Fast webpage classification using URL features

Min-Yen Kan Hoang Oanh Nguyen Thi
Department of Computer Science, School of Computing
3 Science Drive 2, Singapore 117543
{kanmy,nguyent6}@comp.nus.edu.sg

ABSTRACT
We demonstrate the usefulness of the uniform resource locator (URL) alone in performing web page classification. This approach is magnitudes faster than typical web page classification, as the pages themselves do not have to be fetched and analyzed. Our approach segments the URL into meaningful chunks and adds component, sequential and orthographic features to model salient patterns. The resulting binary features are used in supervised maximum entropy modeling. We analyze our approach's effectiveness in binary, multi-class and hierarchical classification. Our results show that, in certain scenarios, URL-based methods approach and sometime exceeds the performance of full-text and link-based methods. We also use these features to predict the prestige of a webpage (as modeled by Pagerank), and show that it can be predicted with an average error of less than one point (on a ten-point scale) in a topical set of web pages.

Categories and Subject Descriptors
H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing – *linguistic processing*

General Terms
Algorithms, Experimentation.

Keywords
Uniform resource locator, word segmentation, maximum entropy, text categorization, webpage classification.

1. INTRODUCTION
Current webpage classification techniques use a variety of information to classify a target page: the text of the page itself, its hyperlink structure, the link structure and anchor text from pages pointing to the target page and its location (given by its URL). Of this information, a web page's uniform resource locator (URL) is the least expensive to obtain and one of the more informative sources with respect to classification.

Past systems have incorporated URL features into machine learning frameworks before, but to our knowledge, only two approaches have attempted to utilize this source of information beyond simple heuristics [9][15]. Surveys on web classification techniques, such as [18], have largely ignored this information.

URLs are often meant to be easily recalled by humans, and websites that follow good design techniques will encode useful words that describe their resource in the website's domain name as advocated by best practice guidelines ([11], pg. 26). Websites that present a large amount of expository information often break their contents into a hierarchy of pages on subtopics. This information structuring for the web often is mirrored in their URLs as well. As the URL is short, ubiquitous (all web pages, whether or not they are accessible or even exist, have URLs) and is largely content-bearing, it seems logical to expend more effort in making full use of this resource.

We approach this problem by considering a classifier that is restricted to using the URL as the sole source of input. Such a classifier is of interest as it would be magnitudes faster than traditional approaches as it does not require pages to be fetched or the full text or links to be analyzed. We have implemented such a classifier which uses a two-step machine learning approach. A URL is first segmented into meaningful tokens using information-theoretic measures. This is necessary as some components of a URL are not delimited by spaces (especially domain names). These tokens are then fed into an analysis module that derives useful composite features for classification. These features model sequential dependencies between tokens, their orthographic patterns, length, and originating URI component.

In the second step, machine learning is used to induce a multi-class or regression model from labeled training URLs that have been processed by the above pipeline. New, unseen test URLs can then be classified by processing them first to extract features, and then applying the derived model to obtain a final classification. A key result is that the combination of quality URL segmentation and feature extraction results in a significant improvement in classification accuracy over baseline approaches.

To assess the system's performance we have conducted a range of experiments that demonstrate its applicability in multi-class and hierarchical classification as well as relevance feedback, using standard datasets. In Sections 2 and 3, we discuss the two-stage approach to feature extraction. Following a short description of maximum entropy modeling, we describe the experiments and analyze the performance in Section 5.

2. RECURSIVE SEGMENTATION USING ENTROPY REDUCTION
Supervised machine learning requires examples of the form $f_1,f_2,...f_n$: C in which n features are distilled from a problem instance, and provided to the learner along with the class label C. In our system, tokens derived from the URL serve as binary features: for each token t_i present in a training URL, $f_i(u) = 1$ if t_i is present in the URL u and 0 otherwise, where i ranges from 1 to $|T|$, the number of unique tokens seen during training.

Thus, it is critical to select meaningful tokens to represent the URL. A simple and effective approach is to segment a given URL into its components as given by the URI protocol (e.g., *scheme :// host / path / document . extension ? query # fragment*). We can further break these components at non-alphanumeric characters and at URI-escaped entities (e.g., '%20') to create smaller tokens. Such a *baseline* segmentation is straightforward to implement and typically results in 4-7 tokens for subsequent classifier induction.

Table 1: URL feature classes and examples. Other tables reference these classes by their single letter key (e.g., Length → 'L').

Sample URL	http://audience.cnn.com/services/activatealert.jsp?source=cnn&id=203&value=hurricane+isabel

Feature Class (class tag)	**Example**
0. Baseline (B)	http audience cnn com services activatealert jsp source cnn id 203 value hurricane Isabel
1. Segments by Entropy Reduction (S)	http audience cnn com services activate alert jsp source cnn id 203 value hurricane isabel
2. URI Components (C)	scheme:http extHost:audience dn:cnn tld:com ABSENT:port path:services … ABSENT:fragment
3. Length (L)	chars:total:42 segs:total:8 chars:scheme:4 segs:scheme:1 chars:extHost:8 segs:extHost:1... segs:extn:1
4. Orthographic (O)	Numeric:3 Numeric:queryVal:3
5. Sequential Bi-, Tri, 4-grams (N)	com|cnn cnn|audience audience|services services|activate activate|alert alert|jsp com|cnn|audience cnn|audience|services ... services|activate|alert|jsp
6. Precedence Bigram (P)	com>services com>activate com>alert com>jsp cnn>services cnn>activate cnn>alert cnn>jsp ... activate>jsp

The first 2 rows of Table 1 show a sample URL and its baseline segmentation.

Notice that some portions of the URL contain concatenated words (e.g., *activatealert*). This is especially prevalent in website domain names. Segmenting these tokens into its component words is likely to increase performance as these sparsely occurring tokens can be traded for more frequent and informative ones. Word segmentation techniques -- for segmenting languages without delimiters (*e.g.*, Chinese) – are applicable to this task. [4] categorizes word segmentation approaches into four categories: 1) statistical methods, 2) dictionary based methods, 3) syntax-based methods and 4) conceptual methods.

Previously, Kan studied different methods for URL token segmentation [9]. One approach was segmentation by information content (entropy) reduction, in which a token T can be split into n partitions if the partitioning's entropy is lower than the token's:

$$H(t_1,t_2,...t_n) < H(T_{letters})$$

where t_i denotes the ith partition of T, and $T_{letters}$ denotes the partitioning in which each letter of the token is a separate segment. Intuitively, a partitioning that has lower entropy than others would be a more probable parse of the token. Such entropies can be estimated by collecting the frequencies of tokens in a large corpus. Fortunately, token statistics on the large 49M WebBase corpus are publicly available[1], from which we can calculate entropy estimates.

Finding the partitioning that results in minimal entropy requires exponential time, as all $2^{|T-1|}$ possible partitions are searched. The brute force algorithm used previously is inefficient for long strings. We propose a recursive solution that searches the space of possible partitions. A token is searched for a single partition point that lowers its entropy. The resulting tokens are then recursively searched using the same method. While this approach does not guarantee that the global minimum entropy is found, the algorithm has a much lower time complexity, *O(n log n)*, and matches the local minima in most cases. The resulting token segmentation example is given in row 1 in Table 1.

[1] http://elib.cs.berkeley.edu/docfreq

3. URL FEATURE CLASSES

In this stage, we want to further enrich these segment features with other useful ones. In [9] ambiguous tokens are expanded using correspondences found in the {*<title>*} HTML tag. For example, *md* could be expanded as *md*, *medical* or *moldova* (among others). According to the study, this technique did not affect classification much, as i) few (~6%) URLs have tokens that are ambiguous and ii) many ambiguous tokens' expansion act as evidence for the same class (e.g., *md* as *medical* or *meds* would both favor a *Health* subject classification).

In contrast, we explore classes of features that we feel are applicable to a broad range URLs and that are likely to have a positive impact on classification. We first discuss feature extraction for URI components, length and orthographic patterns, followed by sequential patterns.

3.1 URI Components and Length Features

A token that occurs in different parts of URLs may contribute differently to classification. Consider the token *ibm*, appearing in the domain name in one URL, and in the document name in another. In the first case, we know that the web page is located on IBM's web server, but could relate to any technical topic. In the second, the document is named "IBM" and is likely to discuss the company itself. Distinguishing where these tokens occur may improve classification accuracy.

This is easily modeled by creating features that indicate which URI component a token comes from. We augment the feature set by copying of all of the tokens from segmentation but qualifying them with their component origin (as shown in Row 2 of Table 1.

We add these features rather than replace the original ones, as the original features are more general and have higher frequency counts which combat sparse data problems.

The absence of certain components can influence classification as well. Advertisement URLs that underlie an advertising image banner often have a second URL embedded in the query arguments (*e.g.*, doubleclick URLs). Thus the absence of a query argument is partial evidence that a URL is not an advertisement link. We add a feature to the feature vector when a URI component is absent (also seen in the example in Row 2).

The length of the URL or its components may also influence classification. For example, departmental staff listings are usually not too deeply nested within an academic website. On the other hand, white papers or software drivers for products are usually very deeply nested. We thus add lengths of the URL and its components as features to the classifier, as shown in Row 3.

3.2 Orthographic Features

Using the surface form of a token also presents challenges for generalization. For example, tokens *2002* and *2003* are distinct tokens and have no correlation with each other in the framework proposed thus far. We borrow the notion of creating word features from related work in named entity recognition [2] and add a few orthographic features to our feature set, as shown in Row 4. We add features for tokens with capitalized letters and/or numbers that differentiate these tokens by their length. These features are added both in a general, URL-wide feature as well as ones that are URI component-specific.

3.3 Sequential Features

Shih and Karger's work on URL trees [15] demonstrated that URL token sequences can be effective for classification. In their work, a tree rooted at the leftmost token (usually *http*) is created, in which successive tokens (read left to right) are inserted as children. Each URL constitutes a path from the root to a leaf, and a tree structure emerges after a number of URLs are inserted. The intuition is that subtrees within the URL tree are presumed to have a similar classification.

While they need to classify URLs within a single website, our scenario involves many websites. As such, we cannot directly apply their approach to general URL classification, in which URLs from different websites need to be classified, as different websites appear as different "subtrees" in the URL tree. This is a difficulty as common subsequences of nodes cannot be generalized (such as recurring patterns in different websites).

We address this problem in our work by noting it is the sequential order of nodes in the URL tree that is of import, and not the rooted path. For example, seeing the token sequence *hurricane isabel* anywhere in the URL should lend evidence to certain categories. Sequential order among tokens also matters, as the token sequences *web spider* and *spider web* are likely to appear in URLs belonging to different classes.

Furthermore, consider the case of the token sequences *states georgia cities atlanta* and *georgia altanta*. Modeling sequences of tokens as features fails to capture the similarity between these sequences, as the intervening token *cities* forces the crucial precedence relationship to be missed. We can capture this by introducing features that model left-to-right precedence between tokens: *georgia>atlanta*.

An important note is that we also reverse the order the components within the server hostname. Hostnames are written specific to general (e.g., " server . domainname . tld . country ") whereas paths to files are written general to specific. This operation helps to preserve the precedence as general to specific throughout the URL.

Rows 5 and 6 in Table 1 show the features that are added by these two operations. Sequential dependencies are modeled by bigram, trigram, and 4-gram features, similar to context features used in [13] used in maximum entropy part-of-speech tagging.

4. CATEGORIAL CLASSIFICATION USING MAXIMUM ENTROPY

Maximum entropy (hereafter, ME) modeling has been successfully applied to text classification problems [12]. From a practitioner's point of view, its advantage is that it can handle a large set of features that are interdependent. Features (defined as "contexts" in the ME literature) are automatically weighted by an optimization process, resulting in a model that maximizes the conditional likelihood of the class labels *c*, given the training data *d*, or *P(c|d)*. The result is an exponential model of the form:

$$P(c \mid d) = \frac{1}{Z(d)} \exp(\sum_i \alpha_i f_i(d,c))$$

where each $f_i(d,c)$ is a feature and α_i is the weight to be determined through optimization. $Z(d)$ is a normalization factor used to force the conditional probabilities to sum to 1.

Maximum entropy is so named as it generates a unique probability model with the maximum degree of uncertainly that fits the constraints given by the data. The resulting distribution does not assume facts beyond the observed data. To do otherwise would be to assume information the learner does not possess. A detailed explanation of ME is beyond the scope of this paper; the interested reader is referred to [1].

Two methods have been proposed to optimize α weights: iterative scaling and, more recently, gradient ascent. Generalized Iterative Scaling (GIS) updates weights by scaling the existing parameters to increase the likelihood of observed features. Iterative scaling needs to calculate only the expected values Ef_i, instead of calculating the gradient of the log-likelihood function. We can substitute gradient ascent for this task, using limited memory variable metrics as advocated by Malouf [10]. We experiment with both GIS and gradient ascent methods, the latter using Limited-memory Broyden Fletcher Goldfarb Shannon (L-BFGS) minimization. Both have been implemented in a publicly-available ME distribution[2]. We carry out 30 iterations in our experiments, unless noted otherwise.

ME modeling can also suffer from overfitting. This problem is most noticeable when the features in the training data occur infrequently, resulting in sparse data. In these cases, weights

[2] Zhang Le, http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

derived by the ME model for these sparse features may not accurately reflect test data. To address this weakness, smoothing has been advocated using a Gaussian prior. We use a Gaussian prior (with $\sigma = 1$) to smooth our models in all of our experiments, unless noted otherwise.

5. EVALUATION

We hypothesize that the different classes of URL features have different levels of effectiveness, depending on the classification task at hand. To test this, we apply ME based classification of URLs to several different scenarios:

1. Binary Classification – In this case, we wish to separate two classes from each other. This problem recurs in web applications as relevance: is a web page is relevant or not? We assess URL classification in two scenarios: as a pseudo relevance feedback mechanism in TREC Web queries, and user-specific link recommendation in news web pages.
2. Multi-class Classification – We can easily construct a multi-class classifier by building *n* binary classifiers that discriminate each class separately. An advantage of using ME is that multi-class classification is handled directly. We report on the performance of URL classification on the standard WebKB corpus and in a related task of classification for focused crawling of scholarly publications.
3. Hierarchical Classification – Subject hierarchies allow internet users to quickly browse a list of topically-relevant web pages. Pages are nested in increasingly fine-grained categories. We experiment with hierarchical classification with a large subset of the Open Directory Project to gauge the effectiveness of URL features on this task.
4. Regression – Regression is needed when predicting a continuous quantity from input features. In web applications, the importance (or prestige) of a webpage may be modeled by a numeric attribute. One such attribute is Pagerank [3], which is usually given as a logarithmic quantity. We assess whether URL features can be used to predict a page's Pagerank. In addition to ME, we use linear regression as another form of learning to take advantage of the ordered structure of the classes.

We report the effectiveness of our full battery of URL features on the above tasks (all feature classes, as reported in rows 1-6 in Table 1, reported as 'All' in our experiments). Additionally, we performed component evaluations to find the efficacy of individual feature classes. We use the single letter tags in Table 1 to denote the feature classes throughout the remainder of the paper. When applicable, we report results from previous work and compare with equivalent feature sets from our work.

5.1 Relevance Feedback: TREC Web

The most ubiquitous problem in web page classification might be defined as *relevance*: is a given page relevant to a particular search query? Current search engines employ textual, link, anchor text, URL, and spatial features to rank web pages for user queries. Would the addition of our comprehensive URL features improve ranking algorithms?

An ideal experiment would be to perform IR experiments on a large, standard collection both using and omitting URL features. Due to \resource limitations, we were unable to conduct such an experiment. Instead, we assess the performance of URL-based classification in relevance feedback (RF). In classical relevance feedback, a user who executes a search for relevant documents on a query first marks relevant documents retrieved by the system. These marked documents are used to find other (hopefully relevant) documents, based on their similarity to the marked documents. While classic RF calculates similarity based on the textual features in the full text of a document, we can substitute the textual features for URL-based ones. The idea is that other relevant web pages possess similar tokens and token patterns in their URLs.

We use the standard dataset and queries used in the Text REtrieval Conference's main web task (TREC 9 and 10, [8]) to perform experiments. The relevance judgments for all 100 queries from both years' data were used. This dataset also provides a list of web pages (on average, about 1,400 pages) previously judged as relevant/non-relevant by human assessors for each query. Note that the documents on these lists are the only documents within the 1.6 M page dataset that were assessed by humans. This means we cannot properly assess an RF algorithm if it proposes a web page was not previously judged.

In light of this limitation, we modify the task to simulated relevance feedback. In this modified task, we ask if a system can distinguish relevant documents on a target input set of documents, given another set of labeled relevant and labeled irrelevant documents. This classification task is not equivalent to the original task, as it ignores the performance of the classifier on documents not on the target list, but may yield indicative results. We executed simulated RF using five-fold cross validation, maintaining an equal ratio of relevant and irrelevant documents in each set. Table 2 summarizes performance using GIS parameter estimation over all 100 queries in the dataset.

Table 2: Mean precision, recall and F_1 for simulated relevance feedback task. Results averaged over all 100 queries.

ME configuration	Avg. Precision	Avg. Recall	Avg. F_1
GIS, no smoothing	.646	.303	.413
GIS, smoothing	.784	.049	.095

5.1.1 Analysis

The results show that using URLs for simulated relevance feedback is promising. Both ME schemes emphasize precision at the cost of recall, which is more favorable in actual RF. We believe this may be due to the overwhelming skew of the assessed document list. While variance was high, an average query had about 60 positive examples but over 1,300 negative ones.

Performance varied greatly from query to query. With smoothing, ME suggests URLs only on a small percentage of the queries (as evident by the low recall (.049). We conjecture that smoothing may help to improve precision in queries where many documents are relevant, but is not effective in queries with few relevant

documents. As smoothing helps to accurately weight low frequency features, we hypothesize that it creates a classification bias towards the majority class.

Can we compare these results against actual IR systems in the TREC Web Task? Unfortunately, the answer is no. Our simulated RF task is considerably easier than the TREC main web task, as we have access to a partial list of relevant documents. In addition, in the simulated RF task, we only judge documents in a limited target list and not on the whole collection. A fairer comparison would be to compare the use of such URL features in the interactive task, where relevance feedback is normally assessed. Unfortunately, this comparison is not possible, as the TREC task does not currently use web documents.

In [8], we see that the best performing systems report a fairly low average precision of .20 and .22 (JustSystem, TREC 9 and FUB, TREC 10, respectively). Although further testing is needed to validate this hypothesis, we believe that the addition of comprehensive URL features can boost IR performance as our simulated RF task shows high precision in comparison to other state of the art systems.

5.2 News Web Page Link Recommendation

In link recommendation, the goal is to build a classifier to recommend useful links given a current webpage in a browser. Such links can be highlighted or placed in a "recommended pages" menu. In [15], such a recommendation dataset was created by asking 176 users to examine five news web pages and click on any hyperlinks to stories that they found interesting.

Using their dataset, we follow their experimental procedure where they perform five fold cross validation, using a leave-one-page-out policy (*e.g.*, train on the judgments for the first four pages, and test to see whether recommendations match for the fifth page). A separate classifier was made for each of the 176 users and tested, totaling 182,325 data points. Shih and Karger tested URL tree based features, sharing some characteristics with our n-gram features in Section 3.3. While they use a tree-based model of learning, we use standard ME, not optimized for tree features. In addition to their tree-based learner, Shih and Karger report results for using similar features in a standard SVM framework.

To extend their experiments, we employ our features in both our ME framework as well as in an SVM framework. For ME classification, we present only the best results using L-BFGS. Table 3 shows the results of the experiment, in which the classifier recommends the top 1, 3, 5, or 10 links on a page with the highest probability of similarity to user clicks on the training pages. The first four rows are reprinted from [15] for comparison.

Table 3: Number of correct recommendations across all 176 users across 5 pages in the recommendation dataset (2 classes, 182K data points). Bolded numbers indicate top performers.

Learner Configuration	Top 1	Top 3	Top 5	Top 10
Random (Lower Bound)	29	104	162	330
Perfect (Upper Bound)	857	2488	3899	6093
TL-URL	**385**	979	1388	2149
SVM-URL	308	839	1268	1953
SVM (S)	355	969	1482	2473
SVM (N+P)	350	956	1421	2365
SVM (All)	363	996	1456	2412
ME (S)	379	1063	1657	2767
ME (N+P)	360	1028	1573	2577
ME (All)	365	**1100**	**1682**	**2775**

5.2.1 Analysis

Shih and Karger's tree learning algorithm (row TL-URL) using their URL features performs best at recommending the single most probable link, but is outperformed on the top 3, 5 and 10 metrics. SVM-URL denotes Shih and Karger's results using SVM features, which performs most poorly among the classifiers shown here. Our contribution here is to show that better classification is possible by extracting more meaningful features from the URL data, and that these features outweigh the gains made by using a specialized learner. This is exemplified in all three rows of SVM results from our configurations, in which we use the same machine learner as past results but use our URL features. Gains of over 20% are achieved by using all URL features discussed.

In this configuration, ME also outperforms SVM classification on this dataset for the top 3, 5 and 10 recommendations. Note that the performance of feature set (N+P), which is our best approximation of Shih and Karger's URL tree features, does poorly compared to the basic URL segmentation (S) under both SVM and ME models. We take this as a sign that our features can improve classification performance, as our composite feature set improves over our implementation of previous work by up to 29% in the best case of top 10 recommendation task.

5.3 Multi-class Categorization: WebKB

We now turn our attention to multi-class scenario. The WebKB corpus is a standard dataset for such benchmarking. It consists of web pages collected from four universities, classified into seven categories. We employ a subset of the WebKB, containing 4,167 pages (the ILP 98 dataset [16]), in which each page is associated with its anchor words (text from the hyperlinks that point to the page). The task is identical to earlier published experiments: only the *student*, *faculty*, *course* and *project* categories pages were used, and cross-validation using *leave-one-university-out* was performed for evaluation. Previous work using the full text have employed support vector machines (SVM) [17], maximum entropy [12], and inductive logic programming [16].

Table 4: WebKB performance (4 classes, 4.1K data points). Past results re-printed on top half. 'NR' = not reported.

Past Learner Configurations [Cite]	Accuracy	Macro F_1
SVM w URL (Kan, [9])	*NR*	.338
SVM w Full Text (Sun *et al.* [17])	*NR*	.492
SVM w Anchor Text (also [17])	*NR*	.582

ME w Full Text (Nigam *et al.* [12])	92.08%	*NR*

URL configuration (ME using GIS)	Accuracy	Macro F_1
S	57.16%	.273
C	23.34%	.094
L	53.60%	.174
O	54.18%	.190
N	32.08%	.341
P	41.50%	.350
All	76.18%	.525
Full Text	78.39%	.603
Full Text + All	80.98%	.627

5.3.1 Analysis

Results using URL features are shown alongside past published results in Table 4. We give performance values for both instance accuracy as well as macro F_1, as both metrics have been used as performance measures in past work. The new URL features perform very well, boosting performance over URL previous work (configuration 'S') by over 30% in the best case, resulting in 76% accuracy. This is impressive as the full text of the words of the page achieve about 95% of the performance of full text methods. A small gain in classification performance results when our URL features are combined with full text, showing that URL features can help to improve even full-text approaches. These results are similar to results shown for anchor text [17].

From the results, the *n*-gram and precedence features seems to be the most helpful, this is likely due to the fact that many URLs in the WebKB share many tokens in their paths. Segmentation of tokens is not very effective here, likely because the corpus does not have many compound tokens (again, featured most prominently in long domain names).

Three caveats need to be explained. First, note that our experiment show a best performance of ~78% accuracy using full text in contrast with [12] which shows 92% accuracy. The difference is that we perform *leave-one-university-out* cross validation, which we feel reflects real-world situations better. Also, Kan and Sun *et al.* used a set of binary classifiers rather than a true multi-class classifier. This can skew F_1 results as the four classifiers can be separately optimized, leaving some instances unclassified (which can improve F_1). Finally, Sun *et al.* perform stemming and stop word removal for SVMs, which we omit; we simply employ each word as a separate feature as input for ME classification.

5.4 Focused Crawling: Scholarly Publications

One potential application of URL classification is in focused crawling. In focused crawling, a web crawler targets a specific type or genre of document. In the context graph model [5], context language models are built to model the content of pages within 1, 2 or more links away from the desired page type. During the crawl, all downloaded pages are classified by the context model. Pages classified to be within 1 link of the desired page type are thus processed before ones on pages classified to be within 2 links, and so on.

With URL classification, we can move classification earlier in the pipeline: from modeling the destination page to modeling the linking URL. Similar to the recommendation scenario earlier, we assess whether a URL on a page will lead us to a target page or a page within 1, 2, or more links of a target page.

In a scholarly publication crawler, we may want the crawler to follow a desired path: *department* ⇒ *staff listing* ⇒ *homepage* ⇒ *publications*. In crawling, the model is used to direct the crawler from the starting class to the eventual target. We can then treat these as a standard multi-class classification problem.

We collected a four-university dataset in December 2004 consisting of 2,577 web pages in five categories: the four above plus a *other* category. Note that since there are many faculty members in each university, that the class distribution in the dataset is heavily skewed towards the target class and the *other* category. Also note the multi-label property of the target class *publication* with *homepage*; many researchers list their publications as part of their homepage.

We conducted tests using the URL features as well as ones using the full text. Here we try full text as well, following [17]'s recommendation to stem the text. Results are shown in Table 5, using *leave-one-university-out* cross validation.

5.4.1 Analysis

As in the WebKB corpus, URL features are competitive with full text approaches. In this corpus however, the features do not interact well to produce optimal results when using all feature classes: component and orthographic features do better on their own. URL features are likely dependent on the dataset and task. Still, given the evidence, we recommend the topic-sensitive crawlers should incorporate a URL classification module. Such a module performs as well as full-text approaches, while decreasing bandwidth usage.

Table 5: Crawling dataset classification performance (5 classes, 2.5K data points, ME using GIS with 300 iterations)

Config.	Acc.	Mac. F_1
S	51.43%	.489
C	64.14%	**.542**
L	49.20%	.266
O	61.54%	.375
N	62.59%	.433
P	59.25%	.457
All	52.98%	.351
Full Text	**64.73%**	.490
Stemmed Text	62.62%	.461

5.5 Hierarchical Categorization: The Open Directory Project

The Open Directory Project (ODP) is a large, publicly available collection of web pages, similar to the Yahoo! and LookSmart directories, both of which have been previously studied [7][6]. It differs from other web directories in that it is entire maintained by volunteers and that the data is freely available for use. Although the ODP is constantly evolving, snapshots of the entire directory structure at specific points in time have been made

available. These publicly available snapshots are thus good datasets for testing hierarchical webpage classification. We use the snapshot dated 3 August 2004, which encompasses over 4.4 M URLs categorized into 17 first-level and 508 second-level categories. Similar to other classification schemes, it also exhibits a skewed category distribution: 1.1 M (25%) of the pages belong to the two most frequent second-level classes, *Regional/North_America* and *World/Deutsch*.

We use 100,000 randomly chosen ODP URLs to assemble a test corpus for our two-level, hierarchical experiments. In this set 360 of the 508 second-level categories were present. We benchmarked the ME classifier using the two-level labels as 360 distinct classes, trained on another set of 100,000 URLs. We then assessed the ME classifier using a sequential hierarchical approach. Using the training set, we trained a classifier for the top-level and 17 other classifiers (one for each top-level class). In testing, the top-level classifier is run first to determine which second-level classifier is run to produce the final class results. Table 6 shows the results, showing both accuracy and macro averaged F_1 for both the flat and sequential hierarchical approach. Statistics for the top-level classifier alone are also shown.

5.5.1 Analysis

Our results validate earlier findings [6] which show that leveraging the hierarchical structure of the classes improves performance: in our case, ~10% gain in test instance accuracy and 50% in macro average F_1. How do URL features compare with earlier studies using text features? Dumais and Chen report a micro average F_1 over the 150 second-level categories in their LookSmart dataset of 0.476. The ODP task has over twice as many classes (360), and macro F_1 is a more difficult metric to score well on for biased datasets(as ME trains to increase instance not per-class accuracy). Yet, URL features do remarkably well: for the sequential ME (All) setup, we calculated micro averaged F_1 and found it to be 0.551. Unfortunately, the categories and their cardinalities differ across hierarchical classification reports, thus the results are not directly comparable.

Table 6: ODP classification performance (training and testing, each ≤100K data points). Best results bolded.

Learner Config.	Two level (360 classes)				Top-level only (17 classes)	
	Flat		Sequential			
	Acc.	Mac F_1	Acc.	Mac F_1	Acc.	Mac F_1
Majority Baseline	0.165	0.001	0.165	0.001	0.295	0.026
S	0.458	0.055	0.503	0.122	**0.577**	0.334
C	0.419	0.043	0.480	0.080	0.550	0.245
L	0.332	0.020	0.314	0.016	0.492	0.148
O	Many URLs without features, no results					
N	0.396	0.101	0.404	**0.151**	0.471	**0.368**
P	0.443	0.089	0.411	0.136	0.487	0.360
All	0.157	0.002	**0.504**	0.110	0.574	0.324

A second observation is that the URL features introduced in this paper seem to have little positive effect. Indeed, using the basic URL segmentation outperforms the full set in macro-averaged F_1 and is comparable in accuracy. An inspection reveals that ODP URLs tend to be domain names, often without path or document information (63% or 71%, of the test URLs respectively), and would not benefit from many of the newly introduced features. URL segmentation is most helpful to classify these URLs.

5.6 Predicting Pagerank

Our final experiments examine the prediction of Pagerank given our URL features. Pagerank models the prestige of a node in a directed graph by iterative refinement [3]. Instead of calculating Pageranks directly from a web graph, we use the scores returned by Google's Internet Explorer toolbar; for the experiments reported here, the Pageranks scores were captured during the period of December 2004 to January 2005. The exact method to compute Pagerank as reported by Google is perhaps not known, but it is likely a normalized, logarithmically-scaled version of raw Pagerank scores. The returned scores then consist of twelve nominal categories: integer scores 0 through 10, and an undefined class which is returned by Google if the Pagerank score has not been calculated. We dispense with URLs with undefined Pagerank in our experiments.

We can treat the Pagerank prediction problem as a simple multi-class problem when using ME. By doing so, we lose the continuous structure of categories: ranking a page as class 0 or class 10 is "equally" incorrect when it belongs to class 1. We can leverage this structure by using regression to predict a continuous value rather than a categorical one. We carried out an experiment on the focused crawling dataset used in Section 5.4. We perform experiments over the 2,043 URLs that had a defined Pagerank value, again using ten-fold cross validation.

To take advantage of the ordered structure of the classes, we use linear regression. Linear regression (LR) fits a linear, weighted combination of features to the training data so as to minimize the square of the error. However, regression grows linearly in complexity in proportion to the number of features used. This means we could not use all of the features that ME uses (over 34,000 in the 'All' feature class). We perform simple feature selection to rank the features by their frequency and power to affect the score. Formally, a feature is scored by:

$$\left| pr_f - pr \right| * \log(\, freq(f))$$

where pr denotes the mean Pagerank in the dataset, pr_f the mean Pagerank for pages containing feature f, and $freq(f)$, the frequency of f in the dataset. We used 100 features selected by the process above. Table 7 shows the error for the resulting learners.

Table 7: Pagerank prediction on crawling dataset (10 classes, 2.0K datapoints). Best performer in bold.

Learner Configuration	Mean Absolute Error	Root Mean Squared Error
Majority Baseline	1.107	1.535
ME (S)	0.768	1.426

ME (C)	0.720	1.348
ME (L)	0.849	1.467
ME (O)	Many URLs without features, no results	
ME (N)	0.775	1.441
ME (P)	0.930	1.661
ME (All)	**0.670**	**1.277**
LR (L, |f|=100)	0.892	1.303
LR (C, |f|=100)	0.842	1.245
LR (All, |f|=100)	0.905	1.332

5.6.1 Analysis

The result show that the ME configurations do quite well; the 'All' configuration drops absolute and RMS error by 39% and 16%, respectively, against the baseline. Length and component feature classes also do well, as both features seem to generalize across sites well. We used this as evidence to conduct linear regression on the same classes to retrieve features from the length, component and 'All' feature classes. However, ME is still competitive with regression. Interestingly, regression with more feature classes ('All') proved worse than with less features (*e.g.*, URL component features alone 'C'). We believe this is caused by the proportion of features used. In the length dataset, 100 features represents over 55% of all possible features, in the 'All' set it is less than 1%. This means more advanced feature selection (perhaps by using singular value decomposition) may be able to take advantage of the larger feature classes, and allow regression to surpass the performance of the unstructured ME model.

We also performed the same experiment on a random sample of 27,252 URLs from the ODP corpus. Here, results were much less encouraging as ME and linear regression models were ineffective at beating the majority baseline performance of 31% accuracy. The linear regression models in the crawling dataset showed that highly weighted features included institution names (e.g., $w_{cornell} = +.69$). Domain names that only appear once in the dataset make such features unobtainable, despite URL segmentation. Here, unlike in the ODP hierarchical evaluation, basic URL features did not work well. We believe this is because URL segments may often correlate with subject (e.g., "frisbee" → sport), but not with prestige ("frisbee" → ? Pagerank).

6. EFFICIENCY

URL based categorization is extremely efficient both in time and space, as the data examined is small in comparison to other approaches. The URL is also often semantically correlated with important classification problems, which we have empirically assessed through ME classification in the previous evaluations.

As there is no network transaction incurred, classification speed is limited solely by the processing power and memory of computer and the ME model. On an Intel P4 2.8 GHz with 1 GB of main memory, the ME training for a single model based on 100,000 instance URLs with all features takes about 20 minutes to train. Prediction is considerably faster, although a large part of the time cost is loading the model into memory. In a production system, the classification module would be run as a server to enhance performance. For our desktop machine, we estimate a throughput of about 1,000 URLs per second, making real-time web page classification possible.

7. CONCLUSIONS AND FUTURE WORK

Given that the URL is a ubiquitous feature of web pages, we study how they can be maximally leveraged for classification tasks. We have extended previous work and added features to model URL component length, content, orthography, token sequence and precedence. We evaluate the use of these features over a large set of tasks including relevance, categorization and Pagerank prediction. Results indicate URL features perform well on classification tasks, on par with or exceeding full-text and anchor text approaches in certain cases. Our method also outperforms earlier results using URL features that employed specialized learning algorithms; in contrast, we employ generic maximum entropy modeling as the supervised machine learning framework. We show that URL features also correlate with Pagerank in our topical collection, allowing prediction of Pagerank within 1 point on average on Google's 10-point scale.

Although using all feature classes introduced in this paper does well in most cases, our analysis indicates certain feature classes correlate better to some tasks than others. Many of our newly introduced features perform well on long URLs, typically found in an intranet setting. These features do not perform as well with typical web site entry points (*i.e.*, just the domain name), as they attempt to leverage the internal path structure of the URL. Future work needs to be done to further improve these features and to explore their correlations to find optimal sets for specific tasks.

Classification by URL features has other advantages aside from real-time efficiency: all web pages have URLs, regardless of whether they exist, are accessible, have incoming links or have any text (some web pages are comprised solely of image maps). On the other extreme, many pages have too many words that contribute noise. Employing text summarization to web pages has already shown to help in this process [14], and is a promising avenue for future work. Aside from its intrinsic performance, URL based classification can additionally be used to overcome the shortcomings of other methods that rely on these data.

8. REFERENCES

[1] A. L. Berger, S. D. Pietra, and V. J. D. Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39-71, 1996.

[2] D. M. Bikel, S. Miller, R. Schwartz, and R. Weischedel. Nymble: A high-performance learning name-finder. In *Proc. of ANLP '97*, pages 195-201, 1997.

[3] S. Brin and L. Page The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 1998.

[4] Y. Dai, C. S. G. Khoo, and T. E. Loh.A new statistical formula for Chinese text segmentation incorporating contextual information. In *Proc. of SIGIR '99*, 1999.

[5] M. Diligenti, F. M. Coetzee, S. Lawrence, C. L. Giles and M. Gori Focused Crawling Using Context Graphs, In *Proc. of VLDB '00*, 2000.

[6] S. Dumais and H. Chen. Hierarchical classification of web content. In *Proc. of SIGIR '00*, 2000.

[7] E. J. Glover, K. Tsioutsiouliklis, S. Lawrence, D. M. Pennock and G. W. Flake. Using web structure for classifying and describing web pages. In *Proc. of WWW '02*, 2002.

[8] D. Hawking and N. Craswell. Overview of the TREC-2001 web track. In *The Tenth Text REtrieval Conference (TREC 2001)*. NIST, 2001. Special Publication 500-250.

[9] M.-Y. Kan. Web page classification without the web page. In *Proc. of WWW '04*, 2004. Poster paper.

[10] R. Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proc. of 6th Conf. on Natural Language Learning*, pages 49-55, 2002.

[11] J. Nielsen and M. Tahir. *Homepage usability: 50 websites deconstructed*. New Riders Publishing, USA, 2001.

[12] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, 1999.

[13] A. Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution.* PhD thesis, Univ. of Pennsylvania, 1998.

[14] D. Shen, Z. Chen, Q. Yang, H.-J. Zeng, B. Zhang, Y. Lu and W.-Y. Ma. Web-page Classification through Summarization. In *Proc. of SIGIR '04*, 2004.

[15] L. K. Shih and D. Karger. Using URLs and table layout for web classification tasks. In *Proc. of WWW '04*, 2004.

[16] S. Slattery and M. Craven. Combining statistical and relational methods for learning in hypertext domains. In *8th Int'l Conf. on Inductive Logic Programming*, 1998.

[17] A. Sun, E.-P. Lim, and W.-K. Ng. Web classification using support vector machine. In *4th Int'l Workshop on Web Information and Data Management (WIDM 2002)*, Virginia, USA, November 2002.

[18] Y. Yang, S.Slattery, and R. Ghani. A study of approaches to hypertext categorization. *J. of Intelligent Information Systems*, 18(2-3):219-241, 2002.