code{4}lib JOURNAL

Issue 17, 2012-06-01

ISSN 1940-5758

Discovering Digital Library User Behavior with Google Analytics

Google Analytics has advanced features for tracking search queries, events such as clicking external links or downloading files, which you can use to track user behavior that is normally difficult to track with traditional web logging software. By tracking behavior, you can use Google Analytics API to extract data and integrate it with data from your digital repository to show granular data about individual items. Using this information, digital libraries can learn how users use the site without extensive HCI studies, and can use this information to improve the user experience.

By Kirk Hess

Introduction

In my first digital library job, every so often I would have to compile reports based on the server logs to answer a question about the site. Questions such as “How many times was X document downloaded”, or “What’s the most popular item in our collection?” were common and I answered them using AWStats data collected from our server logs. However, I was concerned the results were not accurate because of many inconsistencies, such as how many of the IP addresses were from non-.edu domains, or the page entry and exit numbers didn’t match up.

While working on the Illinois Harvest Portal I found there was no web analytics solution enabled on the website, and the web logs were no longer easily accessible. I turned to Google Analytics, which allowed me to answer simple questions about numbers of user hits. But one of the things I found that Google Analytics didn’t do well out of the box was tracking two types of events: clicks on links to other domains, such as the Internet Archive, and clicks which downloaded files. By implementing event tracking, I was able to determine which items were used and which ones weren’t being used, and I could track complete user interactions with the website.

The goal of this paper is to help librarians and programmers who develop and maintain digital library websites to use events and site search within Google Analytics. By gathering better intelligence about what users are doing, you can determine what you should change on the website to make it more useful to users without having to do extensive usability tests or human-computer interaction (HCI) studies.

Web Analytics

The Digital Analytics Association defines web analytics as:

> … the measurement, collection, analysis and reporting of Internet data for the purposes of understanding and optimizing Web usage. [1]

There are many potential solutions, both log based and script based for web analytics. Logging solutions use raw server logs to mine the information for usage trends. Your typical AWStats logs will track hits, which usually are HTTP GET requests for something, a page, an image, a file, etc. Hits on your webserver are going to be from humans, but because of the way the web is crawled, a substantial portion will often be from various automated systems like GoogleBot or BingBot. Solutions like AWStats do a good job of trying to filter out all this noise, but the problem is you never get rid of 100% of the noise. In addition, a significant subset of bots try their best to avoid detection.

Google Analytics uses a JavaScript-based approach that injects a tiny image onto each page for tracking. Because most automated crawlers do not have JavaScript or cookies enabled, and most modern browsers do, you have a high level of confidence that the hits captured are human users. Google Analytics is a simple, inexpensive, and effective way to capture data about your users.

Google Analytics Basics

Google Analytics is easy to set up on any website following the basic instructions. Generally, you sign in, input the information about your site and put the JavaScript <script> element in the <head> portion of your website. Of course, every system builds the <head> element for pages a little differently, and since you want to include the tracking cookie on every page in a consistent way you’ll need to figure out how to do this in your system. This hopefully is as simple as putting it in an include file which is included when the page is rendered.

view plain copy to clipboard print ?

```
<script type="text/javascript">
  var _gaq = _gaq || [];
  _gaq.push(['_setAccount', 'UA-XXXXXXXX-1']);
  _gaq.push(['_setAllowLinker', true]);
  _gaq.push(['_setDomainName', 'illinoisharvest.grainger.uiuc.edu']);
  _gaq.push(['_setAllowHash', false]);
  _gaq.push(['_trackPageview']);
  (function() {
    var ga = document.createElement('script'); ga.type = 'text/javascript'; ga.async = true;
    ga.src = ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') + '.google-analytics.com/ga.js';
    var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(ga, s);
  })();
</script>
```

Finally, once you figure this out it’s hard to tell if tracking actually works because there is usually some minor amount of lag in between installing the code and actually getting data in the online tool. To see the tracking cookie using Firebug (or its Chrome/IE/Safari equivalents), you simply refresh the page with the net, images tab visible. If you see an image starting with _utm.gif, you are tracking that page.

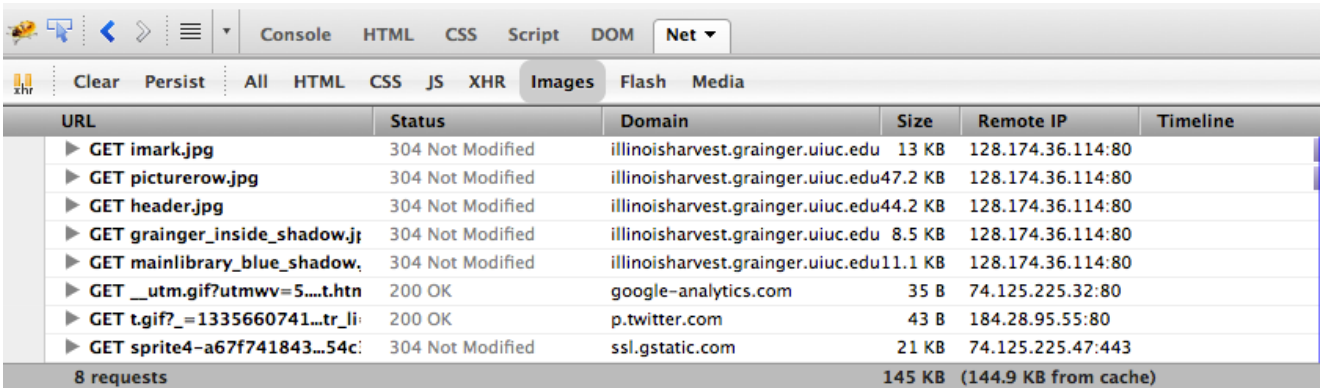

Figure 1.
Firebug showing _utm.gif?… tracking cookie.

Tracking

Events

To gather information such as clicking external links or downloading files, we'll enable event tracking to track those events. Again, Google has this documented pretty well online. Events are segmented into categories, actions, and labels. For our site, I chose two types of categories of events, external and download. For actions, I chose click, and click-, extension varying based on the type of file downloaded. The label is optional, however I chose to include the link URL or the file name. One thing I would have probably preferred was to include the internal identifier for the item that would have made linking files to items easier to manage.

In your system you may simply be able to change the code that displays links, however in our system that was too complex so instead I wrote a jQuery script that does the work after the page has been loaded, which would also be useful for tracking a vendor system that doesn't allow changes to the code. jQuery binds an event handler to the "click" JavaScript event:

```
view plain  copy to clipboard  print  ?
01. <script type="text/javascript">
02. if (typeof jQuery != 'undefined') {
03.     jQuery(document).ready(function($) {
04.         var filetypes = /\.(pdf|txt|dijv|xml)$/i;
05.         var baseHref = '';
06.         if (jQuery('base').attr('href') != undefined)
07.             baseHref = jQuery('base').attr('href');
08.         jQuery('a').each(function() {
09.             var href = jQuery(this).attr('href');
10.             if (href & (href.match(/^https?\:/i)) & (!href.match(document.domain))) {
11.                 jQuery(this).click(function() {
12.                     var extLink = href.replace(/^https?\:\/\//i, '');
13.                     _gaq.push(['_link', href]);
14.                     _gaq.push(['_trackEvent', 'External', 'Click', extLink]);
15.                     if (jQuery(this).attr('target') != undefined & jQuery(this).attr('target').toLowerCase() != '_blank') {
16.                         setTimeout(function() { location.href = href; }, 200);
17.                         return false;
18.                     }
19.                 });
20.             }
21.             else if (href & href.match(filetypes)) {
22.                 jQuery(this).click(function() {
23.                     var extension = (/[.]/.exec(href)) ? /[^.]+$/.exec(href) : undefined;
24.                     var filePath = href;
25.                     _gaq.push(['_trackEvent', 'Download', 'Click-' + extension, filePath]);
26.                     if (jQuery(this).attr('target') != undefined & jQuery(this).attr('target').toLowerCase() != '_blank') {
27.                         setTimeout(function() { location.href = baseHref + href; }, 200);
28.                         return false;
29.                     }
30.                 });
31.             }
32.         });
33.     });
34. }
35. </script>
```

The result is you get code like this injected onto your links:

```
view plain  copy to clipboard  print  ?
01. _gaq.push(['_trackEvent', 'External', 'Click', extLink]
```

Site Search

Another useful option we can enable is search tracking, and this is covered pretty well by the online documentation. Here's an example of our configuration for tracking site search.

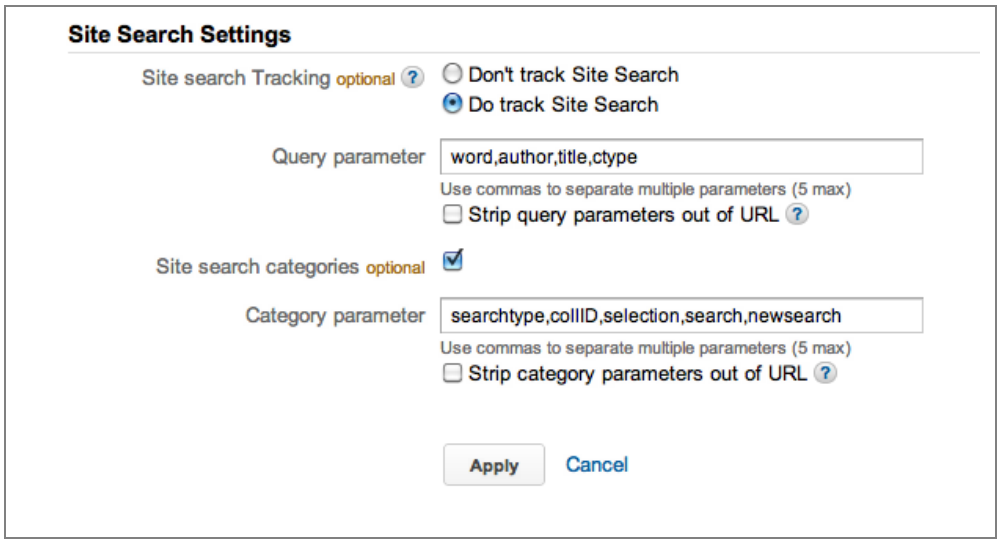

Figure 2.
Site search configuration.

Collect Data

Once everything is set up correctly, you will need to wait as activity is collected. Google Analytics has some very nice dashboard utilities to view the results of the site in real-time. Below are a couple of images of the results that I will discuss further in the Analysis section: top pages by events and search results.

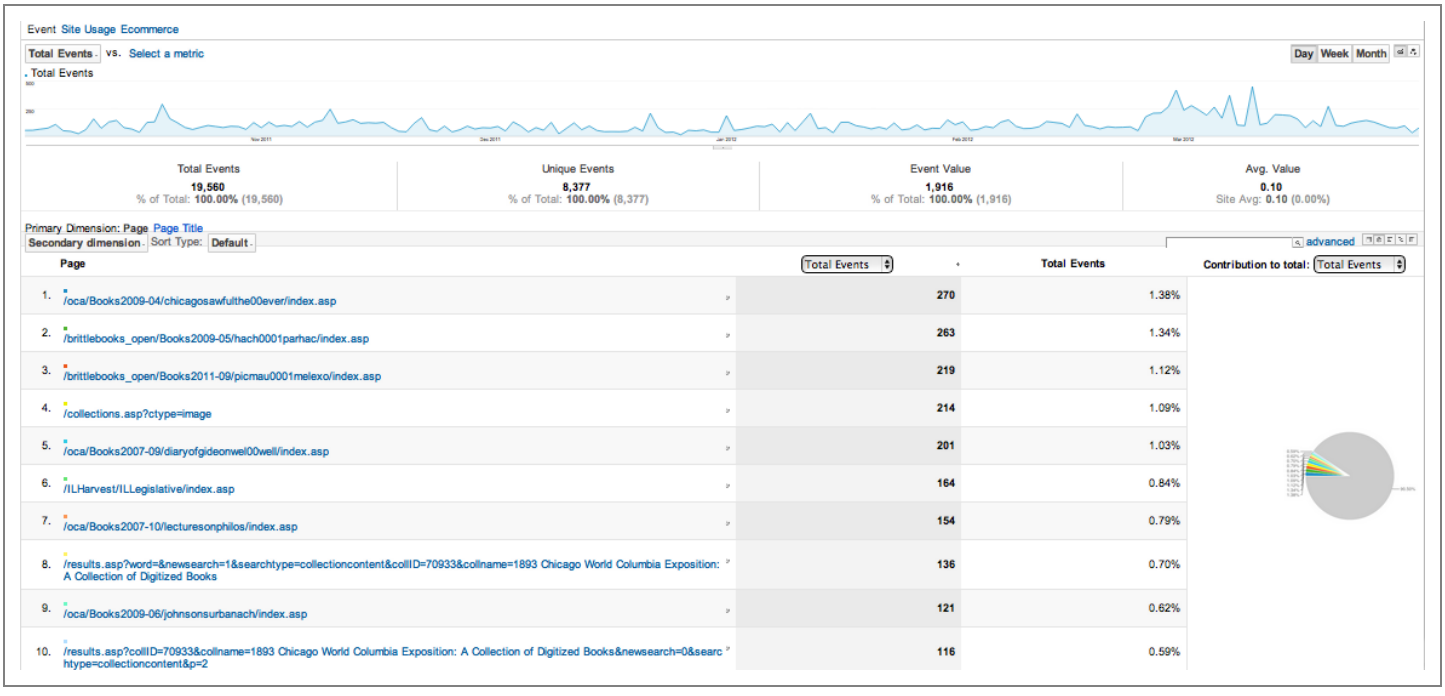

Figure 3.
Report of top pages with events from October 1 2011 to April 1 2012.

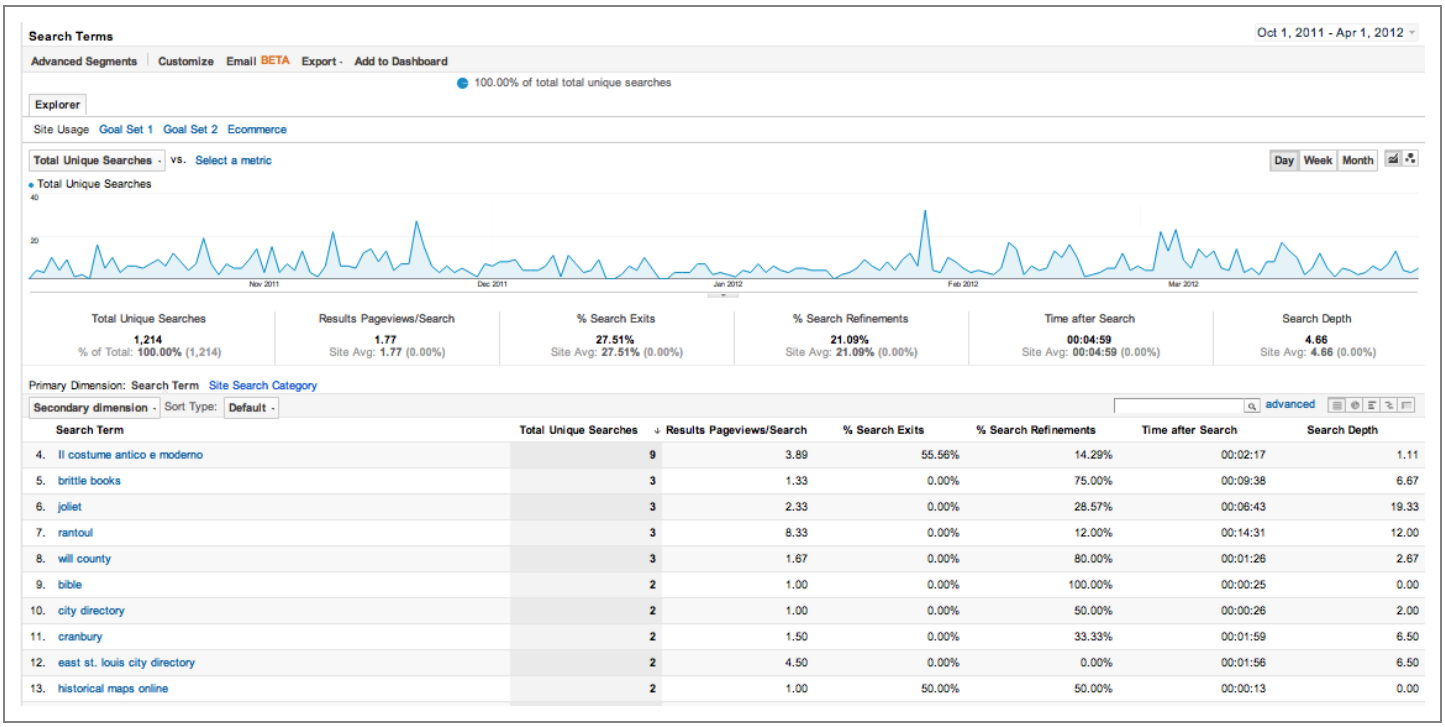

Figure 4.
Report of search queries from October 1 2011 to April 1 2012. Browse results not displayed.

Export Data

By only using the dashboard utility, I found that I wasn't getting the answers I wanted–especially for downloads. The downloads reports either were too granular (by extension) or too broad (all events), and they couldn't answer one important question: what wasn't being used? Google provides an API to extract data into whatever format you like; since our data is stored within a MS SQL database I wanted to export it into a table so I could correlate the results to all items in a particular collection. I found it easier to do with java; other programming languages are supported.

Code example: Java class for extracting data

view plain copy to clipboard print ?

```
01. /**
02.  * Retrieve data from the Google Analytics API.
03.  */
04. public class GoogleAnalytics {
05.
06.   // Credentials for Client Login Authorization.
07.   private static final String CLIENT_USERNAME = "xxx";
08.   private static final String CLIENT_PASS = "xxxx";
09.   private static final String[] eventDates; //define as needed
10.
11.   // Table ID constant
12.   private static final String TABLE_ID = "ga:123456789";
13.
14.   public static void main(String args[]) {
15.     try {
16.       // Service Object to work with the Google Analytics Data Export API.
17.       AnalyticsService analyticsService = new AnalyticsService("gaExportAPI_acctSample_v2.0");
18.
19.       // Client Login Authorization.
20.       analyticsService.setUserCredentials(CLIENT_USERNAME, CLIENT_PASS);
21.
22.       // Get data from the Account Feed.
23.       getAccountFeed(analyticsService);
24.
25.       // Access the Data Feed if the Table Id has been set.
26.       if (!TABLE_ID.isEmpty()) {
27.
28.           // Get profile data from the Data Feed.
29.           getDataFeed(analyticsService);
30.       }
31.
32.     } catch (AuthenticationException e) {
33.       System.err.println("Authentication failed : " + e.getMessage());
```

```
      return;
    } catch (IOException e) {
      System.err.println("Network error trying to retrieve feed: " + e.getMessage());
      return;
    } catch (ServiceException e) {
      System.err.println("Analytics API responded with an error message: " + e.getMessage());
      return;
    }
  }

  /**
   * @param {AnalyticsService} Google Analytics service object that
   *     is authorized through Client Login.
   */
  private static void getAccountFeed(AnalyticsService analyticsService)
      throws IOException, MalformedURLException, ServiceException {

    // Construct query from a string.
    URL queryUrl = new URL(
        "https://www.google.com/analytics/feeds/accounts/default?max-results=50");

    // Make request to the API.
    AccountFeed accountFeed = analyticsService.getFeed(queryUrl, AccountFeed.class);

    // Output the data to the screen.
    System.out.println("-------- Account Feed Results --------");
    for (AccountEntry entry : accountFeed.getEntries()) {
      System.out.println(
        "\nAccount Name  = " + entry.getProperty("ga:accountName") +
        "\nProfile Name  = " + entry.getTitle().getPlainText() +
        "\nProfile Id    = " + entry.getProperty("ga:profileId") +
        "\nTable Id      = " + entry.getTableId().getValue());
    }
  }

  /**
   * @param {AnalyticsService} Google Analytics service object that
   *     is authorized through Client Login.
   */
  private static void getDataFeed(AnalyticsService analyticsService, String[] eventDates)
      throws IOException, MalformedURLException, ServiceException {
    StringBuffer skippedRecords = new StringBuffer();
    int skipCount = 0;
    // Create a query using the DataQuery Object.
    for(String eventDate : eventDates){
    DataQuery query = new DataQuery(new URL(
        "https://www.google.com/analytics/feeds/data"));
    query.setStartDate(eventDate);
    query.setEndDate(eventDate);
    query.setDimensions("ga:eventlabel,ga:eventcategory,ga:pagePath");
    query.setMetrics("ga:totalevents,ga:uniqueevents");
    query.setSort("-ga:totalevents");
    query.setMaxResults(500);
    query.setIds(TABLE_ID);

    // Make a request to the API.
    DataFeed dataFeed = analyticsService.getFeed(query.getUrl(), DataFeed.class);
  }
}
```

Analyze Data and Next Steps

One of the things that I wondered about with Illinois Harvest on my first day was did we have any users at all? The basic statistics showed over the previous 6 months a total of 15k unique users visited the site. On average, those users visited about 4 pages, with a bounce rate (a.k.a. a single-page visit) of approximately 35%.

Figure 3 shows that we had many users clicking on external links and download links – the top 3 pages were digitized book landing pages, while the 4th page for the collection summary contains many external links. Specifically, the reason the top page had so many hits is it's one of the references for a Wikipedia page about the Iroquois Theatre fire.

Figure 4 shows an interesting yet disturbing result – after removing browse results, the site search is rarely used. Traffic to the site lands deep into the website, either at a location determined by an external search engine, the library catalog, or other direct links. The site was designed about 6 years ago when portals were considered important and it doesn't appear to be working for users who arrive from a search engine.

By exporting the data and comparing to our item level records, I was able to see that most items were never used, where use being defined as a click to an external link or downloaded a file. Figure 5 shows one of our collections, and while a couple of links get a fair amount of clicks, the vast majority of items in this collection of about 23k items are never clicked or downloaded. The disparity between the size of a collection and use my colleagues Harriet Green and Richard Hislop have termed 'Effective Collection Size' or ECS, is something we are continuing to investigate as part of larger datasets.

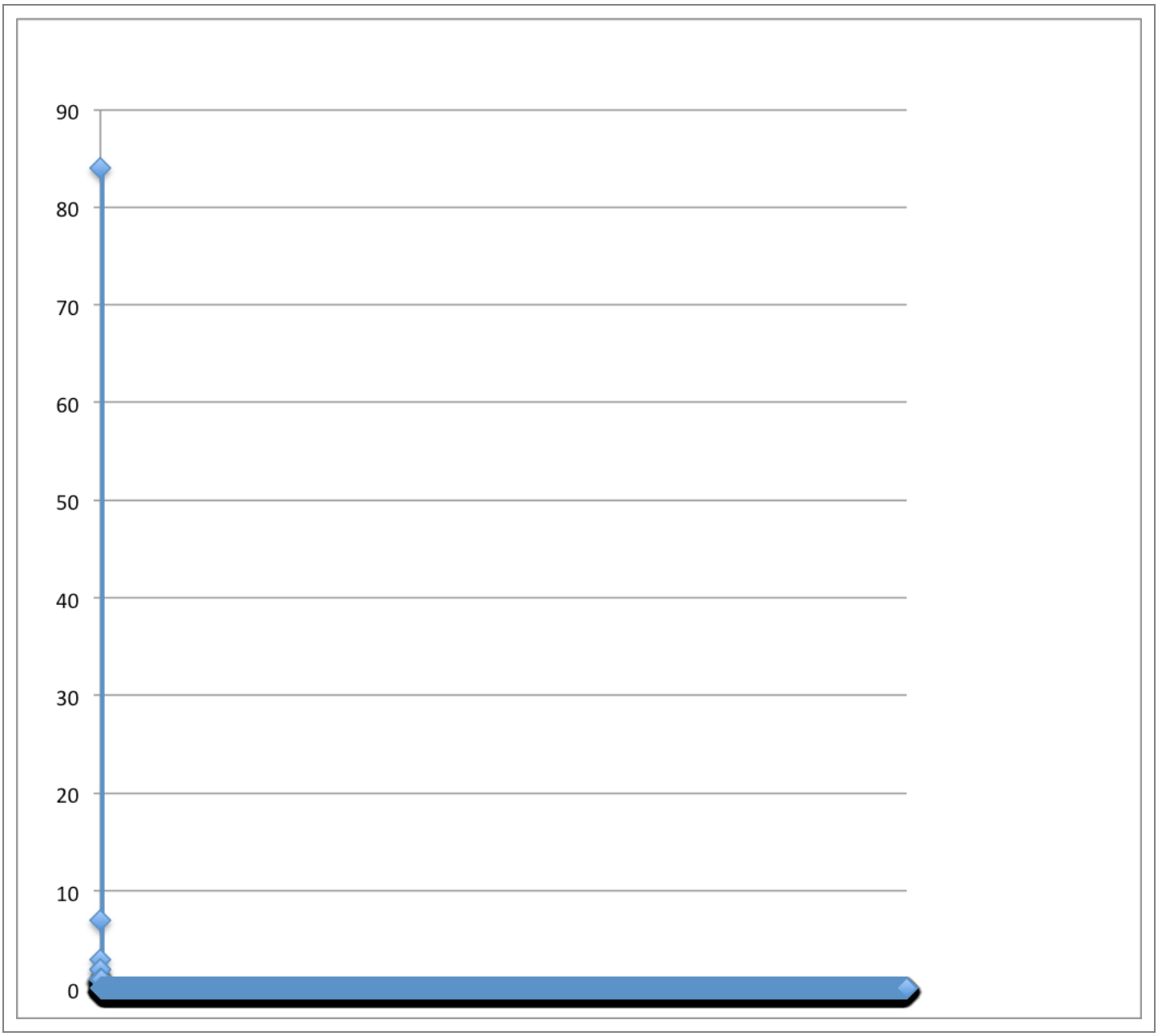

Figure 5.
Graph of item usage from IDEALS collection on Illinois Harvest, July 1 2011 – November 1, 2011

Next Steps

The Illinois Harvest user interface will be redesigned in 2012 and hopefully some of these lessons will be part of those efforts. Specifically, I think site search needs to be examined more closely, probably as a full HCI study, to make sure it's useful for users.

By analyzing a sample set of 40k items held by our Literature and Language Library, we've found a similar disparity between the size of a collection, in this case both physical and digital items, and use. We're going to be exploring how to improve ECS in a collection by examining a much larger data set of 22 million items indexed in the University of Illinois Library catalog. Based on the network analyses resulting from this project, we will begin development of an enhanced recommender system for library catalogs and digital libraries that retrieves richer search results from a library collection search based on network analysis of subject relevancy, circulation data of items, and usage data for items that share interrelated subjects.

Notes

[1] About Us. Digital Analytics Association. Available from http://www.digitalanalyticsassociation.org/?page=aboutus

Appendix: Resources

Tools Used

- JavaScript

- jQuery
- Java
- MSSQL
- Code examples are available on GitHub

Further reading about Google Analytics and Libraries

- Prom, Christopher J. Using Web Analytics to Improve Online Access to Archival Resources. American Archivist, Spring/Summer 2011, Vol. 74 Issue 1, p158-184
- Marshall Breeding, “An Analytical Approach to Assessing the Effectiveness of Web-based Resources,” Computers in Libraries 28 (January 2008): 20–22
- Feng Wei, “Using Google Analytics for Improving Library Website Content and Design: A Case Study,” Library Philosophy and Practice (July 2007), http://digitalcommons.unl.edu/libphilprac/121

About the Author

Kirk Hess (kirkhess@illinois.edu) is Digital Humanities Specialist at the University of Illinois Urbana-Champaign, and holds a MLIS from the University of Illinois Urbana-Champaign.