

Finding Perl Modules

brian d foy, comdog@panix.com

Abstract

The Comprehensive Perl Archive Network has over 11,000 modules in over 3,000 distributions. Any problem worth solving probably has a module, but many people do not know how to find the module that they need. In this article I show how I do it.

1 Introduction

I worked on a web project which included a customer login screen, and I found, after studying the error logs, that people often mistype their usernames. They do not mean to mistype their names, so they often do not know why the login failed. I thought that if I could take what they actually typed and, if it did not match an existing username, find something close, I could create a better interface that helped them solve their login problem while generating fewer support tickets for me.

Donald Knuth talks about this sort of situation in *The Art of Computer Programming, vol. 3*. The simplest of names, including my own, end up in very strange forms in databases. Knuth invented a scheme he called soundex which reduces names to a short string of characters and letters such that similar sounding names—Smith and Smyth, for instance—reduce to the same string despite their different spellings, making them easy to find.

If I wanted to find similar usernames, I could have either written some code to implement soundex or used code that already existed. Given a reasonable level of functionality, I prefer to not create code that I can get somewhere else, and I can find these modules easily with a few simple practices.

2 The Perl modules list

Since I already knew about soundex, I searched for the term “soundex” in the Perl Modules list (search.cpan.org did not exist yet). I had already scanned the complete modules list, so I knew there was a top level category named Text and that it had all sorts of modules that did text manipulation and matching tasks. I also knew that there was a top level category named String. I quickly found Text::Soundex and went on my way.

Sometimes I have to be stubborn though. I may not know that there is a particular module for a task, but I refuse to believe that there is not. This stubbornness does not make it exist, but it sets my resolve in finding it.

When I started to learn Python I looked for the Python module list. The task is a bit more difficult than in Perl. Jarkko Hietaniemi did a wonderful job creating CPAN, and the Perl Authors Upload Server (PAUSE), created by Andreas Koenig, has been a big part of that success. Python’s cryptically named “Vaults of Parnassus” has a lot of same benefit as CPAN, but hides itself under a too-clever metaphor. I knew that

Python must have some sort of code repository, and I did not give up when I did not find a thing called “This is the Python Code Repository”. It took me a while to finally realize that the Vaults were what I wanted, and along the way I discovered many other Python things that became useful later. Getting there may be half the fun, but it is also most of the education. If I do not take the journey, I never learn the lay of the land.

The Perl Module List does not list every module on CPAN though, so sometimes I have to do a bit more work. Enter Google.

2.1 Google

Almost any question I need to ask about Perl somebody has already asked in one of the Perl newsgroups. The groups.google.com website archives all of these posts and allows me to search them in a variety of ways. Google took over this task from Deja.com, which was the first well-known searchable usenet web archive, and which I used when I originally researched this problem.

Today, when I pretended to be unaware of Text::Soundex and searched for “fuzzy match perl”, the first four of 671 search results lead me to threads which mention Text::Soundex, String::Approx, and Text::Metaphone. I already knew about the first two modules, but not the third.

Good search terms make for useful search results. If I want to find something about Perl, I include the term “perl” and I usually get results about the computer language Perl. If I want to know something about L^AT_EX—the tool which typesets this magazine—I use the term “latex”, and I get a mix of articles about the typesetting language and various fetish devices made with the chemical called latex. If I add a second term, I get even better results. In Table 1, I list the results for three different searches in groups.google.com. In search A, I get almost three million results for just “perl”, and none of the top ten results relate to fuzzy matching. In search B I add the term “fuzzy” and get 11,000 results the first of which mentions Text::Soundex, but other results also talk about “fuzzy logic” which does not help me. If I add the term “match” in search C, I get 671 results of which the top five I think are useful.

Search	terms	hits
A	perl	2,970,000
B	fuzzy perl	11,000 first match mentions soundex
C	fuzzy match perl	671

Table 1: Search terms and their returned hits

What if I did not know that I should use the term “fuzzy” though? Computer scientists like to use that word, and somewhere in my career I learned that term, but what if I had not? I could use other terms—approximate, close, almost. When I search for “perl approximate”, the first result that Google returns is a reference to perlfaq6 which answers my question. When I search for “perl close”, most search results are about the Perl built-in function close(), but the first result mentions String::Approx, which finds close matches within strings.

Sometimes I simply type a full sentence into the search box and let the search engine figure it out. If I search for “How do I approximately match a string in perl?”, I get a useful reference in the first ten results—an announcement about a release of String::Approx. Once I read that message, I learn that the term “fuzzy matching” describes my problem, and I can use that to do more searches.

It took me less than 10 minutes to do all of these searches—certainly a lot less time than posting a message

to a newsgroup or mailing list and waiting for the reply.

If you still cannot find what you need, most search engines have some sort of guide that helps you get better results, and the methods you use for different engines depend on how they categorize and organize their information, so I do not discuss that here.

2.2 search.cpan.org

Before Graham Barr created a searchable interface to CPAN, I had to either look at the Perl modules list, look at a directory listing of all of the modules, or know exactly what I wanted. Indeed, as I said before, getting there is most of the education and I am sure that the experience gave me a good idea about what exists in CPAN, but now things are much easier. I can search CPAN based on the module name, author name, or a free text search of the documentation. If I search for “fuzzy match”, I get a result for `String::Approx`. I can read the documentation directly so I do not have to download or install the module to discover if it meets my needs.

A few other search interfaces to CPAN exist, and you might use one or another for different reasons. `WAIT` allows you to perform complex powerful searches, while Randy Kobes provides a search interface from the web and the command line.

3 Perldoc.com

Carlos Ramirez makes most of the Perl documentation searchable at Perldoc.com. Although I usually use `search.cpan.org` to look at a module’s documentation, I use Perldoc.com to search the Perl standard documentation. Now that the Perl documentation takes up over 1,000 printed pages, I no longer suggest that new Perl programmers try to scan all of it, although I think they should read the table of contents.

Carlos recently updated the search capabilities to allow searches on different versions of the Perl documentation, so I do not even have to have a recent version of Perl. If I have access to the internet, I have easy access to all of the Perl documentation that I need regardless of the particular installation of Perl.

4 Conclusion

I can find Perl modules quite easily—I am familiar with the Perl Modules List, know how to use Google, and look at the module documentation before I download the module. Instead of spending too much time looking for modules or reinventing them, I have time to write articles about finding them.

5 References

CPAN FAQ – <http://www.cpan.org/misc/cpan-faq.html>

Grok CPAN – <http://www.cpan.org/authors/id/H/HF/HFB/grok-cpan.pdf>

Perl Modules List – <http://www.cpan.org/modules/00modlist.long.html>

CPAN – <http://www.cpan.org>

CPAN Search – <http://search.cpan.org>

Google Groups search – <http://groups.google.com>

Vaults of Parnassus – <http://py.vaults.ca/parnassus/>

Perldoc.com – <http://www.perldoc.com>

WAIT – <http://wait.cpan.org>

Kobes Search – <http://kobesearch.cpan.org>