

Integration of a Community-Based WWW Filter as a Capstone Research Experience for Undergraduate Computer Science Students

Peter J. DePasquale and Jason M. Snyder
The College of New Jersey
2000 Pennington Road
Ewing, NJ 08057
{depasqua, snyder22}@tcnj.edu

Abstract

The College's of New Jersey's Computer Science department offers capstone mentored research courses to qualifying juniors and seniors. Students can typically enroll in up to two such courses, offering the opportunity to work in small groups or a one-on-one basis with a full-time faculty member, typically through the incorporation a faculty member's research interest area or an ongoing project. These mentored research courses seek to expose the students to a more rigorous exploration of problem solving through individual research. The typical student in a mentored research course is generally one that is intending to or considering pursuing graduate studies.

This paper examines the communiFilter project that has been offered as a mentored research course since the fall 2004 semester. The authors describe the project's goals, design, and implementation - a community-based World Wide Web content filter. Additionally, the paper details the implementation of such a project through the mentored research offerings.

Keywords: filtering, content, WWW, capstone, undergraduate research

1. Introduction

The application and research of text classification approaches has long been of interest to the computer science community. This interest level continues [3, 4, 8] for a number of reasons, including its application toward combating unsolicited commercial

emails (spam). Numerous approaches have been developed and studied to address the spam problem [1, 5, 12]. Yet the use of Bayesian filtering techniques [7, 9] has risen to be one of the most effective and noted approaches, particularly following the publication of Paul Graham's landmark essay *A Plan for Spam* [6].

While these approaches have proven to be popular and effective solutions to the spam email problem, the need for effective World Wide Web (WWW) content filtering continues to be in demand. Entities such as corporations, schools, libraries, and home users with children often find the need to filter "inappropriate" WWW content. However, the definition of "inappropriate" can vary widely between these settings. For the corporation, filtering adult-content, sports, and on-line gambling content may be highly desired. In the library setting however, sporting news may be considered suitable while gambling and adult-content web sites may not.

In 1996, Tim Berners-Lee outlined [2] a need for communities to define their own ethical and moral standards and apply them to the problem of web filtering. [2] argued that the then-current software solutions for web content filtering included commercial off-the-shelf (COTS) solutions programmed by a team of anonymous individuals with whom the purchaser was unfamiliar. Particularly in the case of the typical home user seeking to block objectionable content from their children, home users of the COTS solution would have no knowledge of the software developer's moral and ethical values. These values would be the basis of the filter, yet the end-users would almost certainly

have differing standards that form the basis of what content they deem “inappropriate”.

Berners-Lee proposed a solution that involved the use of local communities (church groups, bowling leagues, parent-teacher organizations) whose combined ethics and moral standards were likely to be closely aligned with each of its members. Acting as a group, the community could establish the content types deemed to be objectionable, and through their actions define a filter for use in each member’s home. In essence, the community would define and utilize a filter common to their collective standards.

With this design in mind, and following the success of the application of Bayesian inference [11] approaches to classify (and therefore filter) “spam” email, we set out to create a community-based World Wide Web content filtering system – the *communiFilter*.

When combined with a community-oriented system of rating web pages based on their content, an effective system can be implemented to successfully filter content based on a community’s moral and ethical standards. This is an improved solution compared to filtering content based on the moral and ethical standards of a third party such as a software development company. The use of Bayesian statistics enables the filter to successfully deal with the dynamic conditions inherent in the Internet.

1.1. The Bayesian Approach to Solving the Spam Problem

In [6], the author presents a design for an email spam/content filter that uses Bayesian statistics to determine the likelihood of a given email being spam. The system uses a corpus of both “good” and “bad” emails. The corpus is parsed, creating a database of “good” and “bad” tokens and the number of occurrences of each token.

When a new email is received, it is parsed from the email message and its tokens are compared to the database to compute the probability of the token being identified as either “good” or “bad”. The 15 tokens with a probability furthest from .5, either above or below, are then used to compute the probability that the email is spam. If the probability is above a certain threshold, the email is blocked or marked as a spam message, otherwise it is untouched.

2. Applying the Bayesian Approach to WWW Content Filtering

2.1. The Community Interface

Building a filter based on a community’s standards obviously requires input from the community itself. To this end, a web interface has been created using PHP and MySQL which supports the community process of building a content filter. The interface allows ordinary members of the community (users) to simultaneously visit and rate web with their usual web browser. When visiting in a page, the users will mark the current page as either “good” or “bad”. This simultaneous viewing/rating capability is driven by the interface’s publication of bookmarklets – JavaScript coded that can be accessed via a bookmark in a web browser. By using the bookmarklets, users can browse to a web page, click a bookmark button to quickly rate the current page, submit the page to the administration system and continue browsing the web.

Elevated members of the community (moderators) then view the users’ submissions and confirm their submission. Once confirmed, submissions are generally classified by the moderators with content “labels” which categorize each submission. Examples of current content labels include “adult content”, “sports”, and “gambling”. These labels facilitate the exporting of the actual filter (described below) by supporting mass inclusion of label into the filter. Following categorization, each submitted page’s HTML content is parsed (including page metadata, image metadata, and other non-viewable textual content data) into word tokens which is then in turn stored for use when the filter is built.

2.2. Building a Filter

A content filter is defined by two hash tables, one “good” and one “bad”, each of which contain a list of tokens and the number of occurrences of each token in the hash table. Note that many common words, such as “the”, “and”, or “a” will occur in both halves of the list with a high degree of frequency. Tokens that only appear in one half of the list or that appear far more frequently in one half than in the other will therefore have vastly different occurrence rates.

Typically a *communiFilter* moderator will create a filter from a selection of categories, designating each category (and therefore the pages that comprise the category) into either the “good” or “bad” hash table. As the categories are added to the filter, the hash tables are updated to include all of the tokens

contained in each page of the category. When added to a hash table, if a given token is found to already be present, the token's count of the number of occurrences is merely increased. Thus, the hash table contains not only the list of all "good" tokens, but their frequency.

The two hash tables are then packaged in a plain text ASCII file for importing into the filtering software (described below). In this fashion, the administrative interface supports the creation of multiple filters from the same base corpus of parsed pages and tokens. Typical filter files currently run on the order of 300 kilobytes.

2.3. Using Apache as a Proxy

The actual filtering mechanism is implemented as an output filter module to an Apache 2 HTTP Server [10] as summarized in Figure 1. To utilize the content filter on inbound WWW traffic, an end-user configures their web browser to utilize the *communiFilter* WWW proxy server to handle their HTTP requests. As each request is made to the server, the content of the responding web page is obtained by the proxy server and passed to filter module prior to sending the results to the user.

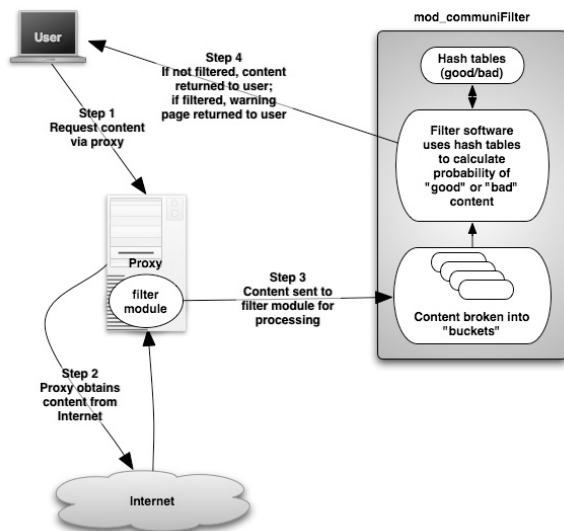


Figure 1 - The *communiFilter* processing model.

The filter module in turn parses the contents of the requested page into content tokens. Each token is compared to the "good" and "bad" hash tables to determine the frequency of the word in each table. These values are then integrated into a series of formulas which calculate a percentage for the token to be classified as "bad". For each token, this value is calculated, and the 15 most outlying percentages

(either "good" or "bad") are used in determining an overall "page probability" – the probability that the page is "bad". In the event a token is processed and can not be located in either the "good" or "bad" hash tables, it is assigned a value of .8 (80% likelihood of being "bad").

If the requested pages is determined to be classified as "bad", an informative message is returned to the user's browser and the page is effectively blocked. If the page is classified as "good", the page passes through the remainder of the proxy's output filters and is returned without modification to the user's browser.

This approach departs from Graham's implementation, whereas a page probability would need to surpass a defined page threshold (0.9) to be considered "bad". However, our testing has drawn the same conclusions as [6]; the page threshold value does not matter. Nearly every page tested was clearly "good" or "bad" (within .1 of the absolute "good" value of 0.0 or "bad" value of 1.0), and that a rare few pages are calculated to be otherwise.

2.4. Testing and Deployment

Following initial development of the administrative system and filter module, a testing system was developed to provide support for regression testing. This system has been designed and implemented to streamline and automate the process in order to gain quantitative data on the operation and efficacy of the filter.

The *communiFilter* testing system is configured to first download the HTML source for 100 different web pages. Of these pages, 50 are "bad" and 50 are "good". The downloaded HTML files are archived to support future testing across the same input set. We took this approach due to the dynamic nature of web content – dynamically fetching each page during the test could easily yield widely varied results with each execution of the test.

The testing system supports the use of a parameterized configuration file which allows the *communiFilter* team to apply a range of page cutoff threshold values as well as the unknown token value (the value given to tokens that do not appear in either hash table). Each page tested through the system is processed with the parameter values, resulting in a corpus of output information such as the number pages processed correctly, the number of good pages that were blocked, and the number of bad pages that

passed the filter. Initial results show that all good pages successfully pass through the filter and approximately one to two percent of bad pages pass when they should not.

At the present time, the testing system is undergoing further refinement to support database archiving of the resulting test output, additional configuration parameterization of the filter, and the collection of timing data to aid in future efforts to further streamline the filter module.

2.5. Current Effort: Going Beyond the Basic Bayesian Approach

With the completion of the initial implementation of the filtering system, one student member of the team has turned their effort to the improvement of the module's efficacy. As such, two hypotheses have been implemented to explore their effect upon the filter – the use of URL blacklists within in the filter, and the use of tokens comprised of pairs of existing adjoining tokens.

While web site (domain) blacklists seek to strictly limit known domains, they are very helpful when dealing with pages with little textual content from which it is difficult to extract tokens. A page that consists mostly of images and a sparse amount of textual content (as is the case with many adult-oriented web sites) is likely to pass our filter since there are few tokens that can be analyzed. Blacklisting obvious known “bad” domains will block any pages originating from those domains, regardless of content. Blacklisting also saves processing time since blacklisted pages do not have to be tokenized and processed through the content filter.

With respect to tokens comprised of pairs of existing tokens, Paul Graham noticed that specific combinations of two words in sequence were often a good indicator of spam, even if the individual words are benign by themselves. This same idea can also be applied to WWW content filtering. One such example is the phrase “after attempt”. The individual words, “after” and “attempt” are fairly common words that are not usually indicators of inappropriate content. However, when used together as a single token pair “*afterattempt*”, they are almost always part of the phrase “point after attempt” which is a nearly certain indicator of a page about American football.

Both blacklisting and token pairs have been integrated into the content filter on an initial basis. Further refinement will be the focus of a student's

contribution this semester. Additionally, the student will be using the newly developed testing system to establish the efficacy of the integration of token pairs into the content filter.

3. *communiFilter* as a Research Project for Undergraduates

3.1. Recruiting Students

Since its introduction as a project in the fall semester of 2003, six students have worked on the *communiFilter* project through the department's capstone mentored research courses. These offerings support individual and small-group research through projects offered by and under the direct supervision of department faculty. Typically mentored research projects are geared toward furthering the research agendas of the sponsoring faculty and require the student to delve into areas of computer science not otherwise experienced in their curriculum.

Initially the project was offered as two distinct investigations. The first was an exploration of building a Mozilla browser extension that facilitated the end user's filtering mechanism. Second, the initial creation of the filter-building web site – used to support the notion of the creation of community-centric filters.

As students rotated through the project over the intervening semesters (some students take choose to take only one mentored research experience, others opt for as many as three) the project evolved into its current form – implementing the filtering mechanism as an Apache output filter module embedded as part of a proxy web server, and a community-centric web site used to submit candidate URLs for addition to the filter.

Given the scope of the work involved to develop a working prototype, students selected available development tasks as the basis of their research work. For example, following one student's investigation of the development of a Mozilla plug-in that would support the application of the filter to the end-users browsing experience, the decision was made to develop the prototype as a filter module in the Apache proxy web server.

Two students at this point then researched the architecture of the Apache module system and created incrementally complex software releases that were used to explore the capabilities of an Apache module. By the completion of their mentored research experience, the project was left with a semi-

working Apache module which facilitated further investigation.

During the summer of 2005, our department offered an eight week research program for computer science majors interested in exploring facets of the discipline not typically covered in their curriculum. This summer experience was designed to introduce an immersive research experience in a small group setting that mimicked the environment typically found in graduate school. Three faculty mentors each worked with a pair of students on individual projects (the *communiFilter* project was among them). In addition, the community of students and faculty participated in an overarching information security examination that incorporated field trips, guest speakers, research readings and in-house analysis activities.

During the summer research program, a rising senior and rising sophomore worked on analyzing the existing prototype and redesigning the Apache module. Ultimately, the existing module was discarded and rewritten from scratch in an effort to streamline the existing implementation. This effort left the project with a greatly improved prototype on which the project continues to grow.

3.2. Mentored Research Deliverables

Generally, the student's deliverables for mentored research experiences are defined by the supervising faculty member. For many students, this typically includes background research and reading, experimentation design and implementation, and original software development.

Additionally, as part of the mentored research experience, each student is required to present an end-of-term research paper encapsulating the body of their work for the semester. In addition to the paper, students enrolled in a mentored research course for the first time are required to present their work in poster format at the end of the semester, alongside their peers. Students who have enrolled in their second mentored research course are required to present their work orally at the conclusion of the semester in a special forum which showcases student research within the department.

Students enrolled in the mentored research offerings are generally viewed to be potential candidates for considering furthering their studies at the graduate level. As such, mentoring faculty often encourage their students to consider authoring (or co-authoring with the sponsoring faculty member) an academic

paper for submission to a conference prior to the completion of their studies. Only a select few pursue the opportunity, but these have led to a number of submissions to the ACM Technical Symposium on Computer Science Education (SIGCSE) in the form of a paper or poster submission, as well as student papers submitted to smaller regional conferences such as the Consortium for Computing Sciences in Colleges – Northeastern Region (CCSCNE).

3.3. Project Longevity

The *communiFilter* project will provide the basis of mentored research projects for the next few years. Contained within the project's architecture are sub-problems that can be easily distributed to interested students in semester-long blocks of time. These sub-problems provide a cross section of computer science issues that can be used to attract students with varied research interests.

For example, a usability study should be performed on the community-based filter-building web site. Such an undertaking could be attractive to students with an interest in human computer interaction, and could yield insights into how community members perceive their roles and activities in building the filter, as well as provide enhancements to the filter creation process.

Another target for future student research work is locating and correcting the causes of intermittent crashes by the filter. Currently, the filter will sporadically crash when using a variety of web browsers (Internet Explorer, Firefox, Opera, etc.) Initially, this problem was identified when the system was tested against non-Internet Explorer web browsers; however this was later proved to not to be the case. Initial debugging attempts indicate that the problem likely rests in the exchange of header information when initially establishing the proxy request (the proxy request packets vary from platform to platform). This problem will likely appeal to students interested in networking issues in software development, as well as those interested in furthering their knowledge of the inner workings of the Apache architecture.

4. Conclusions

Ultimately, it is our desire to begin to transition the *communiFilter* project into on-site testing at a local school or library where we can study the efficacy of the filter "in the wild", as well as how communities of users collaborate to build a web content filter. It is anticipated that we will be able to begin user testing

in a controlled setting within the coming academic year.

The *communiFilter* project holds promise both as an interesting social application and as the basis of interesting research projects for undergraduate computer science students. The breadth of remaining project issues has added a measure of broad appeal for students. Undergraduate majors interested in undertaking a mentored research option in our curriculum can do so under the *communiFilter* umbrella while concentrating their efforts in such areas as software development, networking, server-side applications, web site development, and databases.

References

- [1] Androutsopoulos, I., Koutsias, J., Chandrinou, K. V., and Spyropoulos, C. D. 2000. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *Proceedings of the 23rd Annual international ACM SIGIR Conference on Research and Development in information Retrieval* (Athens, Greece, July 24 - 28, 2000). SIGIR '00. ACM Press, New York, NY, 160-167.
- [2] Berners-Lee, T., *Creativity in Cyberspace*, Keynote Address at Software Development (Expo), Washington, D.C., October, 1996.
- [3] Davidov, D., Gabrilovich, E., and Markovitch, S. 2004. Parameterized generation of labeled datasets for text categorization based on a hierarchical directory. In *Proceedings of the 27th Annual international ACM SIGIR Conference on Research and Development in information Retrieval* (Sheffield, United Kingdom, July 25 - 29, 2004). SIGIR '04. ACM Press, New York, NY, 250-257.
- [4] Dumais, S. and Chen, H. 2000. Hierarchical classification of Web content. In *Proceedings of the 23rd Annual international ACM SIGIR Conference on Research and Development in information Retrieval* (Athens, Greece, July 24 - 28, 2000). SIGIR '00. ACM Press, New York, NY, 256-263.
- [5] Fawcett, T. 2003. "In vivo" spam filtering: a challenge problem for KDD. *SIGKDD Explorations Newsletter*. 5, 2 (Dec. 2003), 140-148.
- [6] Graham, P., *A Plan for Spam*, Internet URL: <http://paulgraham.com/spam.html>, last accessed 4/28/2006, August, 2002.
- [7] Mozilla Foundation, Thunderbird Email Client, Internet URL: <http://www.mozilla.com/thunderbird>, last accessed 4/28/2006.
- [8] Sebastiani, F. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*. 34, 1 (Mar. 2002), 1-47.
- [9] *SpamBayes*, Internet URL: <http://spambayes.sourceforge.net/>, last accessed 4/28/2006.
- [10] The Apache Software Foundation, *Apache HTTP Server Project*, Internet URL: <http://httpd.apache.org/>, last accessed 4/28/2006.
- [11] Wikipedia – The Free Encyclopedia, *Bayesian inference*, Internet URL: http://en.wikipedia.org/wiki/Bayesian_statistics, last accessed 4/28/2006.
- [12] Zhang, L., Zhu, J., and Yao, T. 2004. An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing (TALIP)* 3, 4 (Dec. 2004), 243-269.