Last.fm の API を使用した音楽検索システムの構築

2006MI113 内藤 晶　　2007MI180 小川 由希也　　2007MI251 内山 慎介
指導教員　石崎 文雄

1　はじめに

近年, Amazon.com や Google といった大手 IT 企業が, 自社のサービスを外部からも利用できるように API を公開したことを皮切りに, 世界中の企業が自社サービスの API をこぞって公開するようになった. 単純な検索 API からテキスト, 画像, 音声などの解析 API まで実に多種多様な API が存在し, それらの多くは無償で提供され, 「Web API」としてインターネット上に広く普及するようになった. API を公開する理由の一つとして, API を利用してもらうことで間接的にせよ自社のサービスのユーザが増えることが期待できるためである. Web API とは, アプリケーションの開発者が他のハードウェアやソフトウェアの提供している機能を利用するための手法である. Web API を使うことで, プログラムを開発する際の手間を省くことができる. 開発者は OS などの提供者が定めた手続きに従って, 必要な機能の API を呼び出すようにプログラミングすることで, Web サイトなどの高機能なコンテンツをより短期間・低コストで開発できるようになった.

現在, 様々な音楽検索サイトがインターネット上に存在する. これらはハミング検索, 曲名, 歌手名, アルバム名からの検索がほとんどである. これらのサイトでジャンル毎のトップレートの検索は基本的に, ダウンロード回数や販売数などを元に出している. 販売数などの視点ではなく, 再生回数によるジャンル毎のアーティスト検索をすることでリスナー視点での検索が可能である. この検索方法が Last.fm の公開 API を使うことで実現できる. Last.fm とは, 音楽のストリーミング配信をメインとするコミュニティサイトの名称である. 音楽を中心にユーザー同士のつながりを広げていく SNS（ソーシャルネットワーキングサービス）の要素を持っているのが特徴である. 自分がよく再生する歌, 一番好きな歌, 一定期間に特定アーティストを再生する頻度, 似ているテイストを持った友だちなどを知ることができ, コミュニティに参加すれば, トラックのタグ付け, 意見交換への参加, 最近の流行ウォッチなども利用できる [1].

Last.fm の API を使うと結果が XML 文書で返ってくるため, 本研究では Perl を使用した. Perl には XML を取り扱うためのモジュールが豊富にあり, それらのモジュールを XML ツールとして利用すれば効率良く XML の作成や処理などができる. 今回は XML::XPath というモジュールを使った. XML::XPath とは, XML 文書から必要なデータを抜き出すモジュールである [2].

Last.fm では似たテイストを持つアーティストを検索することができる. しかし, Last.fm の似たテイストを持つアーティスト検索の API を使うだけでは精度があまり高くないことが問題であると気づいた. 本研究ではこの検索の精度を上げ, ジャンル毎のトップアーティストなどを検索できる音楽検索システムの構築を行った.

なお, 内藤晶は主にシミュレーションの実行と実行結果の分析を, 小川由希也は主にプログラミングの作成を, 内山慎介はシミュレーションのためのモデリングを担当した.

2　システムの提案と設計

2.1　Last.fm の Web API 方式

Last.fm の Web API は REST 方式になっている. REST は「リソース」を扱うための考え方であり, 「リソース」とは, ブログの記事であったり, Web ページ全体のコンテンツであったりといった, ひとかたまりの情報を指す. 本研究での REST の考え方は, 単に HTTP と XML を利用してリソースの操作を行うという単純なくくりになっている [3].

2.2　システムの構想

本研究では, Last.fm の API を使用して指定したアーティストに似たアーティストの検索と今聞きたいジャンルの有名なアーティスト名とそのアーティストの有名な曲名を検索でき, その結果から YouTube の検索結果画面に行くことができるシステムを構築しようと考えた.

2.3　システムを実現するために使用する API

4 つの API を使うことでシステムを実現することができた. Last.fm の API が (1)tag.getTopArtist, (2)artist.getTopTracks, (4)artist.getSimilar の 3 つで, YouTube の API が (3)Data API の 1 つである. 使用する YouTube の API には, Last.fm の API のように個別の名称がなかったため総称で呼ぶことにする. それぞれの API の機能を解説する.

(1)tag.getTopArtist

指定したタグに関連するアーティストを, ランクの高い順に整列された XML 文書にする. 以下に, API の記述例を示す. これは, タグを pop と指定した場合の例である. 指定した pop の人気アーティスト 1 位から最大 50 位までをランクの高い順に整列された XML 文書を返す.

tag.getTopArtist

```
http://ws.audioscrobbler.com/2.0/?method
=tag.gettopartists&tag=pop&api_key
=b25b959554ed76058ac220b7b2e0a026
```

(2)artist.getTopTracks

指定したアーティストの曲を, 人気の高い順に整列された XML 文書にする. 同様に, 記述例を示す. これは, アーティストを Madonna と指定した場合の例である.

指定した Madonna の曲 1 位から最大 50 位までを人気の高い順に整列された XML 文書を返す.

artist.getTopTracks

```
http://ws.audioscrobbler.com/2.0/?method
=artist.gettoptracks&artist=Madonna&api_
key=b25b959554ed76058ac220b7b2e0a026
```

(3)Data API

指定したワードで検索した結果を XML 文書にする. 同様に記述例を示す. これは, ワードを Madonna Like a Prayer と指定した場合の例である. 指定した検索ワードと一番関連性の高い結果を XML 文書として返す.

Data API

```
http://gdata.youtube.com/feeds/api/videos?
q=Madonna Like a Prayer&start-index=1&max-
results=1&v=2
```

(4)artist.getSimilar

本研究で使用する similar は, アーティスト A に似ているアーティスト TOP50 を XML 文書として返してくれる API である. 同様に, 記述例を示す (A を Madonna としている). ここで, 似ているアーティストにはそれぞれ match 度という数値がついてくる. この match 度は一番似ているアーティストを 1 として以下, 図 1 のように数値が下がっていく. この match 度はなだらかに下がっていくものもあれば急激に下がるものもある.

artist.getSimilar

```
http://ws.audioscrobbler.com/2.0/?
method=artist.getsimilar&artist=Madonna
&api_key=b25b959554ed76058ac220b7b2e0a026
&limit=50
```

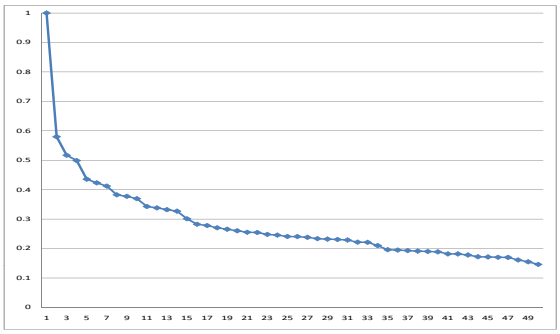

図 1　Madonna の match 度

2.4　similar の問題点

similar による検索では, 問題点がある. それは, 図 2 ようにアーティスト A から見たアーティスト B の match 度が 0.7 でも, アーティスト B から見たアーティスト A が 0.7 になるとは限らないことである. これは基準をどちらに置いたかによって match 度の算出結果が変わってくるためだと考えられる. match 度は類似性スコアを算出した結果の値だと考えられるが, 基準を変えると値も変わるままでは精度が低いと考えられる.

図 2　両方向の match 度

2.5　解決方法

similar の問題を解決するためには両方向の macth 度を調べる必要がある. match 度が 0 になることも考慮に入れ, 両方向からの match 度を処理することでより精度の高い検索結果が出ると考えた. マッチ度の処理方法について, 両方向からの match 度をどのように計算するかを考え試した. いくつかのアーティストを検索, 処理をした. match 度が 0 になることも考慮に入れ, 正常な結果が出た「両方向からの match 度の平均を得点とする」方法をとることで精度の高い検索結果が出ると考えた. 以降この方法を使った検索を「処理を行った similar 検索」, この方法を使わなかった検索を「処理を行わない similar 検索」と呼ぶことにする.

2.6　システムの構築

2.6.1　similar による検索

指定したアーティストに似たアーティストの検索をするプログラム全体の処理の流れを以下に示す.

(1):ユーザが入力したアーティスト A を変数で受け取り, API(artist.getSimilar) を使って A という名前のアーティストがいるか Last.fm に問い合わせる.

(2):問い合わせた結果, A がいた場合は Last.fm から XML 文書で match 度のデータが返ってくる. 該当するものがなければ Last.fm から XML 文書でエラーのデータが返ってくる. そのデータを受け取り, ファイルに保存する. Last.fm が返す XML 文書にエラーに関するノードがあるか調べ, エラーがあったらエラー表示をし, 終了する. XPath というモジュールを使って XML 文書をツリーとしてモデル化し, ノード (要素や属性) への位置を指定してエラーがあるか調べている.

(3):(2) で取得した XML 文書からアーティスト名と match 度を (2) と同様に, XPath を使いノードへの位置を指定して抜き出し, 配列に入れる (これを 1 ホップ目とする).

(4):(3) で抜き出したアーティストにそれぞれ API(artist.getSimilar) を使って Last.fm に問い合わせる. ここでは, &limit=50 を&limit=100 に変えることで A への match 度を得るために深く調べている.

(5):(4) で取得した XML 文書の match 度のデータを受け取り, ファイルに保存する.

(6):(5) で取得した XML 文書からアーティスト A への match 度を (2) と同様に, XPath を使いノードへの

位置を指定して抜き出し, 配列に入れる（これを 2 ホップ目とする）.

(7):1 ホップ目と 2 ホップ目の match 度の平均を出し降順に並べ替え, 平均の高い順に TOP10 のアーティスト名とその平均, 前の順位として 1 ホップ目の順位を表示する.

2.6.2 タグによる検索

今聞きたいジャンルの有名なアーティスト名とそのアーティストの有名な曲名を検索でき, その結果から YouTube の検索結果画面に行くことができるプログラム全体の処理の流れを先と同様に示す.

(1):ユーザが入力したジャンルを変数で受け取り, API(tag.getTopArtist) を使って該当するジャンルがあるか Last.fm に問い合わせる.

(2):問い合わせた結果, 該当するものがあれば Last.fm から XML 文書でジャンルのデータが返ってくる. 該当するものがなければ Last.fm から XML 文書でエラーのデータが返ってくる. そのデータを受け取り, ファイルに保存する. XPath を使いノード（要素や属性）への位置を指定してエラーに関するノードがあるか調べ, エラーがあったらエラー表示をし, 終了する.

(3):(2) で取得した XML 文書からランク 1 位から 10 位までのアーティスト名を抜き出し, 配列に入れる. (2) と同様に, XPath を使いノードへの位置を指定してアーティスト名だけを抜き出している.

(4):(3) で抜き出した 10 組のアーティスト名を使い, それぞれの曲のデータを API(artist.getTopTracks) を使って Last.fm に問い合わせる.

(5):(4) で取得した 10 組それぞれのデータを XML 文書で受け取り, ファイルに保存する.

(6):(5) で取得した XML 文書からそれぞれランク 1 位の曲名を抜き出し, 配列に入れる. (2) と同様に, XPath を使いノードへの位置を指定して曲名だけを抜き出している.

(7):(3) と (6) で配列に入れたデータを検索ワードとした, YouTube での検索結果の URL を Data API を使って YouTube へ問い合わせる.

(8):(7) で取得したデータを XML 文書で受け取り, ファイルに保存する.

(9):(2) と同様に, XPath を使いノードへの位置を指定して (8) で取得した XML 文書から URL とタイトル名を抜き出す. エラーの XML 文書が返ってきた場合は, リンクを表示する変わりにリンクを得られなかったことをエラーとして表示する. そして, 今まで取得したデータを使って, TOP10 それぞれのアーティスト名と曲名, それらを検索ワードとして得た YouTube へのリンクを表示する.

3 実行結果

3.1 similar の比較

similar 検索の比較を行うために, 1980 年代にデビューし, 今も活動している POP なアーティストとして Kylie Minogue を使った. このアーティストに処理を行わない similar 検索をし, 得た結果の TOP5 を表にしたものが表 1 である. 左が順位, 中がアーティスト名, 右が検索アーティストの match 度である.

表 1　処理を行わない similar 検索

	アーティスト名	match 度
1	Dannii Minogue	1
2	Madonna	0.474618
3	Sophie Ellis-Bextor	0.422288
4	Girls Aloud	0.340768
5	Cheryl Cole	0.332762

表 2 では Kylie Minogue に処理を行った similar 検索をした結果の TOP5 である. 表 1 と同じように左が順位, 中がアーティスト名, 右が検索アーティストの match 度である.

表 2　処理を行った similar 検索

	アーティスト名	match 度
1	Dannii Minogue	1
2	Madonna	0.737309
3	Scissor Sisters	0.6250285
4	Sophie Ellis-Bextor	0.5870325
5	Robyn	0.499526

また, この表をそれぞれをグラフにしたものが図 3, 図 4 である. グラフは TOP5 までではなく, TOP50 までを表している. 図 3 が処理を行わない similar 検索のグラフで, 図 4 が処理を行った similar 検索のグラフである. それぞれ縦軸が検索したアーティストへの match 度, 横軸が順位である.

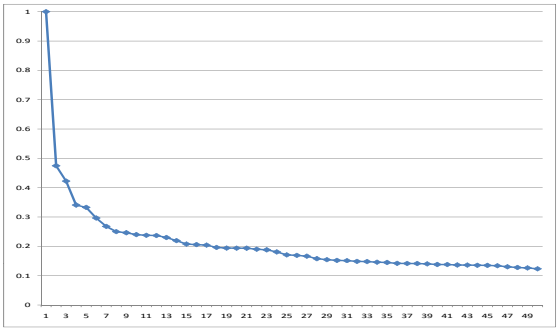

図 3　処理を行わない similar 検索

図 3 と図 4 は POP なアーティストを検索したものであるが, 図 3 に比べると図 4 のが 2 位以下の match 度がなだらかに下がっている. これは, 検索アーティストから似たテイストを持つアーティストへの match 度より, 似たテイストを持つアーティストから検索アーティストへの match 度のが高いためであり, POP なアーティス

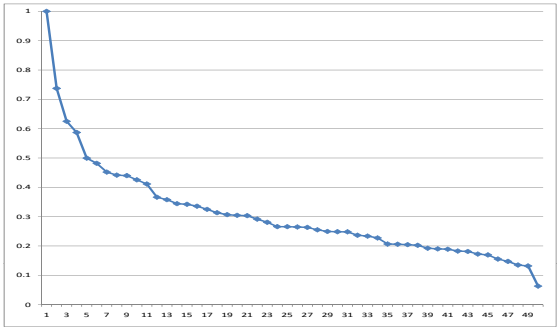

図 4　処理を行った similar 検索

トは他のアーティストを聞いている人も比較的聞きやすいためであると推察できる．よって，このようなグラフの変化があるアーティストは POP なアーティストであると推測できる．これに対して，Avril Lavigne を個性的なアーティストとして検索した場合，2 位以下の match 度が急激に下がっていた．これは，検索アーティストから似たテイストを持つアーティストへの match 度より，似たテイストを持つアーティストから検索アーティストへの match 度のが低いためであり，個性的なアーティストを聞く人は他のアーティストも聞くが，他のアーティストからは個性的なものを聞きにくいと推察できる．よって，このようなグラフの変化があるアーティストは個性的なアーティストであると推測できる．

3.2　ブラウザ表示

今回作ったシステムを Web ブラウザ上で実行できるように CGI を使用した．CGI とは，Web サーバが Web ブラウザからの要求に応じてプログラムを起動するための仕組みである．CGI は様々な開発言語でも使用できるのが特徴的で本研究では Perl 上で使用した [4]．図 5 と図 6 は Web ブラウザ上に表示した本システムである．

図 5 ではジャンル検索により POP と入力した結果である．Rank.1 が Madonna となっており，次の行で Madonna の一番聞かれている曲:Like a Prayer が表示されている．また，最後の行では，Madonna Like a Prayer を検索ワードとして YouTube で検索した際の一番関連の高い結果のリンクが張ってある．

図 5　ジャンル検索

図 6 では処理を施した similar 検索による検索結果が出力されている．左から順位，アーティスト名，match 度，1 回目の検索時の match 度を表している．少し差はあるが match 度の合計が 0.6 以上あるアーティストは検索したアーティストと曲調がかなり似ていると感じた．

図 6　similar 検索

4　おわりに

本研究では，WEB システム開発において各社が公開している API を利用して新しいシステムを作成することに着目し，Last.fm の API を利用して音楽検索システムの試作機作成に至った．

一部の機能を処理することで，似たアーティストの検索から幅を広げていくことができた．この検索方法は，似たアーティストの高度な検索に加えて，検索アーティストが個性的なのか，POP なのかなどが読み取れた．また，ジャンル検索では，Last.fm のデータから指定したジャンルの人気のあるアーティストとそれらのアーティストの代表曲となるものを検索することが出来る．このシステムを利用することによって，自分の聞きたいジャンルで人気のあるアーティストや曲を知り，YouTube を利用することですぐにそれらのアーティストの曲を聞くことが可能になった．

参考文献

[1] Last.fm(accessed 2010.6), http://www.lastfm.jp/api.

[2] 有限会社ミューテック：Perl テクニックブック, C&R 研究所 (2006.12).

[3] Think IT(accessed 2010.6), http://thinkit.co.jp/free/article.

[4] IT 用語辞典 (accessed 2010.10), http://e-words.jp/w/CGI.html.

[5] YouTube(accessed 2010.10), http://code.google.com/intl/ja/apis/youtube/overview.html.