Classification of World Wide Web Documents

Choon Yang Quek
choon@cs.cmu.edu

Advisor: Dr. Tom Mitchell
mitchell@cs.cmu.edu

School of Computer Science
Carnegie Mellon University

Senior Honors Thesis

Abstract

The rich variety of knowledge available on the World Wide Web makes it an attractive target to datamine. A first step to this datamining operation is to be able to classify web pages according to some predetermined ontology. Current machine learning techniques for text classification deal primarily with flat text documents, and do not take advantage of the richer structure offered by the World Wide Web, such as hyperlinks, titles and paragraph headings. This paper investigates several methods designed specifically to classify web pages, compares their relative merits, and shows that using structural information produces classifiers with different performance characteristics.

Acknowledgements

There are many people that I need to thank for their help this past year as I struggled to put this thesis together. The Text Learning Group here at CMU has been most encouraging and helpful with their comments and ideas. The World Wide Knowledge Base Project Group, including Dayne Freitag, Kamal Nigam, Andrew McCallum, Mark Craven, were the people whom I could bounce ideas off, and were always encouraging and nice enough not to laugh when I say naïve things. Thank you for all your help.

Of course, the person I am most indebted to is Tom Mitchell. I think when he agreed to be my advisor, he had no idea what he was in for. He has been most patient with me, setting aside precious time to coach me, even beyond machine learning. He has even helped me learn a new "model" of what research is and isn't, and re-evaluate my outlook. He is at once mentor, colleague, and friend, allowing me to freely embarrass myself.

For all these, and more, thank you, Tom.

Classification of World Wide Web Documents

Choon Yang Quek (Advisor: Dr. Tom Mitchell)

Senior Honors Thesis

Abstract

The rich variety of knowledge available on the World Wide Web makes it an attractive target to datamine. A first step to this datamining operation is to be able to classify web pages according to some predetermined ontology. Current machine learning techniques for text classification deal primarily with flat text documents, and do not take advantage of the richer structure offered by the World Wide Web, such as hyperlinks, titles and paragraph headings. This paper investigates several methods designed specifically to classify web pages, compares their relative merits, and shows that using structural information produces classifiers with different performance characteristics.

1 Introduction

The explosive growth of the Internet, and particularly the World Wide Web, in recent years has put huge amounts of information at the disposal of anyone with access to the Internet. However, the lack of standardized ways of categorizing the available information, compounded by the sheer variety of information, the lack of a standardized structure, and the large number of different sites, makes finding relevant information a laborious and time-consuming task.

The vision is to have software that learns to integrate information available on the World Wide Web and present it in a coherent and cohesive way to the user. As a building block to this larger vision, this paper explores the use of machine learning techniques to classify World Wide Web pages into predefined categories.

Many machine learning techniques have been successfully used for the classification of text documents, but there are currently no classifiers which take into consideration the substantially richer space in which WWW pages reside. We shall present two approaches which make use of the structural information offered by the Web, and ways of combining such classifiers, as evidence that using the richer feature space of the Web provides for more accurate classification of WWW pages.

2 The Problem

Given a limited, fixed ontology [2] of the World Wide Web (WWW) and some training examples of pages in each of the classes within the ontology, we want to be able to train a classifier that will be able to identify new instances of those classes.

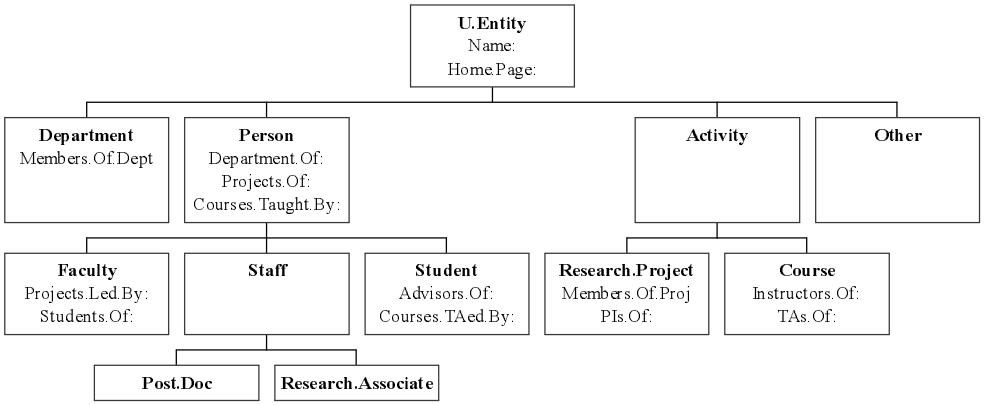

Figure 1. University Ontology

An example of the "university" ontology is given in Figure 1. Each of the boxes represents a class within the ontology, and lines represent the *subclass* relation. Each class also has attributes (slots), some of which define relations to other entities in the ontology. Thus, a web page which is an instance of the Faculty class would have the attributes *Projects.Led.By*, *Students.Of*, *Department.Of*, *Projects.Of*, *Courses.Taught.By*, *Name*, and *Home.Page*.

With this ontology in hand, we would like to be able to train our classifier to recognize which pages belong to which classes within the ontology. Thus, after showing our classifier a number of web pages of each class, we want to be able to present our classifier with a web page that it has never seen before, and it should be able to tell us which class the page belongs to with some level of accuracy. Alternatively, we could specify the level of accuracy we require, and present the classifier with N pages. The classifier would then choose to classify a page or not, depending on how confident it is of maintaining our set level of accuracy across the N pages.

The space of possible classifiers is extensive, but can be partitioned along three axes:

1. those using information on the WWW page itself,
2. those relying on the hyperlinks between pages, and
3. those that examine meta information about the page, such as the URL (Uniform Resource Locator) of the page in question.

Along the first axis, we find classifiers such as the familiar flat text classifiers, and classifiers that use the HTML tags to define a richer feature set (such as the Title/Headings Classifier in 5.1). Along the second axis, we try to classify a hyperlinked page P by examining the hyperlinks to its neighbors, where Q is a neighbor of P if P is connected to Q by hyperlinks, or vice versa. Clearly, it is also possible to find classifiers in

this space that do not fall only on one of the axes, but combine information from separate sources to determine the class of a page. Along the third axis, classifiers may use information such as the URL (Uniform Resource Locator) of the page in question to try to identify its class. For example, a common feature of URLs to home pages of people is the presence of the words "*usr*" or "*user*".

3 The Approach

We assume that there is a bijection between a web page and an entity in the ontology. For example, an instance of the *Student* class corresponds to exactly one web page. All other pages that may belong to this student would be treated as *Other* pages. In addition, we also assume that some of the hyperlinks between web pages correspond to relationships in the ontology. Thus, a hyperlink from a *Student* page to a *Faculty* page might correspond to an *Advisor.Of* relation.

Three major approaches to solve the problem of classifying web pages were investigated. Since we want to be able to compare the performance of each of these methods, a standard data set was used to evaluate each method based on the fixed university ontology in Figure 1.

We first describe the conditions under which the experiments were carried out, the data used, and how the experiments were executed.

We then present a method for text classification which only uses the words on a Web page as features. The performance of this classifier forms a basis for comparison with methods that leverage the richer features provided by the Web. Using the insights gathered from experiments with this method, we proceed to discuss each of the new approaches in turn, and the corresponding experimental results.

3.1 Experimental Testbed

Web pages were collected from four universities: Cornell University, University of Texas, University of Washington, and University of Wisconsin.

To simplify data collection, only the web pages from the computer science departments at each of the universities were collected and classified into one of the leaf classes in the ontology. "Index" pages were used to collect and classify the pages quickly. (An example of an "index" page would be one listing all the faculty members of the computer science department at a university.) There were a total of 4,216 pages and 12,296 hyperlinks interconnecting these pages.

In addition, another set of web pages was collected by visiting pages which were:

- pointed to by one or more hyperlinks in the first set of pages,
- located within the same university, and
- not already in the first set of pages.

There were 3078 such pages and they formed the instances for the *Other* class in the ontology.

3.2 Experimental Procedure

For each of the techniques described, four experiments were performed by successively excluding data from one of the universities, training on data from the remaining three, then testing on the excluded university. The results from these four separate experiments were then averaged together.

3.3 Performance Metrics

For each classifier, the confusion matrix is presented. True classifications are listed across the top of each table, and the classifications given by a particular classifier are listed vertically in the first column. The last row labeled as "%" is calculated as:

$$\% = \frac{N_C(C)}{N_C}$$

where $N_C(C)$ is the number of documents belonging to class C that were classified as class C , and N_C is the total number of documents belonging to class C.

For the precision-recall graphs, classifiers were required to generate confidence scores across all classes for every document. The confidence scores for each class were then sorted in decreasing order. This list of N_{all} scores was then divided into ten equal-sized "bins", each covering an equal interval along the recall axis. As we scan from high to low scores, let the number of predictions we have seen so far be N_n.

Recall is then defined as:

$$recall = \frac{N_n}{N_{all}}$$

For the *ith* bin, precision is calculated as:

$$precision = \frac{N_i(C)}{N_i}$$

where $N_i(C)$ is the number of documents classified correctly as class C in bin *i*, and N_i is the total number of documents in bin *i*.

4 Unstructured Text Classifier

The Naïve Bayes approach [1] to classifying text has been used successfully to classify plain text. This method is based on Bayes' Theorem, which states that:

$$P(c_j|D) = \frac{P(D|c_j)P(c_j)}{P(D)}$$

where c_j is a possible class in the set of all possible classes C, and D is the document to be classified.

This method has the underlying assumption that the probability of a word occurring in a document given the document class c_j is independent of the probability of all other words occurring in that document given the same document class:

$$P(w_1,\ldots,w_n|c_j) = \prod_i P(w_i|c_j)$$

where $(w_1,\ldots,w_n)$ = D. Thus, this classifier picks the best hypothesis c^* given by:

$$c^* = \arg\max_{c_j \in C} P(c_j|w_1,w_2,\ldots,w_n) = \arg\max_{c_j \in C} P(c_j)\prod_{i=1}^{n} P(w_i|c_j)$$

Notice that this method does not make any references to features of web pages other than the words that are actually on that page. We define the set of possible classes C as *Department*, *Faculty*, *Staff*, *Student*, *Research.Project*, *Course* and *Other*. The choice to aggregate *Post.Doc* and *Research.Associate* to *Staff* was necessary due to the shortage of training examples for each of these classes. The results from using this classifier are shown in Figure 2.

True / Classified as	Course	Student	Faculty	Dept	Staff	Proj	Other
Course	**199**	15	0	0	0	1	898
Student	16	504	22	0	26	13	961
Faculty	15	32	128	0	17	31	432
Department	7	2	3	**4**	1	7	321
Staff	0	0	0	0	**0**	0	14
Research Project	1	1	0	0	2	32	271
Other	0	1	0	0	0	0	183
%	84	91	84	100	0	38	6

Figure 2. Experimental Results Using Naive Bayes Classifier

5 Structured Text Classifiers

With the results from the Naïve Bayes classifier as a baseline, we seek to improve performance by leveraging the structural information in a web page.

5.1 Title/Headings Classifier (TH)

The Title/Headings Classifier (TH) uses only the words in a web page marked as part of the title or as headings to classify that page. Thus, words in between <*title*> and </*title*>, and words between <*hn*> and </*hn*>, where n is an integer, were taken as representative of a page. The Naïve Bayes classifier was then used to train and classify pages based on these words.

The rationale for choosing to use only the words in the titles and headings was based on the assumption that people would title a web page with words indicative of the contents.

Moreover, words in headings usually summarize the information in that section, and thus should prove useful in identifying the contents of the page.

The experimental results for this method are presented in Figure 3.

True / Classified as	Course	Student	Faculty	Dept	Staff	Proj	Other
Course	**192**	39	17	1	2	8	1148
Student	29	**396**	40	0	21	12	428
Faculty	1	53	**82**	0	19	7	224
Department	3	25	6	**2**	1	4	276
Staff	0	4	3	0	**0**	0	10
Research Project	4	9	1	0	1	**45**	348
Other	1	6	0	0	0	6	**418**
%	83	74	55	67	0	55	15

Figure 3. Experimental Results Using Title/Headings (TH) Classifier

These results show that using titles and headings as an abstraction of the underlying web page still captures significant information about the contents of the page. The TH classifier performs remarkably well compared to the Naïve Bayes classifier in some categories, such as *Course* and *Other*, especially considering that it uses, on average, less than 10% of the text. However, using fewer words unfortunately also imparts to the TH classifier a greater variance in accuracy. Performance in the *Faculty* and *Student* categories, which are more easily confused even by humans, was considerably worse than the Naïve Bayes classifier. Nevertheless, this experiment shows that abstracting a web page by considering the semantics of the structure of the page, such as titles and headings, benefits a classifier by requiring less text.

5.2 Hyperlinked Text (HT) Classifier

In addition to using the structure of a single web page, we can also leverage the hyperlinked structure of a collection of web pages. The Hyperlinked Text classifier uses all the words which are underlined in all hyperlinks to a web page to predict what class that web page belongs to. For example, there is a hyperlink in my home page (an instance of the *Student* class) to Tom's home page (an instance of the *Faculty* class). The hyperlink contains the words "Senior Honors Thesis Advisor", which are underlined when rendered by most WWW browsers. There is also a hyperlink in the WebKB Project page (an instance of the *Research.Project* class) to Tom's home page. The underlined words in this hyperlink might be "Principal Faculty Investigator". The HT classifier collects all these hyperlinks to Tom's home page and uses the concatenation of all the underlined words as a representation for that page. The Naïve Bayes approach is then used to train and classify these representations.

The rationale for this experiment was based on the assumption that creators of Web pages would underline words which describe the page that a hyperlink points to. If we gather all the underlined words of all the hyperlinks to a particular page, then we should have a fairly descriptive representation of the page to allow classification.

Experimental results using this approach are presented in Figure 4.

True / Classified as	Course	Student	Faculty	Dept	Staff	Proj	Other
Course	**253**	29	16	1	2	5	90
Student	1	**182**	31		11		33
Faculty	0	26	**32**		8		6
Department	0	2	2	**1**	1		15
Staff	1	2	5				2
Research Project	0	0	0			**8**	22
Other	0	0	60	2	21	71	**2793**
%	99	76	22	25	0	10	94

Figure 4. Experimental Results Using Hyperlinked Text (HT) Classifier

The HT Classifier performs remarkably well for the *Course* and *Other* classes, even outperforming the NB Classifier. However, performance for the remaining classes are poor. To see why, here are the top ten words, sorted by information gain, from the training set without Cornell University:

```
0.07111 cs
0.06414 section
0.04192 cse
0.02458 prof
0.02357 home
0.02297 group
0.01972 page
0.01604 computer
0.01517 back
0.01430 project
```

We see that commonly underlined words in hyperlinks such as “home”, “back”, and “page” are confusing the classifier. These words are not descriptive of the actual contents of the page behind the hyperlink, but rather, are navigational aids. This runs counter to our assumption, and causes this classifier to produce less than satisfying results.

6 Combining Classifiers

Given the performance of these three separate classifiers, can we improve classification by combining these classifiers in some way? In this section, we explore two ways of combining classifiers: taking the maximum of scores, and voting.

However, in order to reason about scores across classifiers, we need some way of comparing the confidence scores produced by each classifier. Currently, confidence scores cannot be compared directly since each classifier normalizes the scores. What we can do is to convert the scores to an estimate of true accuracy via a “calibration curve” – a graph of true accuracy versus confidence score. We can easily generate such a graph by calculating the accuracy[1] over different intervals of confidence over the training data[2].

[1] Accuracy was calculated as follows: Each classifier was forced to generate confidence scores for every class. Scores for each class were then sorted into overlapping intervals, each having 10% of the number of documents and exactly 2 documents different – the most and least confident within the interval. The accuracy for each interval is the percentage of documents correctly classified out of the number of documents in that interval.

When we use the calibration curve, scores that fall between our data points are interpolated linearly to obtain an estimate. An example of a calibration curve is the one for the *Student* class used to calibrate the NB classifier in Figure 5 below.

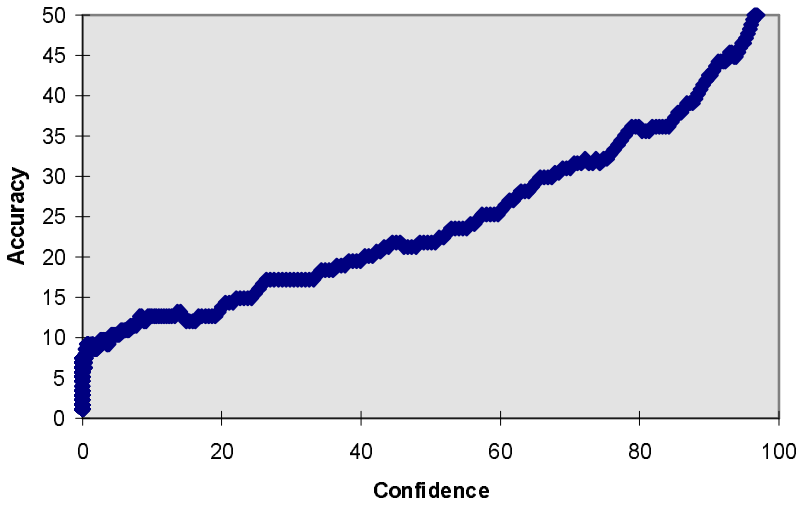

Figure 5. Calibration Curve for Student Class of Naïve Bayes Classifier

Armed with three different sets of calibration curves, one for each classifier, we can proceed to experiment with ways of combining classifiers.

6.1 Bigger is Better

The first approach is to simply take the maximum of the calibrated accuracy estimates for each classifier, and pick the maximum one as the classification. Thus, if Classifier 1 classified a document as a Student page with calibrated accuracy of 0.9, but Classifier 2 classified the same document as a Course page with calibrated accuracy of 0.7, then we decide that the page is a Student page with confidence 0.9.

The rationale for this is that we should be able to get better performance if we always pick the classification of the most confident classifier.

Experimental results for combining the NB and HT classifiers in the way is shown in Figure 6.

[2] Using training data to generate the calibration curve yields a biased estimate of the true accuracy, and therefore the curve will only be an approximation to the true calibration curve. We chose this method over a train/calibrate split of the training data in order to maximize the number of training examples.

True / Classified as	Course	Student	Faculty	Dept	Staff	Proj	Other
Course	**33**	1	0	0	0	0	35
Student	0	**88**	5	0	10	0	43
Faculty	0	4	**11**	0	4	0	8
Department	0	0	0	**0**	0	0	2
Staff	0	0	0	0	**0**	0	1
Research Project	0	0	0	0	0	**0**	3
Other	11	35	18	1	7	20	**527**
%	75	69	32	0	0	0	85

Figure 6. Experimental Results of Max Classifier

Why does this classifier not outperform either the NB or HT classifiers? Graph 1 presents a precision-recall graph of the *Course* class. We would expect that the Max Classifier would follow the higher of the other two lines across the entire range of recall, but this is not so.

It is the case that the average length of a document used in the HT Classifier is much less than the those used in the NB Classifier, since there tend to be fewer underlined words of hyperlinks than the actual text of a Web page. Thus, the Naïve Bayes formula given in Section 4 will always give larger scores for shorter documents since fewer terms are multiplied together. Thus, the next experiment to try is to scale scores by the document length. The obvious choice is some form of averaging scheme to find the "average probability per word". The geometric mean was chosen, and all scores were first scaled by the following formula:

$$ScaledScore = \sqrt[N]{OriginalScore}$$

where N is the number of words in the document. Scaled scores were then subjected to the same calibration process as described above, and the Max Classifier applied. The precision-recall graph for this ScaledMax Classifier is presented in Graph 2. There is some improvement around the 50% recall point, but still, taking the maximum of scores falls short of expected behavior. Note that the classifier does indeed perform as expected for low recall rates, but as we progress towards higher recall rates, the rate of errors increases, but we are still taking the maximum scores. Thus, at high recall rates, the classifier believes that it is correct with excessively high confidence.

6.2 Majority Rules

The second approach is to take a vote among the three classifiers that we have, and pick the class voted by the majority as the classification. However, if all three classifiers should disagree, then we simply fall back on taking the maximum, as in 6.1.

The rationale for this approach is simple. If more classifiers, using different features and data sources, believe that a page belongs to a particular class, then that class ought to be the correct one more often than not.

The experimental results is presented in Figure 7.

True / Classified as	Course	Student	Faculty	Dept	Staff	Proj	Other
Course	**25**	1	0	0	0	0	31
Student	0	**90**	4	0	6	1	47
Faculty	0	1	**7**	0	4	0	8
Department	0	0	0	**0**	0	0	2
Staff	0	0	0	0	**0**	0	1
Research Project	0	0	0	0	0	**0**	0
Other	19	36	23	1	11	19	**530**
%	57	70	21	0	0	0	86

Figure 7. Experimental Results of the Vote Classifier

The precision-recall of this classifier is presented in Graph 3. Although this classifier appears to have high confusion, at a given percentage of recall, it will outperform each of the voters.

7 Discussion

Some classes of the ontology clearly benefit from the use of richer features of the Web. The TH Classifier provides an almost equivalent level of performance compared to the flat-text NB Classifier.

Although we believe that there are potentially significant gains in combining different classifiers using different features, the benefits have as yet to be fully realized. We see glimpses of such a possibility, though, given the high precision-recall of the Vote Classifier. Other methods of combining classifiers (such as hierarchically, as outlined below) could potentially realize these gains.

There is much scope for future work.

- Hierarchical classification

  Given the hierarchical nature of the ontology, is there some way of leveraging this information to obtain better classification accuracy?

- Other ways of combing classifiers

  There are many ways that we can combine classifiers, for example, based on the entropy [3] of the data.

- Learning the URL-to-class function

  URLs contain descriptive information about certain classes of pages. Perhaps it could be combined with other classifiers which do not perform as well on these classes.

- Learning relations among classes

  We have yet to tap the relations that exist among classes in the ontology. If we knew that a page was a *Student* page, the probability that a hyperlink on it points to a *Faculty* page (the student's advisor, for example), is now higher than if one did not know this piece of information.

- Improving the Vote Classifier

  The current Vote Classifier has dismal confusion, despite its high precision at given recall levels. For example, it was only able to predict slightly over half of the 57 documents in the *Course* class correctly if we require that the classifier always output a classification for every document (Figure 7). However, if we allow classifiers to not classify a document if confidence is low (i.e., allow lower recall), we can achieve higher precision (Graph 3). Can we improve this situation?

- Other domains

  Do these results hold true for other domains? For example, if we define an ontology for Web pages of companies on the WWW, will these results remain true? What about other hyperlinked but non-Web domains, such as references in research papers?

8 References

1. *Machine Learning* by T. M Mitchell. McGraw-Hill Companies, Inc.

2. *Learning to Extract Symbolic Knowledge from the World Wide Web* by M. Craven, D. Freitag, A. McCallum, T. M. Mitchell, K. Nigam, C. Y. Quek. Internal Report, School of Computer Science, CMU, January 1997.

3. *A Maximum Entropy Approach to Adaptive Statistical Language Modeling* by R. Rosenfeld.