

# JWEBMINER REFINEMENT FINAL PROJECT REPORT

**Gabriel Vigliensoni**

Music Technology Area, Schulich School of Music, McGill University

`gabriel@music.mcgill.ca`

## ABSTRACT

jWebMiner is an open-source software framework to extract cultural metadata from the web through the use of web services. Our refinement allows jWebMiner to retrieve information from additional web services. We selected a popular music-listener community to extract information from it. Furthermore, we provided jWebMiner with a set of text filters and site weightings to improve its classification accuracy when querying and hit counting the whole network. A number of experiments in genre classification with a ground-truth dataset was performed. We obtained 5.2% and 16.7% better accuracy classification results for 5 and 10 genres classification and realized that the music community information is very useful under certain constraints.

## 1. INTRODUCTION

The amount of digital media and information about it stored in the web is growing exponentially. Data mining researchers study the extraction of patterns from this data and how to classify and organize it to provide better ways to search, query, and access it. In the music field, low-level, high-level, and cultural features can be extracted to classify and organize musical information.

jWebMiner is an open-source software framework that extracts cultural metadata from the web through web services, allowing MIR researchers to use semantic information about music provided by thousands of anonymous users. These cultural features can complement low-level and high-level features extracted from audio signals and musical abstractions to classify and categorize music more accurately and efficiently.

However, jWebminer has not been updated since its version 1.0, and some key features such as the communication with Google through SOAP protocol is no longer possible because the search engine changed its API to AJAX. Furthermore, some inconsistencies have been detected between values provided directly by the Yahoo! search engine and the jWebminer data acquisition. Additionally, a

tested dataset of text strings would be useful to iter and gain accuracy in the results.

The main objective of this project was to improve the functionality and reliability of jWebminer providing additional web services and a set of text filters and site weightings.

## 2. JWEBMINER

jWebMiner is part of the jMIR bundle allowing the extraction of cultural metadata using web services. Basically, it counts hits from different search engines using text strings as a query input (or iTunes XML, ACE XML, and Weka ARFF text files). The number of hits can express the co-occurrence between different text strings (i.e., how many times a text string appears in pages that have another text string), or the cross tabulation between two sets of text strings (i.e., how often a text string of one class appears in text strings of other class) (McKay et al. 2007).

### 2.1 WebServices

Web services allow machine interoperability interaction over a network between disparate computers in a very simple way (Zadel et al. 2004). Through the use of this technology, jWebMiner is capable to communicate with web services providers, query text strings, and acquire hit counts from the net. Among the growing quantity of servers providing access to its API's, there are search engines providing keys to authenticate as users, allowing access to part of their information and features. In general, each web service specifies some use constraints in its *Terms of Service* to control the maximum daily number of requests by IP number, or maximum number of request in a time-period.

### 2.2 Filtering and Weighting

In order to provide flexibility for researchers and users, jWebMiner gives the possibility to configure text filters and site weights. Hence, it is possible to filter hits that does not have any user-definable text strings, weight selected websites in a bigger or smaller degree depending in the user ability to sense how much important is public opinion in one site or another, and use different string searches to look for terms that common people use as synonyms when refer to music. Additional filters for language, region, and filetype, are offered to narrow the search to some given characteristics and do not count redundant results.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.

© 2009 International Society for Music Information Retrieval.

### 2.3 File Output

jWebMiner also offers the possibility to change its statistical functions and the way data is normalized in order to research best ways to extract, weight, and see information. To integrate results in other software environments, the output data possibilities include ACE XML, Weka ARFF, HTML, and text files.

### 2.4 Extensibility

jWebMiner is downloadable as standalone or developer version. The last one allows the possibility to configure or develop new implementations such as the addition of further web services, configure the output visualization, or any special feature or characteristic needed for a project through the extension or change of its application program interface (API).

### 2.5 Search methods: cross tabulation and co-occurrence

jWebMiner offers two different ways of looking for the joint distribution of text queries, *co-occurrence extraction*, and *cross tabulation extraction*. In the latter, the user must type a set of strings in the primary and secondary fields (e.g., artist names and musical styles), and let the system run, querying the selected web services (Yahoo! and/or Google in version 1.0) and measure the cross tabulation of values in different fields. Although the process is simple, the number of search queries could reach very fast the limit of a specific web service (e.g., Yahoo! has a maximum of 5000 daily queries by unique IP). This will limit the possibilities to a set of five hundred text entries classified in ten different classes. If filtering is applied, the number of queries would be even smaller.

In the second mode, the system measure the co-occurrence of each value in one field with other values in the same field, doing  $\frac{1}{2}(n(n+1))$  different queries. Therefore, if we want to extract cultural metadata using the Yahoo! search engine in co-occurrence mode, our data set should be limited to 99 different entries in a daily/IP basis.

### 2.6 Search engines

In addition to the daily/IP search limits given by Yahoo!, Google is no longer providing license keys since 2006, and changed its API from SOAP to AJAX. Furthermore, current Terms of Service prohibit parsing information out of their web interface. Hence, presently there is no way to get automatized information via the Google API.

## 3. FIRST EXPERIMENTS

In our first experiment, we wanted to observe the behavior of jWebMiner in a simple genre classification test. The selected dataset was small in order to perform fast queries and do not reach the Yahoo! limit. It consisted in the artists of the twenty most important popular music albums in the opinion of the *Rolling Stone* magazine website editors, and

nine music styles (*jazz, reggae, grunge, rock, punk, classical, soul, R&B, and pop*). Some artists had two or three recordings in the list (e.g., The Beatles), so there were only fourteen artists in the list.

Our first attempts using jWebMiner were not as good as expected. First, the stand-alone distribution provided in the jMIR website did not run in OSX machines, so to perform our first experiments we had to run it in a Windows machine. Second, the classification accuracy results obtained by jWebMiner using simple and complex text filters, and simple and complex site weightings seemed to be very erratic. When were shared and analyzed with professors and graduate students, there was consensus that the extracted data or the scoring function had something wrong. Table 1 shows the genres mined from the web for every artist with different filtering and site weightings.

	NC	SF	SW	CW	CF
Bob Dylan	Ja	Ja	Re	Re	Gr
Bruce Springsteen	Ja	Ro	Re	Pu	Pu
Elvis Presley	Re	Re	Ro	Cl	Ro
Jimi Hendrix	Gr	Gr	So	Cl	Ro
Marvin Gaye	So	So	So	Cl	Rb
Michael Jackson	Rb	Rb	Re	So	Rb
Miles Davis	Ja	Ja	Ja	Ja	Ja
Nirvana	Gr	Gr	Gr	Gr	Gr
The Beach Boys	So	So	Po	Cl	Rb
The Beatles	Cl	Cl	Ro	Ro	Po
The Clash	Pu	Pu	Pu	Rg	Rg
The Rolling Stones	Rg	Rg	Pu	Cl	Ro
The Velvet Underground	Pu	Pu	Pu	Pu	Rb
Van Morrison	Ja	Ja	So	Cl	Rb

**Table 1.** Genre classification under no constraints (NC), simple and complex filters (SF, CF), and simple and complex weightings (SW, CW)

From the classification results given by jWebMiner, we observed that several different genres were mined for some artists almost erratically (e.g., Bruce Springsteen as *jazz, rock, reggae, or punk*, and The Beach Boys as *soul, pop, classical, and R&B*). Moreover, unarguable misclassification were made (e.g., Bob Dylan as *jazz, reggae, or punk*).

When we analyzed the raw hit counts acquired by the software, we realized that the numbers of stated a more *normal* behavior. In other words, the number of pages with the terms *Bruce Springsteen, Bob Dylan, The Beach Boys, and Rock* were bigger than for the other genres. This classification can be discussable but absolutely it is more common-sense.

To understand and solve these issues a step-by-step refinement methodology was made.

## 4. REFINEMENT

### 4.1 OSX compiling, libraries updating and linking

The downloadable developer version of jWebMiner comes with the source files, libraries, and NetBeans project. Our first goal was to compile the software to run in the OSX operating system. Doing that, we realize that the libraries were not correctly linked (maybe the program first version was fully tested only for windows, and/or OSX and Java have had many changes from 2007 to 2009), so they were re-linked in order to properly compile the software. In addition, a version 2.9.1 of the *Xerces* library for parsing, validating, and manipulating XML documents was downloaded and upgraded.

### 4.2 Result scoring

To understand the misclassification error, we manually searched and extracted the acquired Yahoo’s hits numbers of our first experiment in order to compare with what jWebMiner had extracted. The results were almost identical for all cross validated variables. As the hits counts were right, the scoring function may had some problems. We analyzed the following cross validation scoring function used in jWebMiner and posed in (Geleijnse et al. 2007).

$$S(a, g) = \frac{co(a, g)}{1 + \sum_{b \in A} co(b, g)}$$

Being  $a$  and  $b$  two *primary search strings* (i.e., artist name), and  $g$  the *secondary search string* (i.e., musical genre). The scoring function allows to level all genres with the same number-base. As the term *rock* appears 26 times more than *grunge* in the experiment requested pages, all grunge artists were classified as *rock*. In fact, *Nirvana* and the query *'rock'* returned 42 millions of hit counts, while *Nirvana* and *'grunge'* returned only 6.02 million hits. However, jWebMiner classified *Nirvana* as *grunge*, which we could convey that it is correct. Hence, the scoring function was doing its job, normalizing—weighting—all genres as if they appear in the same quantity of pages. Table 2 shows the weights of the different genres related to *rock*, which was the biggest one in term of hits count.

Genre	Weight Factor
Rock	1
Pop	1.358
Soul	2.5472
Jazz	2.6588
Punk	3.7504
R&B	4.7006
Reggae	5.6237
Classical	5.8864
Grunge	26.3606

**Table 2.** Weight factor for each genre in the scoring function in our first experiment.

What this method provides, is a way to show genres that are under-represented. However, these *little-genres* tend to over-represent themselves. Hence, the scoring function was responsible for many of the *noisy* classifications that we did not understand at a first glance (i.e., that was the reason for which *The Rolling Stones* were classied as *reggae*, *punk*, or *classical*)

As a first learn, doing a genre classification in which there is no valid artists for each one of the styles, will lead the scoring function to overweight too much that genres. So, for every genre in the list must be a number of artists belonging to it (is the same for the opposite example, all artists must belong to a genre in the list, otherwise the system will misclassified it). A second and complementary approach could be the use of a threshold or curve for the weighting factor, allowing us to control how much we want to emphasize little genres.

To continue our refinement with jWebMiner, we decided to test it with a ground truth dataset, allowing us to compare classification accuracy performance and new experiments with previous ones under the same conditions.

### 4.3 Testing jWebMiner with ground truth

The SAC (Symbolic, Audio and Cultural) dataset was assembled by Cory McKay and Ichiro Fujinaga in 2008 to provide matching symbolic, audio and cultural data for use in experiments of combining data for artists and songs genre classification. Although the dataset consists of 250 files (MIDI and audio, as well as metadata for each track), we used only 131 unique artists files because the original dataset has more than one song for some of the artists. The SAC dataset is divided into 10 different genres, consisting of 5 pairs of similar styles, allowing to perform 5-genre or 10-genre classification for the same set of artists. Weighted and un-weighted classification accuracy rates were measured for the 10 genre classification, determining how effective was the classification, but also how serious the misclassifications were, providing insight on error types (McKay et al 2008).

### 4.4 New web services

To update web services that jWebMiner could use, we look for API’s offering different methods and constraints. The following API’s were reviewed: Last.FM API, Billboard API, and the Google AJAX API. The new Amazon web services—AWS—were discarded because they evolved to a commercial service.

Last.FM is a music service working since 2003 for the discovery of music powered by its community of listeners. It allows users to create personal profiles and contribute *social tags* for each one of the songs and artists that they *scrobble* (i.e., it automatically add the tracks played and tagged by the users in different media players and environments to Last.FM with a piece of software called Scrobler), sharing information and connecting people with similar tastes (Turnbull et al. 2008). The amount of information managed by Last.FM is enormous, summing more

than 36 billion tracks scrobbled, and augmenting 321 millions more each week<sup>1</sup>. They provide free access to portion of the data through the Last.FM API, allowing developers and users to build their own applications and programs using different *methods*. For our development we used the *artist.getTopTags* method, which allows to retrieve Last.FM top tags for artists, ordered by popularity. Using this method, we will have access to know how common people—final users—classifies different artists. Although we know that jWebMiner was originally designed to count hits from search engines, we wanted to extract user information from a *music community* to compare and complement the one acquired from search engines.

The Billboard is one of the world’s most important popular music publication. The charts information it collects had served the music and media industry for more than 100 years. Nowadays is a fundamental source of information on trends and innovation in popular music. Although it seemed to be a very powerful data resource, the methods provided nowadays only refers to music charts.

The Google search engine, which originally provided access to its API via SOAP to query the web through automated processes, changed its protocol and access information politics. The search engine migrates to a *Google AJAX Search API*, which allows to incorporate a search box in a website, but does not allows to access information through automated means (such as scripts or web crawlers), as established in its Term of Service<sup>2</sup>. These restrictions make its use impossible into jWebMiner.

#### 4.5 Text Filters, Synonyms and Site Weighting

In order to improve the classification accuracy we designed and tested text filters for the *Required Filter Words* and *Excluded Filter Words* fields. Although in (Geleijnse et al. 2006) and (Whitman et al. 2002) some word constraints were recommended (such as *music*, *review*, artist *x* played style *y* *music*), we empirically obtained better results with the words *mp3* and *store* as excluded filter words.

To broaden the search possibilities, finding different ways people express about music genres, and artists names, we develop a set of synonyms for different styles and name of artists. Thus, jWebMiner could query the term *Bop* as synonym of *Bebop* and *Be-bop*. On the other hand, an artist such as *Derek and the Dominos* could be found as *Derek and the Dominoes*.

The site weightings were tweaked in order to find the best performance and accuracy. Three possibilities were investigated: querying the whole network as raw hit counts, only three *manual annotated sites* (*wikipedia.org*, *allmusic.com*, and *amazon.com*) with a value of 0.333 for each one, and a mix of the first and second, giving the whole network a weight of 0.5 and 0.167 for each one of the three sites.

## 5. RESULTS AND DISCUSSION

The average classification accuracy rates using cross tabulation for each one of the experiments in 5 genre classification and previous results with the same dataset are shown in Table 3.

MK08	NC	F/W1	F/W/S	Last.FM
87.2	82.4	90.1	92.4	93.9

**Table 3.** Average classification accuracy rates for each experiment in 5 genre classification. The code-terms for each experiment are *MK08* (McKay et al. 2008 experiment), *NC* (no constraints), *F/W1* (filtering and weighting 1), *F/W/S* (filtering, weighting and synonyms), and *Last.Fm*

It can be seen that the *NC* experiment classification accuracy was worse than *MK08*. However, applying the proposed filtering, weighting and synonyms, we observe improvements of 2.9% and 5.2% over that mark, and 7.7% and 10% over *NC*. However, the best classification accuracy was achieved with the Last.FM social tags. This result led us to think that data extracted from music related communities sites give better results than extracting information over the whole internet.

For 10 genre classification, we performed almost the same experiments, but we separate synonyms from the filters and weighs to understand how each one behave. Furthermore, the three different choices of site weighting explained above were used. Table 4 shows the accuracy rates for weighted *W* and unweighted *UW* results.

	MK08	NC	SYN	F/W1	F/W2	Last.FM
UW	61.2	56.5	50.4	67.2	77.9	43.5
W	67.4	63.7	51.9	76.7	82.8	43.9

**Table 4.** Average classification accuracy rates in 10 genre classification, weighted and unweighted

We observed that the *SYN* experiment performed worse than *NC* in both *W* and *UW* conditions. On the other hand, *F/W1*—and specially *F/W2*—were more accurate than *NC* in 10.7% and 21.4% respectively. We concluded that the principal factor to obtain better classification accuracy using cultural metadata is to restrict the search and query to sites with music oriented content, preferably with manual annotated information. However, the *Last.FM* genre classification experiment did not follow that trend. Indeed, it performed the worse of all experiments. To understand this phenomena, we studied the tags that Last.FM users uses and found that in general, they are very well defined for the *big-genres*, however when they want to go deeper for a thinner classification, there are many differences between in how people understand and tag the same artist. Moreover, there is big noise in the different way people spell

<sup>1</sup> www.last.fm

<sup>2</sup> http://code.google.com/apis/ajaxsearch/terms.html

and write artists and genres (e.g., *rock*, *rock and roll*, and *rock&roll* could be used for an user to refers to the same artist). Furthermore, tags do not represent only genres or styles, but anything the user want, from *mood* to *BPM*, so they are very noisy when we want to extract more detailed things.

## 6. CONCLUSIONS AND FUTURE DEVELOPMENT

We have developed an extension for the jWebMiner software framework allowing to use it with the Last.FM API to retrieve artists tags and use it for genre classification. The retrieved data can be reported and saved as usual as all other queried data from search engines using jWebMiner. In addition, a set of required text filters and site weightings were proposed to enhance the accuracy in genre classification. These constraints were tested with the SAC dataset to measure it with ground truth, and have comparison points. Doing different experiments, we obtained improvements of 5.2% and 16.7% for 5 and 10 genre classification with previous experiments. In addition, there is some space to perform better if the scoring function would be modified, allowing threshold points or non-linear curve.

We concluded that extracting data from music-related sites give better results than extracting information from all over the web, so text-mining techniques should be studied to retrieve information from specialized music sites, artist biographies, album reviews, and music blogs. However, extracting user tags from music communities is very noisy if the data is not pre and post processed.

Our future research will study *big-genres* extracted from Last.FM, to be used in companion with the artist name in jWebMiner to look for small genre classification. In addition, similar techniques could be used for mood extraction.

## 7. REFERENCES

Arnold, K., J. Gosling, and D. Holmes. 2000. *The JAVA programming Language*. Boston, MA: Addison-Wesley.

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

Barbedo, J.G.A., and A. Lopes. 2007. Automatic genre classification of musical signals. *EURASIP Journal on Advances in Signal Processing* (1):1–12.

Baumann, S., and O. Hummel. 2003. Using cultural metadata for artist recommendations. *Proceedings of the Third International Conference on Web Delivering of Music*. Leeds. 138–41.

Eck, D., T. Bertin-Mahieux, and P. Lamere. 2007. Autotagging music using supervised machine learning. *Proceedings of the 8th International Conference on Music Information Retrieval*. Vienna. 367–8.

Geleijnse, G., and J. Korst. 2006. Learning effective surface text patterns for information extraction. *Proceedings of the EACL Workshop on Adaptive Text Extraction and Mining*. 1–8.

— — —. 2006. Web-based artist categorization. *Proceedings of the 7th International Conference on Music Information Retrieval*. Victoria. 266–71.

Knees, P., E. Pampalk, and G. Widmer. 2004. Artist classification with web-based data. *Proceedings of the 5th International Conference on Music Information Retrieval*. Barcelona. 517–24

Liang, Y. D. 2001. *Introduction to JAVA programming*. Upper Saddle River, NJ: Prentice-Hall.

McKay, C., and I. Fujinaga. 2007. jWebMiner: a web-based feature extractor. *Proceedings of the 8th International Conference on Music Information Retrieval*. Vienna. 113–4.

— — —. 2008. Combining features extracted from audio, symbolic and cultural sources. *Proceedings of the 9th International Conference on Music Information Retrieval*. Philadelphia. 597–602.

Pachet, F., and D. Cazaly. 2000. A taxonomy of musical genres. *Proceedings of the 6th Content-Based Multimedia Information Access (RIAO) Conference*. Paris. 1238–45.

Pampalk, E., A. Flexer, and G. Widmer. 2005. Hierarchical organization and description of music collections at the artist level. *9th European Conference on Research and Advanced Technology for Digital Libraries*. Vienna. 37–48.

Raymond, Y., and M. Sandler. 2008. A web of musical information. *Proceedings of the 9th International Conference on Music Information Retrieval*. Philadelphia. 263–8.

Reed, J., and C. H. Lee. 2007. A study on attribute-based taxonomy for music information retrieval. *Proceedings of the 8th International Conference on Music Information Retrieval*. Vienna. 485–90.

Schdel, M. 2008. Automatically extracting, analyzing, and visualizing information on music artists from the world wide web. PhD thesis, Johannes Kepler University.

Schdel, M., T. Pohle, P. Knees, and G. Widmer. 2006. Assigning and visualizing music genres by web-based co-occurrence analysis. *Proceedings of the 7th International Conference on Music Information Retrieval*. Victoria. 260–5.

Turnbull, D., L. Barrington, and G. Lanckriet. 2008. Five approaches to collecting tags for music. *Proceedings of the 9th International Conference on Music Information Retrieval*. Philadelphia. 225–30.

Whitman, B. 2005. Learning the meaning of music. PhD thesis, Massachusetts Institute of Technology, USA.

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

Whitman, B., and P. Smaragdis. 2002. Combining musical and cultural features for intelligent style detection. *Proceedings of the 3rd International Conference on Music Information Retrieval*. Paris. 47–52.

Zadel, M., and I. Fujinaga. 2004. Web services for music information retrieval. *Proceedings of 5th International Conference on Music Information Retrieval*. Barcelona. 478–83.

*Allmusic website*. <http://www.allmusic.com>

*Amazon web services*. <http://aws.amazon.com>

*The Echonest API*. <http://developer.echonest.com>

*Java technology*. <http://www.java.sun.com>

*jWebMiner*. [http://jmir.sourceforge.net/index\\_jWebMiner.html](http://jmir.sourceforge.net/index_jWebMiner.html)

*Last.fm API*. <http://www.last.fm/api>

*Netbeans IDE*. <http://www.netbeans.org>