribution)



Graph 5

In addition to the three baselines, we propose a novel two step classification framework. The key observation is that the set of researcher web pages is a proper subset of the set of personal pages. Therefore we can break down our classification task into two phases. In the first phase, we filter out non personal pages and in the second phase, we only have to classify the remaining pages. Our expectation is that by reducing the feature space for the second classifier, it should perform better.

The two classifiers are chosen to be the best classifiers in the previous pipelines. Feature groups are chosen similarly. The work flow is illustrated in Graph 5.

4.2 Feature Selection

The initial feature set is large and contains redundant information plus noise. These redundancy can affect the performance of classifiers. Therefore dimensionality reduction is essential. For this project, we use the method mentioned in (Chen et al., 2009). For each word in the feature set, we calculate its DPM. Then we rank all the words by their DPM values. The exact procedures for calculating DPM is the following:

Step 1: For each word, calculate the document frequency inside the i^{th} category and document frequency outside the i^{th} category:

$$DF_i^{in} = \frac{n(doc(term_k)_{all}, cat_i)}{n(doc_{all}, cat_i)} \qquad DF_i^{out} = \frac{n(doc(term_k)_{all}, collection - cat_i)}{n(doc_{all}, collection - cat_i)}$$

where $term_k$ is the k^{th} term; cat_i is the i^{th} category; $doc(term_k)_{all}$ are all documents that contain $term_k$; doc_{all} are all documents. $n(doc(term_k)_{all}, B)$ is the number of documents with $term_k$ in collection B. $n(doc_{all}, B)$ is the total number of documents in B.

Then compute the difference for cat_i:

$$\delta_i = |DF_i^{in} - DF_i^{out}|$$

Step 2: For the whole collection, sum up the total difference due to a feature , and select the top ranking words to form our feature space.

$$DPM = \sum_i \delta_i$$

4.3 Alternative Interpretation

While the pipeline view of the methods is intuitive, it is not convenient for evaluation of result. An alternative way to reason about the four methods is to view pipeline 2 and pipeline 3 as 8 feature groups (6 for on page feature, 2 for neighbor feature). We will stick to this interpretation in evaluating result.

5 Experiments Results and Analysis

We did experiments using single feature group and combination of feature groups on both training and test data to determine the usefulness of each group.

5.1 Classification Accuracy for on Page and Neighbour Page Feature

Figure 3-10 presents the experimental results on training data when using one feature group. They show how the size of each feature group affects classification accuracy. Lines with different colors represent different classifiers. From the plots, we see that Linear-SVM and RBF-SVM perform poorly compared to other classifiers. Random Forest, Extra Tree and Gradient Boosting perform better. Title, anchor, paragraph and list tags are useful because using any of them alone can obtain accuracy over 80%. Figure 9 and 10 show that child data appears more informative than parent data. According to these 8 figures, classification accuracy is almost constant with respect to the number of features except for Linear-SVM, RBF-SVM and Multinomial Naive Bayes. This gives us a hint that only a small fraction of features is enough to obtain good accuracy.

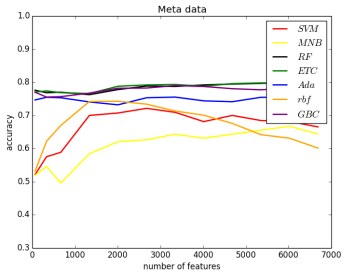


Figure 3. Classification using only

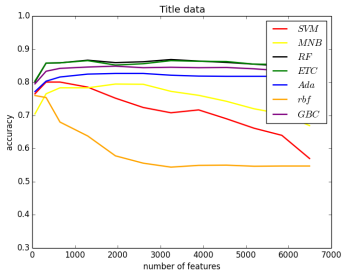


Figure 4. Classification using only

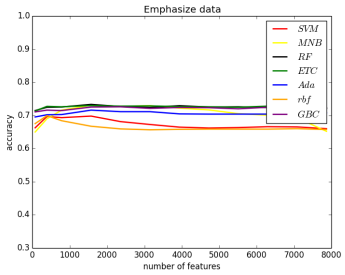


Figure 5. Classification using only

meta data in training data set

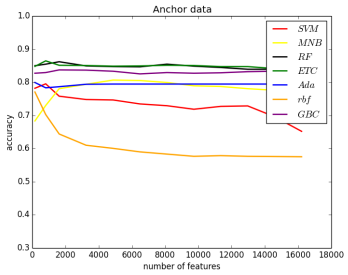


Figure 6. Classification using only anchor data in training data set

title data in training data set

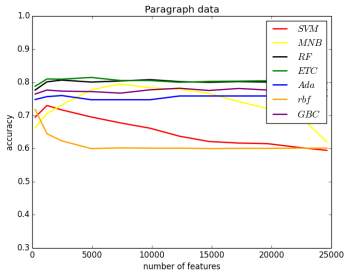


Figure 7. Classification using only meta data in training data set

emphasize data in training data set

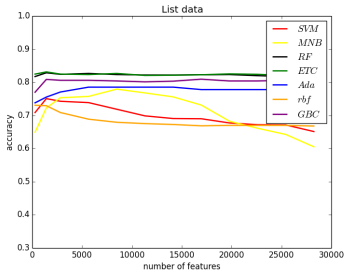


Figure 8. Classification using only list data in training data set

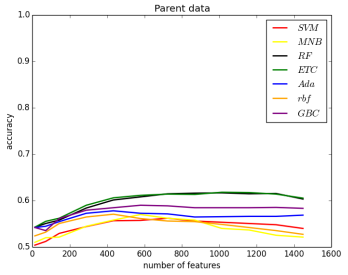


Figure 9. Classification using only parent data in training data set

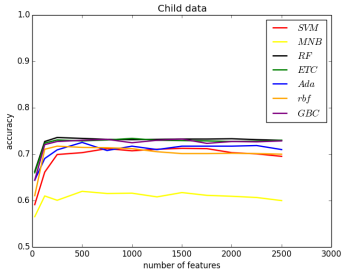


Figure 10. Classification using only child data in training data set

Figure 11-16 show experimental results on test data. We did not include parent and child feature because it is hard to generate the graph if we test only one page each time. From these 6 figures, SVM classifiers still perform badly. Title and paragraph tags are more informative than other groups. We find that more features does not guarantee a better performance. For ensemble learning classifiers, only 10% of features are enough to obtain promising result.

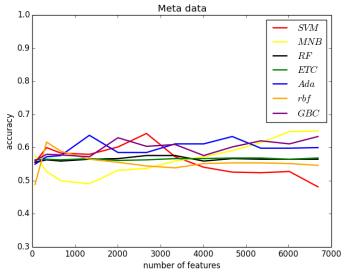


Figure 11. Classification using only meta data in test data set

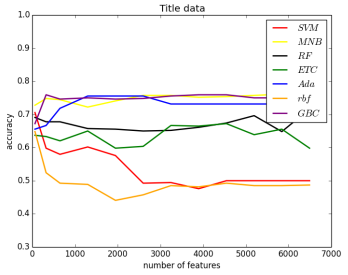


Figure 12. Classification using only title data in test data set

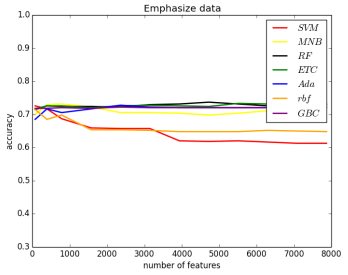


Figure 13. Classification using only emphasize data in test data set

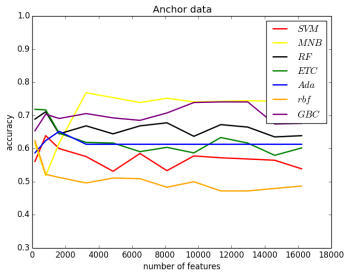


Figure 14. Classification using only anchor data in test data set

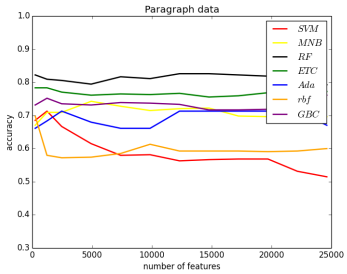


Figure 15. Classification using only paragraph data in test data set

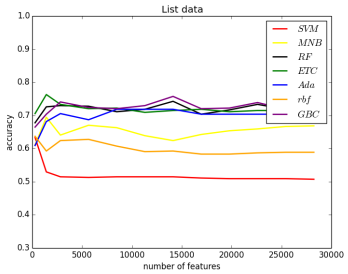


Figure 16. Classification using only list data in test data set

5.2 Classification Accuracy for Leave-one-out Feature Combination

Table 3 shows experimental results on training data when we combine feature groups. If we use all the features, the performance of each classifier is encouraging (about 88%) except for SVM classifiers. Even when we leave one feature out, the classification accuracy is still above 85% for most classifiers. If we test on the test data, the performance shrinks. In table 4, the best result we get when all the features are included is 85% from Multinomial Naive Bayes. SVM classifiers perform only slightly better than a random guess(50%). If we exclude meta tags, Multinomial Naive Bayes, Random Forest and Extra Tree obtain accuracy above 84%. This may imply that information stored in meta tags are less useful. We also find that 7 classifiers perform differently on different feature combinations. For example, when meta tags are not used, Gradient Boosting classifier provides 77% accuracy while Multinomial Naive Bayes reaches 85%; when title tags are not used, Gradient Boosting achieves 85% accuracy but Multinomial Naive Bayes gets 77% accuracy.

Featre	Linear SVM	RBF SVM	Naive Bayes	Random Forest	Extra Tree	Ada Boost	Gradient Boosting
All feature	**0.819**	**0.541**	**0.868**	**0.881**	**0.899**	**0.879**	**0.891**
No meta	0.819	0.541	0.868	0.897	0.897	0.873	0.886
No title	0.810	0.568	0.859	0.876	0.879	0.845	0.876
No Emphasize	0.811	0.541	0.863	0.876	0.892	0.859	0.883
No anchor	0.819	0.545	0.852	0.883	0.895	0.860	0.870
No paragraph	0.787	0.550	0.868	0.899	0.896	0.852	0.885
No list	0.805	0.541	0.870	0.879	0.893	0.857	0.883
No Parent	0.819	0.543	0.868	0.888	0.894	0.879	0.889
No Child	0.815	0.546	0.872	0.887	0.895	0.877	0.887

Table 3. Experiment results for different feature combinations on training data.

Feature	Linear SVM	RBF SVM	Naive Bayes	Random Forest	Extra Tree	Ada Boost	Gradient Boosting
All feature	**0.539**	**0.502**	**0.85**	**0.759**	**0.7**	**0.78**	**0.817**
No meta	0.543	0.526	0.857	0.852	0.846	0.737	0.774
No title	0.567	0.507	0.778	0.809	0.739	0.776	0.856
No Emphasize	0.549	0.500	0.818	0.765	0.676	0.761	0.824
No anchor	0.5839	0.491	0.8	0.772	0.674	0.783	0.874
No paragraph	0.5949	0.494	0.837	0.722	0.637	0.735	0.798
No list	0.6079	0.535	0.817	0.707	0.648	0.709	0.796

Table 4. Experiment results for different feature combinations on test data.

5.3 Classification Accuracy for Other Classifiers

We also experiment with the URL-based classifier and the two-step classifier. The results are shown in table 5. Using only URL, we get a 80% classification accuracy on test data. This is quite promising because training a URL classifier is much faster. In terms of our 2-step classifier, the best result we can get is 81.2% from Gradient Boosting. The poor performance may be caused by the fact that the same algorithm was used for two classifiers without tuning parameters.

	url-based classifier	**2-step classifier**
Linear SVM	0.815	0.501
Naive Bayes	0.794	0.713
Random Forest	0.796	0.771
Extra Tree	0.798	0.769
Ada Boost	0.757	0.711
RBF SVM	0.802	0.500
Gradient Boosting	0.787	0.812

Table 5. Experiment result on two classifiers.

5.4 Words with Discriminative Power

Table 5 shows words with discriminative power within each feature group. Some words appear many times across different groups such as ‘research’, followed by ‘journal’ and ‘public’. These words matches our intuition.

Meta	**Title**	**Emphasize**	**Anchor**	**Paragraph**	**List**	**Parent**	**Child**
list	public	journal	eec	one	win	faculty	Inform
css	abstract	scienc	pdf	univers	sum	alumni	menu
cnn	work	research	research	professor	univers	research	stanford
homepage	contact	symposium	journal	scienc	journal	employ	research
research	biographi	proceed	public	journal	research	event	calendar

Table 5. Words with high discriminative power in each group

5.5 Efficiency of Algorithms

Of all the baselines, URL feature runs the fastest while neighbor page feature runs the slowest as it needs to build a graph dynamically. For the classifiers, ensemble learners runs much slower compared to SVM and Naive Bayes. Therefore, practical implementation should consider using URL feature with Naive Bayes classifier.

6 Conclusion and Future Work

According to experimental results above, we conclude that, some of the feature tags such as title and paragraph tags are helpful in researcher web page classification. Within these tags, tokens such as ‘research’, ‘journal’, ‘public’ have discriminative power in judging a web page’s label. Surprisingly, SVM classifiers did not perform well in our experiments since we didn’t tune parameters. Ensemble learners such as Random Forest, Extra Tree and Gradient Boosting obtain a promising result on both train and test data. This implies that ensemble classifiers may be good choices for web page classification.

Currently we are assigning equal weight to different feature groups, which may not be optimal. For

example, on-page features could possibly outweigh neighbouring page features in terms of discriminative importance. Therefore, future researchers can experiment on different weighting schemes for each feature group. The 2-step classifier did not perform well so far. One possible reason is that we did not tune parameters. Future researchers are encouraged to apply techniques like grid search on the parameter space. Another limitation of our experiment comes from our dataset. Since most data are collected from 4 universities, they are not representative of the entire Internet. Therefore our selected features and trained models are biased. To better scale to become a real world model, our approach have to be trained and tested on broader datasets.

7 Contribution

Jiaxuan Wang: Proposed the architecture of pipelines for this project. Implemented page rank summarization using cosine similarity. Built a graph of 5000 pages (implementation of pipeline 3). Provided a template for machine learning pipeline. Crawled Stanford data.
Yifei Li: Crawled web pages. Annotate each web page. Implemented feature selection framework based on DPM. Preprocessed web page. Generated training vector and test vector using inverted index. Implemented 2-step classifiers. Conduct experiments on different feature groups using different classifiers and evaluated the performance of each feature group.
Zidong Wang: Crawled and annotated web pages. Built template to train classifiers for classifying web pages. Implemented bootstrapping annotation method. Extracted URL features from data set. Implemented classification and evaluation based on URL features.
Jin Zhang: Implemented web page preprocessor. Built inverted index for web page tag and neighbor page. Analyzed annotation agreement. Crawled University of Pennsylvania data. Annotated each web page.

References

[1] 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.
[2] Chih Ming Chen, Hann Ming Lee, and Yu Jung Chang. 2009. Two Novel Feature Selection Approaches For Web Page Classification. Expert Systems with Applications 36(2009) 260-272.
[3] 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.
[4] Daniele Riboni. Feature Selection for Web Page Classification. D.S.I., Universitadegli Studi di Milano, Italy, dr548986@silab.dsi.unimi.it.
[5] 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.
[6] 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.
[7] Kamal Nigam, Andrew Kachites Mccallum, Sebastia Thrun, Tom Mitchell. 2009. Text Classification from Labeled and Unlabeled Documents using EM. Machine Learning, , 1-34.
[8] Nello Cristianini, John Shawe-Taylor.2000. An introduction to Support Vector Machines (and other kernel-based learning methods), Cambridge University Press.
[9] 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.
[10] 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.
[11] 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.