Smartphone Forum Searching:
A more efficient search engine for smartphone topics

Daniel Freedman (dfreedm2)
David Paola (dpaola2)

May 8th, 2010

Contents

List of Figures

1 The Function of Smartphone Forum Searching

Our software was created to exploit a niche in searching that other search engines fail to serve correctly: forums. In addition, we felt that in the forum space, there is no better store of knowledge underrepresented in normal searche engines than information about smart phones. Thus, we decided to make our software provide an easy, comprehensive way to search for information about smartphones. Out of this drive to better serve the communities of smartphone users came our *(rather poorly named)* Smartphone Forum Searching software.

1.1 Why is this useful?

Our software is useful because the existing experience for searching for smartphone information is quite poor. Searching on broad search engines such as Google and BoardSearcher have extraneous data in their results that obscure the useful information that is contained in forums specializing in this kind of information. Therefore, that our tool goes straight to the source of this useful knowledge means that we don't have anything getting between the user and the knowledge they seek.

2 Us vs Google

There are a few tools that do what our software is attempting to do, but they are all more general search engines. Google, being the most widely used search engine in the world, has attempted to do some indexing of forums, but unfortunately weights those documents much too low to be actually useful. In the following example, we searched for the term **webos set wallpaper** on both Google and our search engine. As you can see by the results, Google linked to images of wall papers and applications that sell wallpapers for WebOS phones at the top, while our software showed a forum thread that asked about how one might set the wallpaper, complete with a correct answer and some back and forth about nice wall papers to use. This shows that while a broad search engine like Google might provide a lot of sources of information about the query, our narrow focus on the forums of smartphone owners allows our results to be more accurate.

Figure 1: Google is uninformative on this query

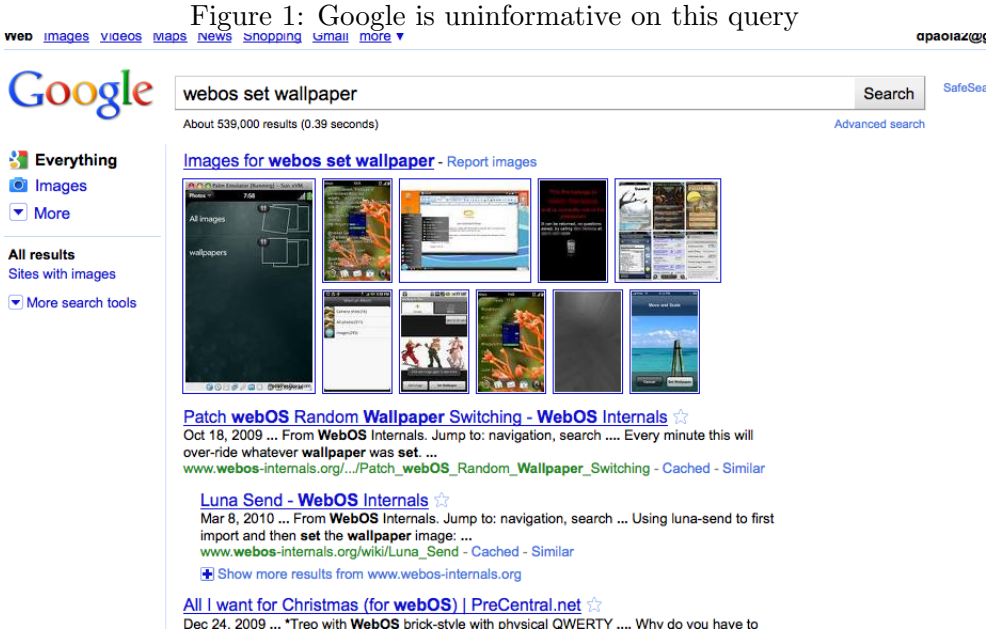
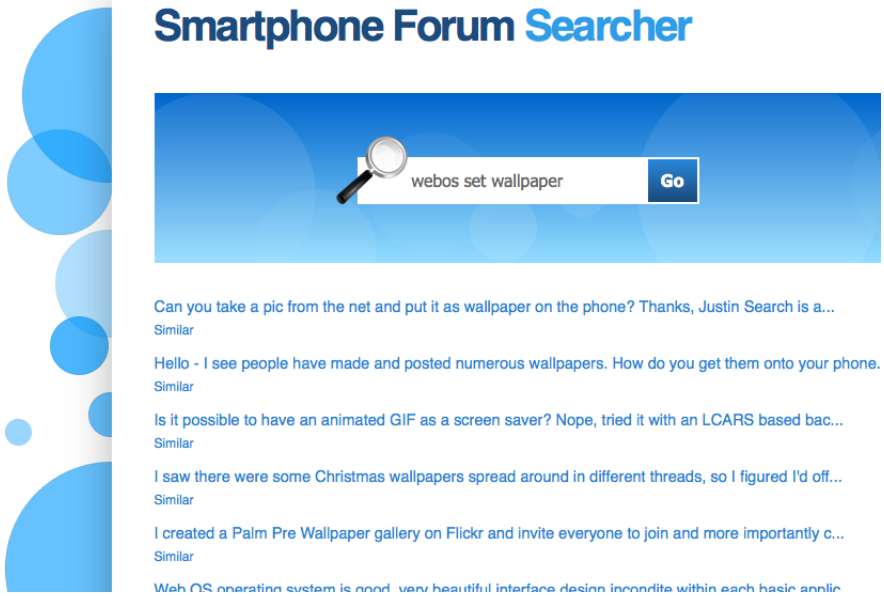


Figure 2: We are much more informative, showing both the question AND answer



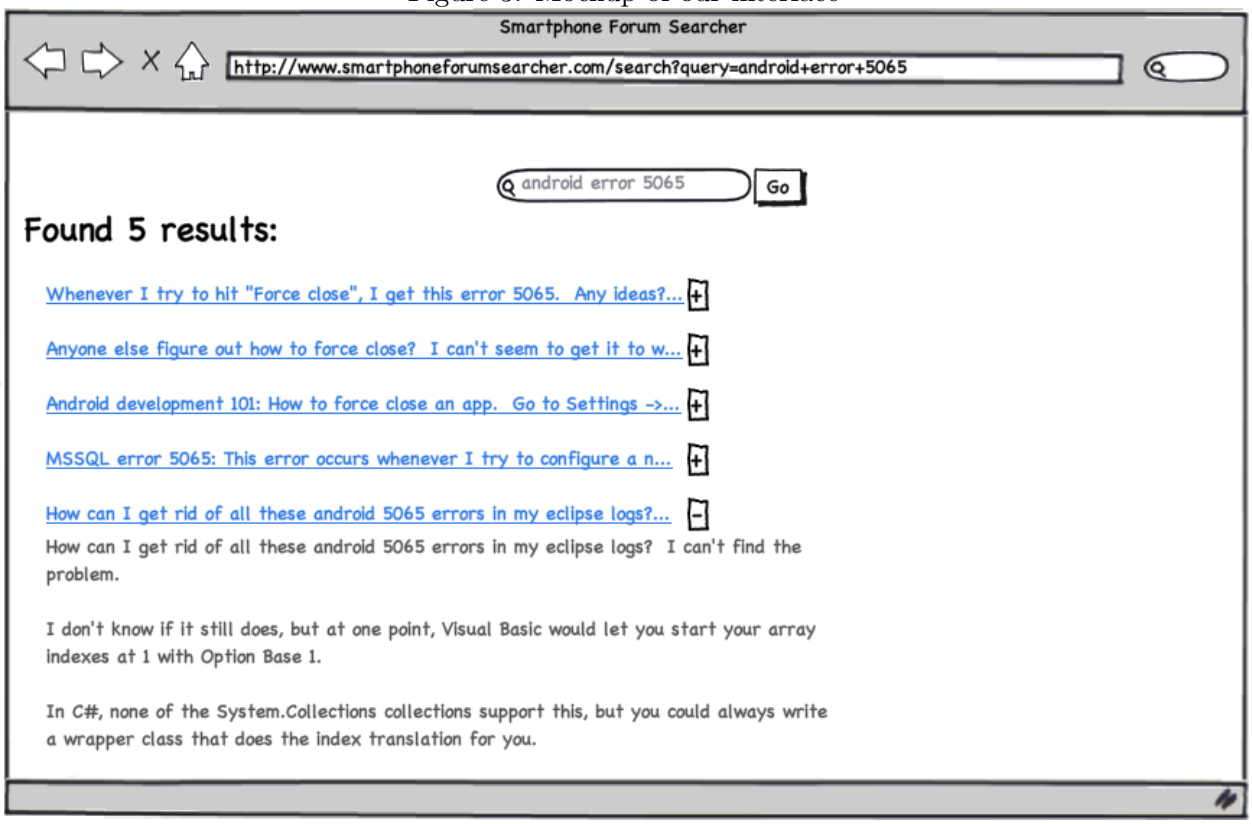
2.1 Novelty of our solution

Our solution is novel in that other tools treat posts as a document or unit for searching. We decided that this technique was not optimal because posts to a thread are intended to be parts of a conversation between users, not a one off source of knowledge. Thus, we decided that the threads of the forum would make for a much more useful unit of searching. When a user searches with

our software, they are presented with the full conversation, not just a fragment without context. In addition, we have passed the thread through an HTML parser, selecting only the actual text in the posts to save. This way, there is only the relevant text of the thread left, without any of the poster's personal data or images to hamper the reading of the thread. We also highlight the matching terms in the relevant threads, making it easier for a person to find the relevant lines of text that answer their query the most successfully.

3 Implementation

Figure 3: Mockup of our interface



3.1 Crawler

We implemented our software as two separate entities: the crawler and the website. We decided to separate the tools because each of us picked what we wanted to do, and decided that different languages would work better for the crawler and website.

For the crawler, we decided on using Perl with the WWW::Mechanize and HTML::TreeBuilder

modules. The WWW::Mechanize module is a useful Perl library that wraps around a web browser with scriptable events. This means that we can have Mechanize click on links and do actions based on Perl regexes. Mechanize can also return the content of the page without rendering it, which we then give to HTML::TreeBuilder.

TreeBuilder is a library that understands HTML that can turn HTML into a tree internally, and allow one to search for specific tags and/or content. This allowed us to figure out what the posts in the thread use as an id, then use a regular expression to return a list of posts. Included in the library is an HTML parser that can return arbitrary HTML as processed text. We used this functionality to clean the thread posts into pure text, which is much more useful to the user than text full of HTML tags that distract from the knowledge being presented.

Using these libraries, we made a script that scanned the first 20 pages of threads from each board in the forum, then retrieved all pages of the thread and ran a search over the threads to return post messages. We turned these messages into text with TreeBuilder, then saved the whole thread to a file.

3.2 Website

The Smartphone Forum Search website is a mashup of several technologies. We chose to use the Python language, which means we had access to a plethora of tools, frameworks, and libraries. On the backend, we deployed an instance of Django, a popular Model-View-Controller framework for creating robust websites and web applications. Inside django, URLs are mapped to specific functions. We have a "search" function and "similar" function, for example. In turn, these make calls into our SearchFiles.py library. SearchFiles returns python lists containing the relevant search result documents and the django instance renders them into html for the user to view in their browser.

SearchFiles utilizes a library called PyLucene. PyLucene is an Apache project that aims to mirror the Java Lucene library. SearchFiles (and our indexer) use PyLucene to retrieve results and find similar documents based on a given document. These files are straightforward enough, making calls into the PyLucene API and packaging up the data for the django instance.

The last piece of the puzzle is Javascript and jQuery. An important part of any web technology is usability and good design, so we chose to provide the user with a quick, easy, and useful way to

view the documents relevant. We use jQuery (a popular Javascript library) to allow the user to click the document title and see the document contents drop down. The original URL is maintained at the bottom of each thread. So, the user never has to leave our website to view the information they're seeking. This is an area we plan to improve. Good design is difficult!

3.3 Challenges

3.3.1 PyLucene Documentation Issues

PyLucene turned out to be a pain. Not being familiar with Lucene to begin with, the lack of good documentation for PyLucene was nearly crippling. It was, at times, difficult to figure out which parts of Lucene PyLucene mirrors. For example, PyLucene contains NONE of the benchmarking features provided in the Lucene API. This is a serious drawback that was not documented anywhere. PyLucene also chose to collapse the namespace of Lucene. So, for example, Java's lucene.index.IndexWriter becomes lucene.IndexWriter in the PyLucene library. Quite confusing.

These problems led me to publish a blog post detailing a very elementary tutorial and summary about PyLucene and how to use it. Hopefully this will give other new users of this library to be brought up to speed quickly. This post is available at `http://davezor.posterous.com/using-pylucene`.

3.3.2 WWW::Mechanize Memory Usage

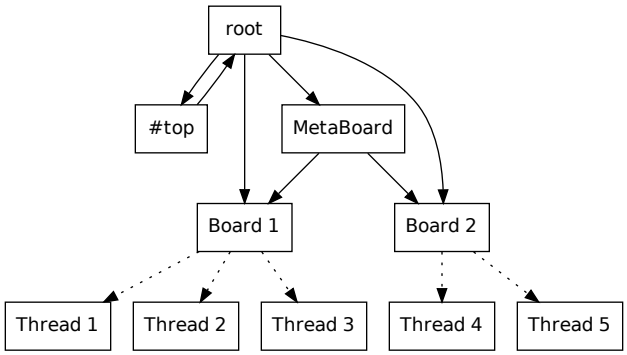
We experienced many problems with the crawler script using all the memory available on the server. This turned out to be caused by Mechanize saving a copy of each page it visits in memory, slowly building up as the crawler searches the whole forum. Fortunately, we found a constructor option that disables this website saving. With this option enabled, we were able to finish scanning the forums without the Out of Memory process killing our script.

3.3.3 Forum sizes and formats

Web forums come in many varieties of software, each with its own templates for threads, messages, and users. Unfortunately, the time alloted for this project only allowed us the opportunity to build a crawler that supported three websites, out of the 50 or so smartphone forums. This means that we

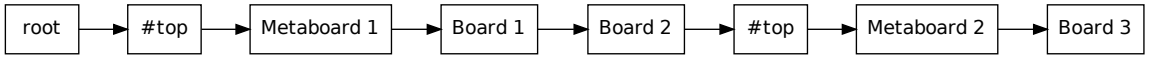
were only able to capture a tiny portion of the available information on smartphones that would fit our indexing and user interface requirements. Of the websites we were able to program for, one in particular gave us much trouble. The forum containing information about Palm had a particularly annoying setup to their forum boards, where they linked to "metaboards", or boards containing other boards, as well as boards containing actual threads from the front page. However, all the boards are also linked from the main page, leading to this interesting graph layout for the forum, which confused the crawler script.

Figure 4: Intractable Graph Problem



We solved this by taking an inorder traversal of this graph, which led to this breakdown.

Figure 5: In order traversal



This structure with the `#top` and `metaboard` nodes being in order is consistent throughout the forum, so we solved the issue by finding the location of all `#top` nodes and removing them and the node after. We could then use the crawler as expected.

3.4 Evaluation of our Performance

Unfortunately, we were unable to use Lucene's QualityReports modules, as these APIs are not exported in PyLucene, and we did not have enough time to write our own class to wrap them.

However, we did do some search analysis against Google, and found that Google's results were more useful on minimally specified queries, while our search results were much more useful than Google's when many-term queries were entered. When it comes to speed, Google beats us by an order of magnitude (milliseconds vs seconds), but this is likely due to the fact that Google has thousands of servers querying their database, while we have only the one.

4 Future Improvement

4.1 Immediate

In the immediate future, we will work on increasing the number of forums that the crawler can understand. This should allow us to very quickly increase the document count, hopefully increasing the chances of finding more relevant threads, and improving the smoothing given to the queries by Lucene.

4.2 Mid-term

Between now and the far future, redesigning the website to be more user friendly would allow for more qualitative information on the search results to be shown to the user. As of now, all we do is show the user what documents were relevant to the query, without showing any statistics or precision results along with it. Showing this information would allow for users to give us feedback, allowing us to further refine our Lucene settings.

4.3 Far future

In the far future, we would like to be able to have an online crawler, as well as allowing users to post replies to the threads. These replies would then be indexed and sent to the forum that the thread was found from, allowing users to post their own replies and continue the conversation. We feel that this would add much in the way of utility to our project, increasing its utilization and letting us take advantage of powerful feedback information on what threads were being replied to, what information is most commented on, etc.

5 Contibutions by Author

5.1 Daniel Freedman

Daniel wrote the crawler script, and this report.

5.2 David Paola

David wrote the website and the presentation slide deck.

5.3 Public Repository

We have the code hosted on GitHub at `git://github.com/dpaola2/PhoneSearch.git`