

Finding Computer Science Syllabi on the World Wide Web

Matthew Phillips, Fernando Das Neves, Edward A. Fox

Digital Library Research Laboratory
Virginia Tech
Blacksburg, VA 24061, USA
{mphillip, fdasneve, fox}@vt.edu

1. Abstract

Syllabi contain information useful to students, faculty, and many other people, and given the ubiquity of the WWW many schools are now putting their syllabi online for these people and the general public to access. Even though these syllabi may be available, they might be hard to find. This means that faculty, students, and anyone else who might have an interest in viewing those syllabi might find it useful to be able to browse a collection of reliable syllabi. To build a collection of reliable syllabi it is necessary to find those syllabi on the WWW but this is made easy with a tool like the Google Web API. Once the syllabi are found on the Web it is necessary to examine those syllabi and look for desired characteristics to be sure they are desired syllabi. The syllabi that contain the desired characteristics are kept and the rest are discarded. This elimination process can be accomplished using a tool like a classification tree, more specifically, tools like the Orange Data Mining Library and C4.5. This paper describes the process of finding syllabi on the WWW using the Google Web API, retrieving those syllabi using Python, and filtering them using the Orange Data Mining Library and C4.5 so that a reliable set of syllabi can be constructed.

2. Introduction

As the WWW (Web) expands into almost every part of our society we see many changes coming about. One of these changes is the transition from materials being printed and distributed physically to materials being published online. One thing that falls into this category is a syllabus. Schools publish syllabi online instead of publishing them on paper due to cost, time, and other factors. Given that many of these syllabi are now available online and generally open for public viewing, a browsable collection can be created to gather the wealth of information that exists in these syllabi.

An easy way to gather these syllabi is through an existing web search engine. Given that such a large number of schools are publishing a large number of their syllabi on the Web a powerful way of using a search engine is needed. One of the easiest to use and most powerful search engines available today is Google. Google provides an API called the Google Web API that allows for easy gathering of large numbers of web pages using a programming language like Python. When a program language like Python is used, a direct path to Google's vast index of web pages is available.

Note: This report was originally drafted in the Summer of 2004 as a product of an independent study with Dr. Edward A. Fox and Dr. Fernando Das Neves. With additional research it was revised on 10/2004 and on 2/2005. Matthew Phillips took a break from school and thus the finalization of this document was suspended until the Fall of 2006.

A language like Python can be used to gather syllabi via the Google Web API and examine those syllabi for preferred characteristics. These characteristics can be stored in the form of a table.

These characteristics can then be compared with the characteristics of a preferred set of syllabi using a classification tree. Once a potential syllabus has been assessed it is easy to either keep or discard that potential syllabus. If a syllabus is determined desirable it can be added to the collection of desirable syllabi and if it is found to be undesirable it can be discarded. In this fashion a collection of good or desirable syllabi can be built.

3. Literature Review

Syllabi

Given the ubiquity and the ease of use of the Web many educational institutions now have a number of their syllabi online. This is especially true for computer science syllabi given that most computer scientists are Web enthusiasts.

Having syllabi available online is beneficial for schools as it saves resources (e.g. paper, toner, etc.) and time. The cost to host a web page containing a syllabus is very small compared to the cost of printing out many syllabi for many students. In addition to saving the school money, it provides students a convenient, solid reference. That is, in general, the location of the online syllabus does not change.

Another benefit of online syllabi is that it provides an easy way to have a readily available archive. That is, once a school starts a new semester, a new syllabus with a new name associated with the semester and school year is put up on the web-server and the old syllabus remains there too. This is not a problem since each syllabus will have a distinct name.

In general, schools have no need to keep their syllabi private and therefore they share them with anyone who has access to the Web. MIT is a prime example of this with their Open Courseware Project (OCW) [1]. MIT has provided a great deal of information about their courses including their syllabi through their OCW program, but that discussion is beyond the scope of this report.

Given that a large number of schools have substantial internet presence and that within each of these schools many courses are offered, many syllabi are found on the Web. At the time (2004) of the first version of this report, Google reported that it had more than 1,690,000 results relating to online syllabi [2].

These online syllabi are useful to their target audience, the students and faculty at that school, but they are also useful to many other people. These other people include potential students trying to learn material taught in a specific course without taking that course, a faculty member who is slated to teach a class and needs to reference existing syllabi, etc.

A syllabus from a computer science related class is likely to contain many things that would be useful to a person seeking information relating to the syllabus topic. One of the most important things a syllabus can contain is a list of topics, a course outline, that is to be covered in the class. This is generally a fairly detailed breakdown of the topics that a class will cover in a semester. The reader of the syllabus can take these topics as a good starting point for fleshing out a study on their own. That is, a person wanting to learn more about the area of computer science covered in the syllabus could view the main points covered in a course and then do further research on each one of those points.

Another important thing contained in many computer science syllabi is a reference list, or a bibliography. A bibliography in a syllabus usually contains a listing of related papers that have been published. These papers are often supplemental material to a textbook used in a class, but if they are complete enough, they can be used as the only reading material for a class. Having

access to existing reading lists can be helpful to a faculty member who is in the process of putting together a new syllabus. Referencing existing bibliographies can obviously save the faculty member time, but it also can strengthen the field of computer science by keeping topics covered in a class somewhat standard when moving from school to school.

Many other things contained in syllabi on the Web can be useful to people looking for information relating to syllabi. These things include: texts being used for the course, projects/assignments given in the course, homework being assigned in the course, grading schemes being used in the course, reference material other than that contained in the bibliography needed for the course, schedules being used to cover the material, and many other things.

The content of a syllabus is important to a user trying to find out information relating to the topics discussed in the syllabus, but someone like a faculty member who is trying to construct a syllabus might be looking for other things in a syllabus such as structure. The form or structure of a syllabus is sometimes almost as important as a syllabus's content [3].

Given the importance of a syllabus and their abundance on the Web we need to have a way to locate them. This is where something like the Google search engine [4] becomes important.

Google

Google is a powerful search engine that indexed more than 4,285,199,774 web pages [5]. Within Google's stored index there are at least 1,690,000 web pages relating to syllabi [2]. Not all of these web pages are good, useful syllabi, but they are somehow related to syllabi.

To deal with the large volume of web pages that are indexed by Google, Google Research has provided developers and researchers with the Google Web API [6]. The Google Web API is a set of tools that allows developers and researchers direct access to Google's index and provides them with a more powerful interface than the one found through a web browser.

Instead of interacting with Google through a web browser, communication is accomplished through the Simple Object Access Protocol (SOAP). SOAP is a high level protocol that transforms information into the eXtensible Markup Language (XML) and uses HTTP to communicate the XML via the internet. SOAP simply uses standard HTTP methods such as GET, POST, HEAD, etc. to communicate the XML between web-servers [7]. Given that the only thing needed to communicate using the Google Web API is SOAP, which only relies on common internet technologies, the Google Web API is accessible from most any web-server.

The common setup for using the Google Web API is to have a script (usually written in something like Perl or Python), running on a local server, access SOAP, which uses XML to pass data via HTTP to the Google Web API server. Visually, the process looks something like Figure 1.

SOAP, XML, and HTTP are all standard installations on a web-server. The only other thing needed to gather information from the Google index is the Google Web API. The Google Web API is freely available for download [6]. The Google Web API can be used with many different languages including Perl, Python, Java, C++, C#, etc. The most popular languages used with the Google Web API are scripting languages such as Perl and Python, but just about anything can be used [8].

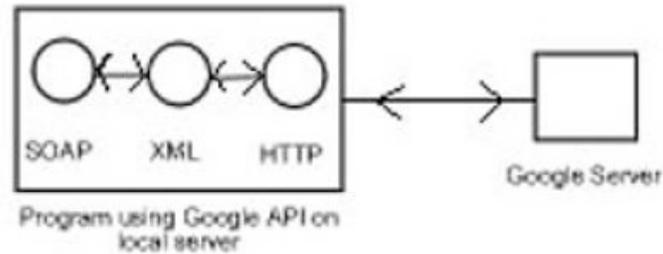


Figure 1: Communication process of the Google Web API.

Python

Python is a versatile and powerful scripting language that includes many built-in tools. It is an excellent choice for working with the Google Web API because, in general, when working with the Google Web API, most information being manipulated is in plain text. Python has great support for text parsing with its string package. Inside the string package Python has full support for text parsing and string manipulation including support for regular expressions [9].

Python is also an excellent choice for working with the Google Web API because a Python Google toolkit has been designed specifically to allow easy interaction between Python and the Google Web API. This toolkit is called PyGoogle and is the work of Mark Pilgrim [10]. Pilgrim has designed PyGoogle so that it is a Python wrapper module that handles of the underlying SOAP and XML interactions. Since PyGoogle takes away the messy SOAP and XML interactions it leaves the developer free to work on higher level problems such as performing useful tasks with the information gathered from the Google Web API. PyGoogle is a set of scripts that are simply downloaded to the local server. There is no installation process associated with PyGoogle; you simply have Python import PyGoogle as a package and you are free to use it.

To make sense of the large number of pages available through the Google Web API, a Python package like the Orange Data Mining Library needs to be used [11]. The Orange Data Mining Library is a machine learning Python package that includes support of R. Quinlan's C4.5 [12].

The Orange Data Mining Library and C4.5

There are many functions in the Orange Data Mining Library that are useful for machine learning and artificial intelligence, but for sorting though information retrieved via the Google Web API it is most straightforward to use the built in support for C4.5.

C4.5 is the work of J. R. Quinlan and has grown to be the de facto standard for constructing decision trees in the machine learning world [13]. C4.5 is a program for inducing classification rules in the form of decision trees from a set of given examples. Specifically what happens is that you give C4.5 a table of good, or controlled data and it trains itself as to what to look for by generating association rules. It then uses those association rules to sort the test data that is entered into two groups: data that is to be discarded, and data that is to be kept.

To get a feel for C4.5 consider the following example. This is taken from [13]. Suppose we want to find out if a specific golfer will play given the data in Table 1.

Table 1: Golfer data. [13]

OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY
sunny	85	85	false	Don't Play
sunny	80	90	true	Don't Play
overcast	83	78	false	Play
rain	70	96	false	Play
rain	68	80	false	Play
rain	65	70	true	Don't Play
overcast	64	65	true	Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play
rain	75	80	false	Play
sunny	75	70	true	Play
overcast	72	90	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't Play

We will build a classifier which, based on the features found in Table 1, OUTLOOK, TEMPERATURE, HUMIDITY and WINDY, will predict whether or not the golfer will play. There are 2 classes: (play) and (don't play). There are 14 examples. There are 5 examples that give the result "don't play" and 9 examples that give the result "will play."

From our data in Table 1, C4.5 will create the decision tree found in Figure 2.

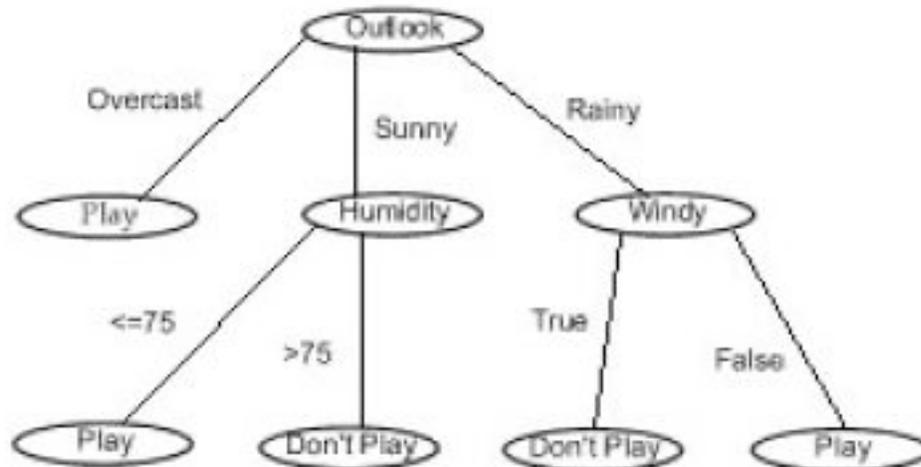


Figure 2: Decision tree for golfer data from Table 1. [13]

You can see how the golfer data shown in Table 1 can just as well represent most any other type of data that has discernible attributes. In keeping with the information in this report, Python can be used to gather syllabi on the Web using the Google Web API. After the syllabi are gathered information can be generated about their characteristics, and those characteristics can be placed in the form of a tables. Once the tables are constructed they can be used as input for C4.5 and compared to a training table to produce a decision tree so that two groups of syllabi are sorted: syllabi that are to be discarded, and syllabi that are to be kept.

4. Experimental Report

Finding Syllabi

To build a high quality collection of computer science syllabi a script was written using Python version 2.2. This Python script, getsyllabi.py, used PyGoogle version 0.5.3 and passed to it the number of web pages it wanted, 10, and the entry number (index) in the result list at which to start. The Google Web API only allows retrieval of 10 web pages at a time, so a loop is needed to bump the index by 10 each time through the loop. For example, getsyllabi.py calls PyGoogle the first time and says “get me 10 web pages starting at index 0.” So the first time through the loop, web pages 0 through 9 are gathered and the second time through the loop, web pages 10 through 19 are gathered and so on. The Google Web API limited searching to 1000 queries every 24 hours so care had to be taken not to accidentally set the loop to an outrageous number. It was found that in general a lot of noise, or non-syllabi, occurred after about 1000 web pages. gathersyllabi.py was programmed to only search educational domain names, .edu, to improve the likelihood of only gathering syllabi. So, when a search for syllabi relating to an Operating Systems course, the Google Web API search string looked something like “operating systems site:.edu”.

A getsyllabi.py search was used for each syllabus topic and a text file was created containing all the matching URLs returned from getsyllabi.py. To clarify, for each course of interest, getsyllabi.py found about 1000 web pages and create a text file for each set of about 1000 web pages and then put the URLs associated with those web pages in the text file. A total of ten topics were used in the search:

- algorithms
- compiler design
- data structures
- formal languages
- network architecture
- numerical methods
- operating systems
- programming languages
- software engineering
- UNIX

It is understood that this is far from an exhaustive list of courses offered by most computer science departments, but it is somewhat representative of the core classes offered by many computer science departments.

After finally gathering all the syllabi, 10 times about 1000 syllabi, or 10000 syllabi related to 10 different computer science course areas had been accumulated.

Retrieving Syllabi

In the previous step the getsyllabi.py script only retrieved the URL of the web page that is related to the syllabus. In order to determine if the URL contains a desirable syllabus, the HTML pages associated with each of the stored URLs needed to be retrieved. To do this a script was written called getsource.py which read a URL from one of the 10 text files and then using the `-source` option in the web browser lynx, the html source was dumped into a new text file with the name being the URL of the retrieved data. So, for each URL, a command like `lynx -source http://www.vt.edu/syllabus.html` was run after running getsyllabi.py about 10000 URLs along with the corresponding HTML source had been accumulated.

Examining Syllabi

Now that the collection of syllabi with their html sources had been gathered, it was time to determine which ones were wanted to keep and which ones needed to be discard. To determine this good examples of syllabi were examined and their characteristics were picked out. After looking through many examples, five high-quality syllabi were chosen. It was noticed that each one of these syllabi had certain keywords that they shared. All of these syllabi contained the keywords in this list:

- Lecture or Lectures
- Course or Courses
- Grade or Grades or Grading
- Instructor or Instructors or Teacher or Teachers or Professor or Professors or Lecturer or Lecturers
- Exam or Exams or Test or Tests or Quiz or Quizzes or Homework or Project or Projects

Please note that this list of keywords is not case sensitive.

A this time, a script called `buildtable.py` that used a Python string function called “find” was written to search through each html syllabus for each item in the bulleted list above. If it found an item it would insert the proper note in the field of the table, thus building a table containing the characteristics of each html syllabus. This table had about 10000 rows, one for each html syllabus that had been gathered. An abbreviated sample of this table is found in Table 2.

Also, a training table constructed that contained the characteristics of the five syllabi that were thought to be good examples of a syllabus.

Table 2: Syllabi data.

Lecture or Lectures	Course or Courses	Grade or Grades or Grading	Instructor or Instructors or Teacher or Teachers or Professor or Professors or Lecturer or Lecturers	Exam or Exams or Test or Tests or Quiz or Quizzes or Project or Projects	Syllabus
True	True	True	True	True	Keep
True	True	True	True	False	Don't Keep
False	False	True	True	True	Don't Keep
True	True	True	True	True	Keep

After the two tables were constructed, they were run through the Orange Data Mining Library’s implementation of C4.5. More specifically, the table that the `buildtable.py` script built running against the training table was used to generate a classification tree.

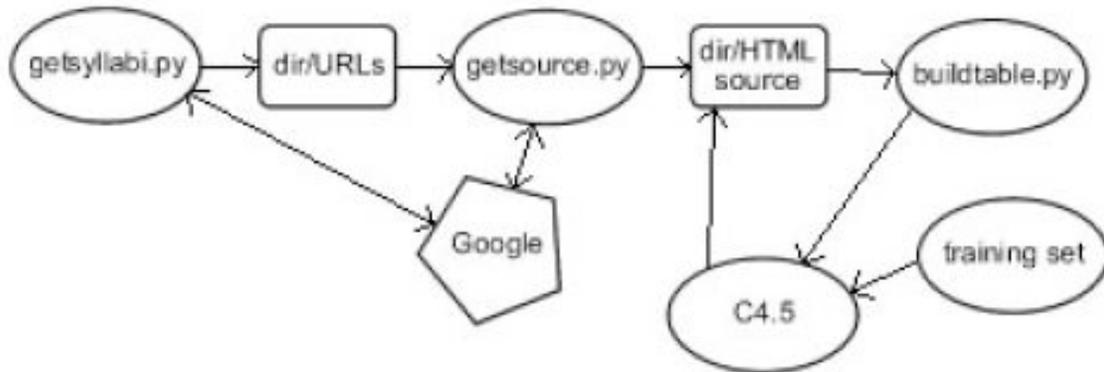


Figure 3: Overview of finding, retrieving, examining, and filtering of syllabi.

Filtering Out Undesirable Syllabi and Overview

The Orange Data Mining Library makes the filtering of data trivial as it only requires four instructions: load training table, build classification tree, run test data through classification tree, print out results. Figure 3 shows a high-level overview of this process that has just been discussed.

Results

After running the test data through the classification tree it was found that 8399 out of about 10000 met the requirements set previously for a syllabus.

After receiving these results, a script called `getfinal.py` that finds the difference between the two sets of syllabi and moves the rejected syllabi out of the directories and into a directory called `rejected.nameofcourse` was used. After this, 10 directories containing the html source for what was determined to be good quality syllabi was created in addition to a set of directories called `rejected.nameofcourse` containing the rejected syllabi.

5. Future Work

A great deal of work can be done to expand this project. Three of the bigger things that are not thoroughly researched are:

- More syllabi need to be gathered according to course topic: Only gathered syllabi for 10 course topics. There are many more courses that are part of computer science departments and the collection is incomplete without them. This is something of a difficult problem because some courses have different names at different schools. For instance, a class dealing with object oriented programming might be called Object Oriented Programming, or it might be called Object Oriented Design, or it might be called Object Oriented Programming with C++, etc.
- Further examination of syllabi: Other characteristics can exist in syllabi. Only syllabi according to words they contain were examined. It is possible that common structures can be easily found in syllabi. It is possible that URLs can be taken into consideration more when determining if a syllabus is desirable.
- Additional file formats need to be included: Only examined plain text and html syllabi, but other formats exist. Many syllabi were encountered that were in PDF,

PS, RTF, or Microsoft Word format. It would be useful to examine these syllabi too.

References

- [1] MIT Open CourseWare Project, <http://ocw.mit.edu/OcwWeb/Global/AboutOCW/aboutocw.htm/>, 2003
- [2] Google, <http://www.google.com/search?q=syllabus8/>, 2004
- [3] Judith Grabiner, Course syllabus: perspectives on computers and society, ACM SIGCAS Computers and Society, Volume 12, Issue 4, 1982
- [4] About Google, <http://www.google.com/about.html/>, 2004
- [5] Google, <http://www.google.com/>, 2004
- [6] Google Code Home, <http://www.google.com/apis/>, 2004
- [7] Reuven Lerner, At the Forge: Introducing SOAP, Linux Journal, Volume 2001 , Issue 83es, Article No. 11, 2001
- [8] Tara Calishain, Rael Dornfest, Google Hacks 100 Industrial-Strength Tips & Tools, O'Reilly Publishing, Sebastopol, CA, USA, 2003
- [9] Mark Lutz, David Asher, Learning Python, O'Reilly Publishing, Sebastopol, CA, USA, 2003
- [10] Mark Pilgrim, Python wrapper for the Google web API, <http://diveintomark.org/projects/pygoogle/>, 2003
- [11] University of Ljubljana Orange Data Mining Project, <http://magix.fri.uni-lj.si/orange/>, 2003
- [12] J. R. Quinlan, Introduction to Decison Trees, Machine Learning, Volume 1, Pages 81 –106, 1986
- [13] Ian Witten, Eibe Frank, Data Mining Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann Publishers., San Francisco, CA, USA, 2000