Accessing The Deep Web:
When Good Ideas Go Bad

Alfredo Alba, Varun Bhagwan, Tyrone Grandison
650 Harry Road, San Jose, California 95120, USA
{aalba, vbhagwan, tyroneg} @us.ibm.com

Abstract

Prevailing wisdom assumes that there are well-defined, effective and efficient methods for accessing Deep Web content. Unfortunately, there are a host of technical and non-technical factors that may call this assumption into question. In this paper, we present the findings from work on a software system, which was commissioned by the British Broadcasting Corporation (BBC). The system requires stable and periodic extraction of Deep Web content from a number of online data sources. The insight from the project brings an important issue to the forefront and underscores the need for further research into access technology for the Deep Web.

Categories and Subject Descriptors H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – Retrieval models

General Term Algorithms, Design, Reliability, Experimentation, Standardization.

Keywords Deep Web, access, crawling, application programming interfaces

1. Introduction

The problem of accessing Deep Web content has many significant issues yet to be solved, such as challenges with dynamic, unlinked, private and non-html content. These concerns are further exacerbated by the rapid growth of Deep Web content, fueled by the success of social networking online, the proliferation of Web 2.0 content and the profitability of the companies that steward in this new era. Business models, resource management strategies and long-term vision play a significant role in driving technical directions and influencing access methods.

As technology products transition to relying on Deep Web content, the gaps between reality and the assumed becomes clearer. It then becomes apparent that the reliability and efficacy of the *established* (or *preferred*) information retrieval techniques are below expectation and require improvement.

In this paper, we discuss the lessons learned from an industry-specific application deployment, the Sound Index, which showcases a set of interesting issues that must be addressed.

OOPSLA'08 October 19–23, 2008, Nashville, Tennessee, USA.

2. Setting the Context

The Sound Index is a catalogue of the most popular artists and tracks that are currently being talked about on the Internet. It incorporates listens, plays, downloads, sales and comments from a multitude of online communities and social networks.

The Sound Index system can be divided into four distinct phases [1]. The first phase is ingestion, which is the act of gathering relevant unstructured and structured content from various websites such as MySpace, LastFM, YouTube, iTunes, Google Groups and Bebo. Ingest occurs every six hours and each run consists of tens to hundreds of millions of entries. Once gathered, the content is analyzed and transformed into a standard schema. When this is completed, the now structured content is stored in a database. Finally, music charts are created by applying relevant ordering schemes. The end result is the top 1000 artists and tracks with the highest online buzz. Interested readers may investigate the Sound Index further at http://bbc.co.uk/soundindex/.

To facilitate ingestion, the system utilizes screen scrapers, application programming interfaces (APIs) and RSS feeds. During the pilot phase of the project, the anomalies exhibited in accessing some of the Deep Web content repositories showcased very interesting phenomena and emphasized a pressing (academic and industry) problem that must be fixed.

In this paper, we use two exemplar data sources, YouTube and Last.FM, (without loss of generality) to demonstrate our findings. YouTube is a video-sharing website that allows users to upload, view, and comment on videos. Last.FM is an online radio station and music community website. Each source can be accessed via a myriad of different Deep Web retrieval technology.

3. Screen Scrapers

Screen scraping is the practice of reading text data from a computer's screen [2]. By analogy, screen scraping has also become synonymous with automated parsing of the source text used to render web pages. In all cases, the screen scraper has to be able to not only process the text data of interest, but also to recognize and discard any unwanted data, such as images and display formatting.

Deep Web source data is usually behind dynamic content pages, web forms, AJAX-capable web pages, etc. These rendered pages are visually and aesthetically pleasing but in order to accomplish this, their source usually contains complex (X)HTML, accompanied by cascade style sheets, JavaScript, and even ActiveX components, amongst many other technologies. For instance, below is a snippet of source code from a YouTube video page, which exhibits a number of features clearly associated with dynamic page content, such as an em-

bedded JavaScript-driven AJAX call for displaying comments, **bolded** below:

```
<form action="" name="comments_filter">
<span class="smallText"><b>Show:</b></span>
<select class="xsmallText" name="commentthreshold" on-
   Change="showLoading('recent_comments',
   this.value);getUrlXMLResponseAndFillDiv('/watch_ajax?v=5K
   x2nJGIdcc&amp;savethresh-
   old=yes&amp;action_get_comments=1&amp;p=1&amp;page_s
   ize=10&amp;commentthreshold='+this.value, 're-
   cent_comments');">
      <option  value="-1000">all comments</option>
      <option  value="10">excellent (+10 or better)</option>
      <option  value="5">great (+5 or better)</option>
      <option  value="0">good (0 or better)</option>
      <option  selected="selected"  value="-5">average (-5 or bet-
         ter)</option>
      <option  value="-10">poor (-10 or better)</option>
</select>
```

As shown above, the *precious* data lies within a plethora of components, text, source code, markups, styles and tags. In order to extract any data, all superfluous UI-related content must be carefully filtered out and only the interesting data preserved. Also, all semantically relevant information contained in the page's text needs to be extracted too.

3.1 When To Use Screen Scrapers

For some companies, their web sites are driven by quality of service, accuracy, appeal, accessibility, availability, and other user-focused issues. They have not only made the accuracy of their site a top priority but also maintain and improve the site continuously. They also deem it important to increase the site's appeal by continuously modernizing its look and feel, increasing its accessibility, performance, and availability. For all these reasons, the web page rendered data is considered to be *the ultimate truth* for these companies; therefore becoming the de-facto source for current and accurate data.

3.2 Concerns with Screen Scrapers

While screen scraping is very useful, and the rendered web page is the most likely repository for the most accurate and up to date data available, screen scrapers can fail to provide data – or worse, provide imprecise or erroneous data – when there are any changes to the site's source code. These changes can be minor (removing a typo from a function name), or major (moving to a different layout).

The user interface paradigms also shift rapidly, more so than ever before. With the advent of increasingly flexible and scalable frameworks that enable the entire UI of high volume web sites to be revamped overnight, the site owners and developers are empowered to keep up with consumer demands of faster paced feature availability. For example, code re-factoring, which is supported by commercial development tools used by web-based companies, is often used and leads to changes to the actual structure of the rendered content within a given site. All of these activities are very beneficial for the user experience but they often imply that any given rendered page may change radically from one day to the next; including the page's location and even the semantics of the data that one is interested in.

3.3 Our Observations About Screen Scrapers

The YouTube UI is the most accurate source for data on YouTube activity, not only in terms of availability, but in terms of accessibility. As long as the crawl does not deviate too far from normal user navigation patterns, data such as user comments, views, comment counts, etc. are consistent and accessible. It should be noted that comments retrieval, in particular, presents interesting challenges. For instance, the "View all comments" link changes the ordering from reverse chronological to chronological, and it still has the potential to produce more than a single rendered page containing all available comments, e.g. Britney Spears' Gimme More video (id: m3ceCMpPJgc) has over one hundred thousand comments, and produces multiple pages even when "View all comments" is clicked. The multiple pages that normally summarize comment activity need to be navigated to achieve complete extraction. This fact means that additional logic has to be included in the crawler to consistently find the latest comments for a video.

Last.FM tracks *scrobbles* or user-listens, of individual music tracks. They also provide total listeners (referred to as *reach*) on a per-track, per-album, and per-artist basis, as well as user comments at every level. During the course of this project, *scrobbles* have manifested behavior that calls into question their reliability and sensitivity to gaming. However, the number of unique listeners appears more consistent. The data gathered via the screen scraper for Last.FM is up to date, and in-line with current user behavior.

4. Application Programming Interfaces (APIs)

Web Services (WS) technology, along with Web 2.0 tooling, have encouraged Internet-based products to provide APIs for the development community, which are suitable for automated agent consumption and programmatic access. Today, there are an increasingly large number of web APIs, e.g. Google's AdSense API, the Amazon's eCommerce API, or the FaceBook API, etc.

The data that these APIs provide access to has tremendous value. They also have the added benefit of being structured enough to be machine consumable with superb resiliency to user interface paradigm shifts, improvements, re-designs, etc.

4.1 When To Use APIs

With APIs, the fundamental data transport structures can remain consistent over time as the User Interfaces evolve in exciting and new directions, and the data is continuously consumed by the internet community and any automated agents that harvest it. Additionally, APIs make it possible for clients to get access to functional and aggregate data, which perhaps is not immediately available from the UI. Finally, APIs allow for targeted retrieval of data – a client can fetch only a specific data element rather than, say, a complete web page.

The rising popularity of mashups [3], i.e. the practice of combining data from multiple sources to create a new service not provided by any of the individual constituents, has made the Web API even more prominent. The benefit to User Interfaces is clear, as they effectively isolate the data access layer from the presentation layer; hiding the complexities of the backend systems. The case for APIs is further bolstered by the fact that they are also suitable for consumption by UI frameworks, which facilitates the easy creation of value-added ecosystems. While the internal APIs, which UI APIs are built upon, may be much richer than the publicly available interfaces, current architectures allow for consistent data access throughout the software stack.

4.2 Our Observations About APIs

APIs are very valuable but do not always exhibit reliable and consistent behavior. Poor documentation and quirky functionality implementations often make it difficult to take full advantage of these valuable and easy to use tools.

4.2.1 YouTube

In an effort to empower society at large and industries in particular, YouTube extended their API's to provide data extraction capabilities. However, this commendable effort came with constraints. For example, the API allows fetching of a maximum of 1000 comments for a video, and provides only lifetime aggregates of views. Also, comments are only listed in chronological order, which when coupled with the 1000-comment limit makes retrieval of recent comments, from the more spiritedly discussed videos, very difficult. The YouTube API also exhibited odd behavior when asked to return comments for a video with over a thousand comments - random (unrelated) comments were returned.

Ingesting *views* data for videos via APIs also proved to be challenging. Based on the data shown on the website, the system has the ability to track daily, weekly and all time *views*. However, the API only provides lifetime totals. A more detailed discussion on the implications of the shortcomings of APIs follows in section 5.

4.2.2 LastFM

The API provided by LastFM provided extremely useful data, but also proved to be challenging at multiple levels. Our first observation was that the value of *reach* for tracks, albums and artists sent through their API was rarely consistent with the value displayed on their web pages. Our conjecture was that this was due to caching (for performance reasons). Secondly, the semantics of *reach* was purported to be all-time number and then a 6-month rolling count. Our empirical evidence points to the latter. Thirdly, was the inconsistency within the API, where the *reach* value for a given track in one album was close, but not quite the same as the *reach* value for the same track in another album. As an example, take the artist Daft Punk's track Something About Us, the API sent the following data:

Album	Reach
"Discovery "	113957
"Discovery"	116571
"Musique Vol 1 (1993 - 2005)"	116571
"Musique Vol.1 1993 – 2005"	116571

Our fourth observation was that there was no specified update interval or period for the API numbers. Finally, minor spelling differences and or mistakes in the data exposed another interesting concern. For example, for the same artist, we have:

Album	Track	Reach
"Discovery"	Superhereos	2904
"Discovery "	Superheroes	95411

5. The Way Things Are

There are factors above and beyond the realm of technology that are having strong influences on the ways that the Deep Web is accessed.

We observed that the primary revenue generator for the business tends to be the platform with the most accurate and up-to-date data. For some businesses, their web interface is where the bulk of their money is made and APIs are viewed as useful add-ons that are paid little attention. For other firms, their business ecosystem provides the lion share of their earnings and their life blood hinges on the APIs they provide members in this community. For these companies, their user interfaces are important, but are often second class citizens.

The impact of this phenomenon is 1) there often tends to be a mismatch or incompatibility between the functionality offered by companies that support both screen scrapers and APIs (i.e. there is a feature reduction or degradation for the technology not considered vital for the business), and 2) there is often a lack of technical rigor applied to the second class citizen (e.g. data cleansing and consistency checking techniques would easily increase the functionality of these secondary technologies), which ultimately leads to their demise and or obsolescence.

Given a company's appreciation of scare resource allocation and market targeting, and also the requirement of today's user to have multiple open channels, this is an opportunity for researchers to investigate methods for deploying adaptable, reliable second class technology in some sort of staged manner, such that all access avenues are usable, which is currently not the case.

5.1 APIs vs. Screen Scrapers

User Interfaces are considered accurate, coherent and functionally rich. However, it is their feature richness and their need to change (frequently) that makes them especially challenging for automated agents to interact with them.

APIs are convenient, structured and increasingly more available. However, their current accuracy and general coherence leave a lot to be desired.

Feature	Screen Scraper	API
Accurate	Yes	No
Ease of consumption by automated agent	Complex	Simple
Semantically coherent	Yes	No
Affected by the evolution of site functionality	Yes	Not nearly as much
Affected by User Interface paradigm shifts	Yes	No

The table above provides a succinct summary of our findings. The table does not capture the facts that in terms of the time and support efforts required, access response times and time to react the pre-announcement of changes, APIs are superior to screen scrapers. In an ideal world, both of these methods may be interchangeably used, without any loss of any kind. Unfortunately, this is not the current state of affairs.

5.2 Summary

In the current Web environment, there are a lot more companies that make the bulk of their money based on Web Interfaces. Thus, presently screen scrapers work better than APIs because:

1. The popularity, and hence the revenues, of websites are dependent on what the data users see on the site's web pages.
2. Websites are extremely motivated to ensure correctness, accuracy, and consistency on the web pages shown to the end user.
3. Websites do not accord the same level of significance to the data delivered by the APIs.

The examples in section 4.2 illustrate that APIs don't work the way they should. As the problem of providing a reliable and robust API is one facing larger firms (with resources to handle this concern), then it is reasonable to assume that mid-tier and smaller companies have similar issues.

The message that keeps being repeated during implementation of the Sound Index project (and that is evident from the above examples) is that currently screen scrapers are the Deep Web access technology of choice, in spite of their susceptibility to UI changes.

However, the present trend is for most companies to provide APIs (as standard practice and operation) and while everyone is trying to do better, the effort to make them truly useful will take time. Also, as the Web evolves, the user interfaces will become more complex and sophisticated, e.g. 3D interfaces, and will require more frequent updates to appeal to the target audience.

Our position is that given the current inconsistency between APIs and screen scrapers, the fact that screen scrapers currently outperform APIs where it matters – providing reliable data – and that all the indicators signal the demise of screen scrapers, there is an opportunity for computer science research to lead the way in finding a long-term solution that allows all Deep Web access technologies to co-exist, be consistent and be well-thought of with regards to the implications on areas such as privacy policies to actual site asset protection.

6. Related Work

Utilization of the Deep Web is currently the foundation of many emerging Internet applications. We will describe the impact of our findings on several categories of these applications and briefly describe existing example applications in the category. The emphasis in this section is on APIs, as many practitioners do not use screen scraping and view it as *past its prime*.

6.1 Mashups

The full impact of limited and buggy APIs on the mashup economy is yet to be fully realized. The effects could range from skyrocketing maintenance costs of screen scrapers (to keep the essential data accurate and up to date) to the loss of valuable time spent evaluating the feasibility of using APIs, whose functionality currently falls short of their promises and expectations.

An example of a simple and relatively successful mashup is housingmaps [4], which leverages Google Maps and Craigslist to enable users to visually search for housing, via dynamic overlays of Craigslist home listings on Google Maps. Another good example is LivePlasma [5], a visually rich application that combines the Amazon API to show the relationship between movies, artists, actors, etc. Using this mashup you can go straight from interacting with the system to making purchases. Both demonstrate the success of mashups in environments where the parent companies have a vested interest in ecosystem building.

6.2 Collaboration Software

Collaboration tools and applications can greatly benefit from APIs, as well as the social communities that interact and share data through them. In this particular case, the scalability and reliability of the APIs will play a fundamental role in the effectiveness of the operational and business models.

Co-Scripter [6] is an example of a system for capturing, sharing, and automating tasks on the Web. Co-Scripter scripts contain human-readable instructions for completing Web-based processes, such as changing your mailing address or searching for real estate.

6.3 ScrAPIs

The term ScrAPI is a combination of the words "screen-scraper" and "API", and was coined by Paul Bausch in 2002 [7]. As the name suggests, ScrAPIs originally came about to support websites that did not provide an API but whose content was considered useful enough to be used in multiple applications [8]. In the face of sites with misbehaving APIs but accurate UIs, ScrAPIs present an interesting direction and opportunity by harnessing the data using screen-scrapers while enjoying the traditional benefits of APIs. However, ScrAPIs suffer from the same issues as screen scrapers and remain untenable in future.

6.4 Other

Industry wide business intelligence, compliance applications, and special interest communities are only a few of the examples of applications and social groups that would greatly benefit from rich, accurate APIs aimed at surfacing deep web data. It is also worth mentioning that these APIs are very likely to speed up the evolution of the current internet business models by enriching the current ecosystems in which they evolve and facilitating interaction and integration among entities.

7. Conclusion

Without question, the most accurate data is found today *at the glass*. Thus, screen scrapers remain the ultimate source of Deep Web data harvesting. This is due to the fact that the popularity, traffic volumes, and revenues of these web sites are driven by the content the data users see.

API data is today not nearly as functionally complete, let alone semantically coherent, as user interface data is. APIs in other contexts, such as shrink wrapped products, do have the traits of being accurate and as semantically coherent as their corresponding user interfaces. Unfortunately this is not the case today on the Web.

As more systems move toward leveraging their partners and ecosystem building becomes more dominant, technical ways to strengthen and deploy APIs (and other emerging Deep Web access technology) to provide data of the quality that screen scrapers can produce (without the volatility present due to UI changes) will become critical to the survival of the next set of Web applications.

Acknowledgments

We would like to thank the BBC, specifically Geoff Goodwin (Head of BBC Switch) for their support and encouragement. Without the efforts of Daniel Gruhl, Anna Liu and Jan Pieper, the systems development and subsequent analysis would not have been possible.

References

[1] Varun Bhagwan, Tyrone Grandison, Daniel Gruhl. "Sound Index – Charts For The People, By The People". To appear in Communications of the ACM, 2008.

[2] From Wikipedia, http://en.wikipedia.org/wiki/Screen_scraping

[3] Paul Gil. "What exactly is a Mashup?". http://netforbeginners.about.com/od/m/f/whatismashup.htm

[4] Google. Housing Maps, http://www.housingmaps.com

[5] LivePlasma, http://www.liveplasma.com

[6] Greg Little, Tessa A. Lau, Allen Cypher, James Lin, Eben M. Haber, Eser Kandogan. "Koala: Capture, Share, Automate, Personalize Business Processes on the Web." CHI 2007 Proceedings

[7] Paul Bausch. "Rambling about APIs". http://www.onfocus.com/2002/04/2717

[8] John Musser. "scrAPIs". http://blog.programmableweb.com/2006/03/21/scrapis/