# WEB SERVICES FOR MUSIC INFORMATION RETRIEVAL

*Mark Zadel and Ichiro Fujinaga*
Faculty of Music
McGill University
Montréal, QC H3A 1E3
{zadel,ich}@music.mcgill.ca

## ABSTRACT

In the emerging world of networked and distributed digital libraries, the Web services framework will be a key to facilitating simple inter-application communication between them. Yet, despite the popularity of Web services in the business sector and their seemingly obvious applicability to the digital library domain, and to MIR in particular, the adoption of these new protocols has not been widespread.

To demonstrate the tremendous potential of Web services for MIR, this paper presents an application using the Google and Amazon.com databases to generate clusters of related musical artists based on cultural metadata. The use of cultural metadata to determine artist relatedness is valuable and interesting because it captures emergent popular opinion about music. Starting from an initial seed artist, Amazon *Listmania!* lists are traversed to find potentially related artists. Google is used to determine which of these candidates are in fact related by assessing the co-occurrence of the two artists' names on Internet web pages. A list of artists related to the seed is returned once a given number of artists is found.

The positive results generated by the system illustrate the use of Web services for exploiting the vast amount of untapped data that are available today and highlight their importance for the future, when even more musical data will become available.

## 1. INTRODUCTION

Over the last few millennia, humans have amassed an enormous amount of information and material that is scattered around the world. It is becoming abundantly clear that the optimal path for creating useful sources of information is to distribute the task of digitizing the wealth of historical and cultural heritage material that exists in analogue formats. These may include books, manuscripts, music scores, maps, photographs, videos, analogue tapes, and phonograph records. In order to achieve this goal, libraries, museums, and archives throughout the world, whether large or small, need well-researched policies, proper guidance, and efficient tools to digitize their collections and to make them available economically.

The topic of this paper is to suggest an answer to the question of how to access and retrieve the data once they are stored. This problem arises even for new digitally-born materials.

Thousands of libraries worldwide cannot be expected to agree on the same database or query systems to access their data. This paper investigates the possibility of using Web services, an emerging technology which is designed to exchange information between different systems, to address this issue.

The use of Web services is becoming increasingly popular in business environments. Web services allow easy interoperability between disparate computer systems, which has historically been difficult. This recent popularity has prompted many businesses to add public Web services interfaces to their databases, allowing direct programmatic access to them. The alternative for accessing these data is to extract the information from browser pages intended for human viewing, known as "web scraping." This approach is brittle and error-prone as small changes in page formatting can break the extraction algorithm.

Cultural metadata is information describing public opinion and cultural trends, distilled from large amounts of unstructured text produced by the public. This text is typically drawn from the web. Recent MIR research has used cultural metadata to assess similarity between musical artists [3][14]. This approach has the advantage of using current cultural information to make judgments about genre and similarity, and does not rely on centralised, and potentially biased, systems of classification. Community-based, collaborative systems for classification and filtering have been explored previously [6][11][12]. These systems make recommendations to a user based on the opinions of others who have demonstrated similar tastes. They require that users explicitly evaluate material and use this information to try to predict the material's relevance to other like-minded users. Using cultural metadata to capture the same information eliminates the explicit extra effort required by these systems.

While the web can be seen as a direct reflection of public opinion, its size makes it difficult to harness and exploit. This issue has received significant attention, and now web searching systems have been successfully scaled and improved such that they are able to cope with the large amounts of data available. An important example is Google, a very powerful and popular search tool which effectively manages this mass of content. Previously, these systems were only accessible via browser interfaces, but now Web services allow them to be used directly. Suddenly, the unstructured cultural information contained in these databases is easily and reliably accessible to external computer programs and, specifically, to MIR programs.

This paper will demonstrate the potential of Web services through an application that exploits their recent development and highlights the relevance of Web services and Internet resources to MIR. The application addresses the problem of assessing artist relatedness using only cultural metadata. This approach should yield more relevant classifications of artists than previously possible since it is based on large amounts of current public opinion. Here, two artists are considered “related” if their names co-occur on the same web page.

The application makes use of the Amazon.com and Google databases in tandem, and assesses artist relatedness by measuring the co-occurrence of artist names on web pages. Co-occurrence analysis for music classification has been explored in [1][7]. These papers use web crawling to retrieve relevant web pages, which are parsed and analysed. The application presented here differs in that it examines all web pages, not just a predefined subset of them, and it does so directly through Web services interfaces. The rest of the paper is organized as follows. In Section 2 we introduce Web services and their components, including the services implemented by Google and Amazon.com. In Section 3 we describe a sample application that uses these two databases, and the results of some experiments using the application are shown in Section 4. In Section 5 we discuss and analyze the experiment, and we conclude in Section 6.

## 2. WEB SERVICES

Web services allow inter-application communication over a network. They adhere to a well-documented standard [4] and are strongly supported by industry and the primary web standards organization, W3C. The standard is designed to be lightweight and platform agnostic, allowing communication between broad classes of devices (e.g., cell phones and Sun workstations). Web services are based on common, accepted standards; this ultimately facilitates their implementation and adoption. They are implemented in all major languages commonly used for network programming (an example in Python is given in Figure 1). The technologies on which the Web services architecture is based include: Simple Object Access Protocol (SOAP); Web services Description Language (WSDL); and Universal Description, Discovery, and Integration (UDDI).

The core technology in Web services is XML, a common standard for data representation. It is simple and human-readable, and is ubiquitous in contemporary computing. A large software infrastructure exists for working with XML data. SOAP, based on XML, is used as the standard message-encoding format. Messages between computer applications are encoded as SOAP messages and sent via HTTP (see Figure 2). Responses are similarly encoded and are returned to the requestor.

Also based on the XML format, UDDI is used to register each institution’s services (functioning as a virtual yellow pages directory). WSDL is used to describe the type of service, its access protocol, and its location. Thus, one uses UDDI to look for services or data, and WSDL to find out how and where to use the service, all via SOAP messages.

As Web services are generally applicable to system interoperability and access, they have obvious relevance to MIR research. The general unstructured information on the web can be harnessed (as is demonstrated in this paper), and MIR databases can be made to communicate and integrate each other’s resources, as well as a host of other applications.

The attractiveness of this technology for application in distributed digital libraries is that it assumes that each system (library) will be different. Web services provide users with the “what, where, and how” required to access information from heterogeneous systems. These types of universal methods for finding out how to access various archives and collections, each with different database languages and different kinds of services, are not only useful but will become more and more essential as archives and libraries around the world begin to convert their collections into digitally accessible formats.

Despite their tremendous potential and significant industry activity, Web services have not been used widely in the digital library domain. The open-source digital library management system Fedora is one of the few exceptions. It promotes the distributed digital library architecture through interoperable access to digital objects and communication across the Internet based on Web services [8].

As mentioned, industry interest has resulted in a great deal of development of Web services and related standards. Although in its infancy, some early adopters are successfully using the technology, demonstrating its potential. Two such examples are Google and Amazon.com.

### 2.1. Google Web Services

Google is by far the most popular search engine currently on the web. It maintains a massive database with a potential wealth of information. Google provides access to its database via its Web services API (application programming interface) [5]. The interface allows programmatic access to standard Google searches, cached page retrieval, and spelling suggestion searches. All of the information returned in a typical Google query done by hand is encoded in the response: search results, page snippets, result

```
import SOAP

server = SOAP.SOAPProxy("http://services.xmethods.net/soap/servlet/rpcrouter")
print server._ns("urn:xmethods-Temperature").getTemp(zipcode="90210")
```

**Figure 1**. A complete Python program for getting the current temperature in a U.S. zip code region using SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
    <SOAP-ENV:Body>
        <ns1:getTemp xmlns:ns1="urn:xmethods-Temperature" SOAP-ENC:root="1">
            <zipcode xsi:type="xsd:string">90210</zipcode>
        </ns1:getTemp>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**Figure 2**. The raw XML SOAP request produced by the example in Figure 1

counts, etc. This allows Google data to be programmatically queried and the results aggregated to identify patterns and trends. (See Figure 3 for an example and see Section 3 for its explanation.)

A free license key must be obtained to use the service, which must be included in each request. Up to 1000 queries may be made per day with a given key.

## 2.2. Amazon Web Services

Amazon provides programmatic access to its database via Web services as well [2]. A wide variety of queries are accepted: product number (ASIN), musical artist, author, film director, manufacturer, etc. Queries return detailed lists of products offered for sale, which include product name, author, availability, manufacturer, price, release date, ISBN, etc. The database is large and well organized.

Amazon provides various ways for customers to rate and review sales items. One method is *Listmania!*, allowing Amazon users create lists of their favourite or related items. A product page will include links to lists that reference it to help consumers find other products they might be interested in. These lists can be queried via Amazon Web services as well.

Like the Google Web API, Amazon Web services require a free license key which must be included with each request. Amazon asks that clients generate only one query per second, and that search results are cached locally.

## 3. A SAMPLE APPLICATION

As an example of the potential application of Web services to music information retrieval, a program was written that uses the Google and Amazon Web APIs to access information that would otherwise be difficult to obtain. The following problem was posed: given an initial seed artist, generate a list of related musical artists based on cultural metadata. The main advantage in using this approach to determine relatedness is that it relies on current trends in public opinion, and not on a centralized, top-down approach. These lists of related artists could be used for recommendations, or to track genre evolution over time.

The Google and Amazon databases contain large amounts of information contributed by the general public. Google indexes the World Wide Web and Amazon's *Listmania!* lists are contributed and edited by Amazon users. The example application accesses this information via Web services and uses it to address the above problem. The following assumptions are made: artists who appear on the same *Listmania!* list are likely to be related, and related artists' names are more likely to appear on the same web page than if they were not related. Only information generated by the general public was used in this experiment.

The program works as follows. Starting from a seed artist, a list of potentially related artists is generated from Amazon data. The pool of potentially related artists is grown recursively from this initial list, finding artists related to *those* artists, etc. This tree is pruned using a relatedness metric based on Google data. Once some given number of related artists has been reached, the program terminates.

Amazon *Listmania!* lists are used to generate pools of potentially related artists from an initial one. First, the initial artist's releases are each queried in the Amazon database. For each of these releases, a set of *Listmania!* lists are returned which include the release. These lists are each queried, and the artists included in each list are returned. Thus, given an initial artist, we can generate a pool of potentially related ones.

Google results counts are used to assess the actual relatedness of two artists. Three queries are done, and the results count is retained from each: *"artist1"*, *"artist2"*, and *"artist1" "artist2"* (where *artist1* and *artist2* are replaced with the names of the artists being compared). Each

```
# return a scalar that measures the relatedness between
# a pair of artists (artist1,artist2)

def google_relatedness(artist1,artist2):

# enclose artist names in quotes
    artist1 = ’"’ + artist1 + ’"’
    artist2 = ’"’ + artist2 + ’"’

# find the ratio of the intersection to the smaller of the two sets
    artist1count = get_google_results_count(artist1)
    artist2count = get_google_results_count(artist2)
    combinedcount = get_google_results_count(artist1 + ’ ’ + artist2)

    relatednessmeasure = float(combinedcount)/min(artist1count,artist2count)

    return relatednessmeasure
```

**Figure 3**. Sample Python code to query Google for two artists

name is enclosed in double quotes to ensure that it is considered atomically. A single scalar value is computed for relatedness according to:

$$\frac{\text{results}(\mathit{combined})}{\min(\text{results}(\mathit{artist1}), \text{results}(\mathit{artist2}))}. \quad (1)$$

Thus, the relatedness is considered to be the percentage of pages for a given artist that include the other artist’s name (see Figure 3). The minimum is used to correct for the difference in popularity of the artists. This metric, although simple, provided reasonable results.

Thus, we generate a list of potentially related artists and eliminate the ones that are deemed unrelated according to the above Google relatedness metric. The remaining (related) artists are queried recursively. Relatedness is always measured with respect to the original seed artist specified in the first iteration.

### 3.1. Implementation Details

The application was implemented in Python. Python was well suited to this experiment since it features a wealth of high-level modules and lends itself to rapid development. In particular, modules specifically designed for interfacing with the Google Web API and Amazon Web services are available [9][10]. Their existence speaks to the ubiquity of the technologies Web services are built on; Python modules handling XML are included in the standard distribution, and modules handling SOAP messaging are readily available [13].

The application works by tracking the set of artists left to examine. Initially, the set only contains the seed artist. One artist is taken from the set, and a set of artists potentially related to that artist are generated by querying Amazon data as described above. Artists are removed from this set that are deemed unrelated to the seed artist according to the Google relatedness metric. A relatedness threshold of 0.05 was set empirically, above which artists were considered related. The remaining artists are retained as being related to the seed artist for final output, and are added to the set of artists left to examine. This process is repeated until the set of related artists reaches a given size.

The Google results counts are cached locally in a Python *dict* variable which is saved between application invocations. This reduces the number of calls into the Google database, reducing network traffic and saving queries associated with the license key. This is especially important since various queries are re-executed often (e.g., searches for the seed artist, multiple executions of the program). A wrapper function (`get_google_results_count`) is used which first checks the local cache before calling into the Google database. Similarly, sets of artists and *Listmania!* lists already checked are maintained, and no artist or list is checked in the Amazon database twice. These caches can be safely assumed to be valid over one or two days.

Using Python proved to be a sensible decision for this project for the reasons mentioned above. The implementation is clean and lightweight. Elegance of implementation was prioritised over computational efficiency as the script runtime was dominated by the time spent waiting for query responses

## 4. RESULTS

The program was tested starting from a variety of seed artists, with varying styles and popularity. The seeds used were Four Tet (electronic), Slayer (heavy metal), Christina Aguilera (dance pop), Kenny G (adult contemporary), and John Coltrane (jazz). In each case, the program successfully returned a list of artists that seem reasonably related to the seed artist. For each artist the entire process takes an average of about two minutes. The artists were typically ordered by genre, but some crossover occurred. This is to be expected, as cultural metadata encapsulates more than simple genre. Example results for Christina Aguilera are shown in Tables 1 and 2, which show 40 of a total of 168 artists returned. The results for John Coltrane are shown in Tables 3 and 4. Overall, the results are positive: solo female pop artists and mainstream jazz artists and groups are ranked highly, and there are no glaring aberrations. The last twenty entries are smaller artists or are from different

| Artist | Google Relatedness |
|---|---|
| Christina Aguilera/Lil Kim/Mya/Pink | 1 |
| Aguilera | 0.905683192 |
| Britney Spears | 0.721886336 |
| Sisqo | 0.691678035 |
| Spears | 0.684401451 |
| Toni Braxton | 0.658564815 |
| Willa Ford | 0.632508834 |
| Emma Bunton | 0.498381877 |
| Susan Tedeschi | 0.480686695 |
| Patricia Manterola | 0.461621622 |
| Melanie C | 0.425213675 |
| BBMak | 0.419417476 |
| Jennifer Lopez | 0.396614268 |
| Lil' Bow Wow | 0.395973154 |
| Billy Gilman | 0.391975309 |
| Faith Hill | 0.374774775 |
| Michelle Branch | 0.353571429 |
| Justin Timberlake | 0.350148368 |
| O-Town | 0.347962382 |
| Madonna | 0.330108827 |

**Table 1**. Top twenty results for Christina Aguilera

| Artist | Google Relatedness |
|---|---|
| Rosemary Clooney | 0.033737864 |
| Phoebe Snow | 0.032386364 |
| Amel Larrieux | 0.03168 |
| The Go-Go's | 0.030064935 |
| Rowland | 0.029382304 |
| Muse | 0.027569528 |
| Tim Pierce | 0.022851920 |
| Lynda | 0.021138211 |
| Heitor Pereira | 0.018274112 |
| Martin Tillman | 0.017647059 |
| Various Artists | 0.017533253 |
| Johnny Mori | 0.016697588 |
| Anahi | 0.015185950 |
| Julia Migenes | 0.010829493 |
| Elastica | 0.008759036 |
| Interpol | 0.007124464 |
| Various Artists | 0.006873239 |
| Bruce Fowler | 0.004733728 |
| Craig Eastman | 0.002016129 |
| Michael Fisher | 0.001507937 |

**Table 2**. Last twenty results (out of 168) for Christina Aguilera

| Artist | Google Relatedness |
|---|---|
| Miles Davis | 0.413888889 |
| McCoy Tyner | 0.393333333 |
| Miles Davis Quintet | 0.391872279 |
| Sonny Rollins | 0.373853211 |
| Thelonious Monk | 0.371115174 |
| Dexter Gordon | 0.365187713 |
| Cannonball Adderley | 0.361276596 |
| Horace Silver | 0.358108108 |
| Hank Mobley | 0.352231604 |
| Charles Mingus | 0.350877193 |
| Ornette Coleman | 0.333221477 |
| Art Blakey | 0.311858974 |
| Dizzy Gillespie Quintet | 0.286738351 |
| Rahsaan Roland Kirk | 0.283450704 |
| Wayne Shorter | 0.280924855 |
| Kenny Burrell | 0.277922078 |
| Sonny Clark | 0.276315789 |
| Peter Brotzmann Octet | 0.276119403 |
| Johnny Griffin | 0.266968326 |
| Brown | 0.251851852 |

**Table 3**. Top twenty results for John Coltrane

| Artist | Google Relatedness |
|---|---|
| Ryan Adams | 0.014444444 |
| Compay Segundo | 0.014344262 |
| Prince & the Revolution | 0.014089347 |
| Philip Glass | 0.013927227 |
| Queens of the Stone Age | 0.012962963 |
| Fugazi | 0.012472648 |
| Helmet | 0.012222222 |
| Fiona Apple | 0.011153342 |
| Joseph Arthur | 0.010578947 |
| Doves | 0.010555556 |
| Gregorian Chant | 0.009694444 |
| Morello | 0.009017857 |
| John Adams | 0.007777778 |
| Howie Day | 0.006848249 |
| Herring | 0.006805556 |
| Damien Rice | 0.006228669 |
| Interpol | 0.006120370 |
| Ibrahim Ferrar | 0.005325444 |
| Franky Perez | 0.002735562 |
| Spottiswoode | 0.001424051 |

**Table 4**. Last twenty results (out of 139) for John Coltrane

genres. Even with such a simple relatedness calculation, the system performs very well.

A number of problems are apparent, however, and this example illustrates some of these issues. Some of the "artist names" do not correspond to actual artists (e.g., "Various Artists" and "Christina Aguilera/Pink/Lil Kim/ Mya"). Names in Amazon's database are case sensitive, and are sometimes duplicated with different capitalization. Also, artists appear multiple times with different names (e.g., "Britney Spears" and "Spears"). These are all issues with inconsistency in the Amazon database.

## 5. DISCUSSION

Artist relatedness is doubly enforced in this experiment, through proximity in both the Amazon and Google databases. It is not clear whether either method on its own would produce good results. Amazon *Listmania!* lists by definition group related items, but not necessarily in ways that one would expect (e.g., "Really terrible music from the eighties," "Artists born on a Tuesday"). Similarly, the Google relatedness metric performs well when used on a pool of potentially related artists as given by the Amazon search. It is not clear how it would perform if used on a randomly selected pool of artists.

The Google metric is simple and should be subjected to more experimentation. One observation was that the threshold used (0.05) seemed too low for a large artist like Christina Aguilera. It stands to reason that very popular artists will be talked about more, and perhaps the size of the results sets for both artists should be incorporated into the relatedness calculation. Also, more queries or better query strings could be used with the Google database to help focus the results. For example, searches could be restricted to a particular range of dates, or to a particular country. The results counts returned by Google are often only estimates; an investigation should be done to see whether this has an adverse effect on the relatedness measurement.

Some artists use names that are common parts of speech or are associated with other things (e.g., "Muse," "Brown"). This will cause these artists to be ranked lower than they would have been otherwise since the extra results will count toward the total. This effect is less likely to distort the co-occurrence count since if the two names appear on the same page, they are both likely to refer to musical artists. Due to the size of the web and the Google database, the effects of these exceptions will be attenuated in the large overall number of pages.

The automated access to this vast amount of data is the key to the use of Web services. Here we only used two sources, but theoretically one could automate access to hundreds of distributed databases over the Internet. Even with this simple and modest example using two databases, rich source of relationships between artists and their works can be retrieved. The important point in this experiment is that Web services allow us to conveniently access these resources, and that they contain large amounts of extremely valuable information.

## 6. CONCLUSION

Web services carry tremendous potential for MIR research. Coupled with business initiatives on the Internet, they allow easy access to the mass of both unstructured and structured information on the Web. The sample application presented here takes advantage of these publicly accessible resources to generate groups of related artists based on a reflection of general public opinion. It quantifies relatedness as a function of the frequency of co-occurrence of the artists' names on web pages, which would be difficult without Web services access to the Google database. This approach is valuable since it encapsulates current public opinion. The results were generally promising, correctly clustering related artists. They prompt further exploration, and some potential refinements have been suggested here. The application serves as an example of the promise of Web services for MIR, and suggests further research into applications of these newly accessible resources.

## 7. REFERENCES

[1] Aucouturier, J.-J., and F. Pachet. 2003. Representing musical genre: A state of the art. *Journal of New Music Research* 32 (1): 83–93.

[2] Bauche, P. 2003. *Amazon hacks*. O'Reilly and Associates.

[3] Baumann, S., and O. Hummel. 2003. Using cultural metadata for artist recommendations. *Proceedings of the International Conference on WEB Delivering of Music*, 138–41.

[4] Booth, D., H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. 2004. *Web services architecture*, W3C Working Group Note, available online: `http://www.w3.org/TR/2004/NOTE-ws-arch-20040211`

[5] Calishain, T. and R. Dornfest. 2003. *Google hacks*. O'Reilly and Associates.

[6] Hill, W., L. Stead, M. Rosenstein, and G. Furnas. 1995. Recommending and evaluating choices in a virtual community of use. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 194–201.

[7] Pachet, F., G. Westermann, and D. Laigre. 2001. Musical data mining for electronic music distribution. *Proceedings of the International Conference on WEB Delivering of Music*, 101–6.

[8] Payette, S., and T. Staples. 2002. "The Mellon Fedora project: Digital library architecture meets XML and Web Services." Sixth European Conference on Research and Advanced Technology for Digital Libraries. *Lecture Notes in Computer Science*, Vol. 2459. Berlin: Springer-Verlag, 406–21.

[9] Pilgrim, M. 2004. PyAmazon. Available online: `http://josephson.org/projects/pyamazon`

[10] Pilgrim, M. 2004. PyGoogle. Available online: `http://pygoogle.sourceforge.net`

[11] Resnick, P., N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 175–86.

[12] Shardanand, U., and P. Maes. 1995. Social information filtering: algorithms for automating "word of mouth". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 210–7.

[13] Ullman, C., and B. Matthews. SOAPpy, 2004. Available online: `http://pywebsvcs.sourceforge.net`

[14] Whitman, B., and S. Lawrence. 2002. Inferring descriptions and similarity for music from community metadata. *Proceedings of the International Computer Music Conference*, 591–8.