электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.02

Обзор существующих средств распознавания музыкальных произведений

Унчикова Е.С., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»

Научный руководитель: Гапанюк Ю.Е., к.т.н., доцент,
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
gapyu@bmstu.ru

На данный момент существует множество приложения различных платформ для идентификации музыкальной композиции. Самые известные приложения MusicBrainz, TrackID, Shazam. На серверах хранятся около 150 000 музыкальных композиций, многие компании делают функционал открытым для добавления пользователями произведений.

Получить название и автора можно, записав небольшой отрывок интересующей нас песни. Возьмем для примера всем известный «Shazam» и рассмотрим его алгоритм работы.

Основные шаги:

1. Сначала Shazam снимает так называемые «отпечатки пальцев» для полного каталога музыки и сохраняет их в базе данных.

2. Когда начинается распознавание аудио-дорожки, создаются метки. Чтобы данные метки были созданы, достаточно анализировать звук в течение 10 секунд.

3. Приложение загружает записанные метки службами Shazam, которые одновременно осуществляют поиск «отпечатка» в их базе данных.

4. Если совпадение найдено, то пользователь получает информацию о песни, в обратном случае выводится ошибка.

Со стороны пользователя начинает работать кодогенератор, который создает уникальный «отпечаток» записанной музыкальной дорожки и отправляет результат на сервер.

Основная идея алгоритма - спектрограммы. Спектрограмма – это зависимость мощности сигнала от времени. Каждую аудио-дорожку можно представить в виде такого графика. Три оси соответствуют времени, частоте и интенсивности. Предполагается, что

на Ox находится время, на Oy представлены значения частоты, линия по горизонтали будет представлять собой непрерывный звук, который совершает небольшие гармонические колебания, а вертикальная линия будет представлять собой кратковременную вспышку стационарного шума. Точка на графике - интенсивность на данной частоте и в определенную единицу времени.

На рисунке 1 представлен пример спектограммы.

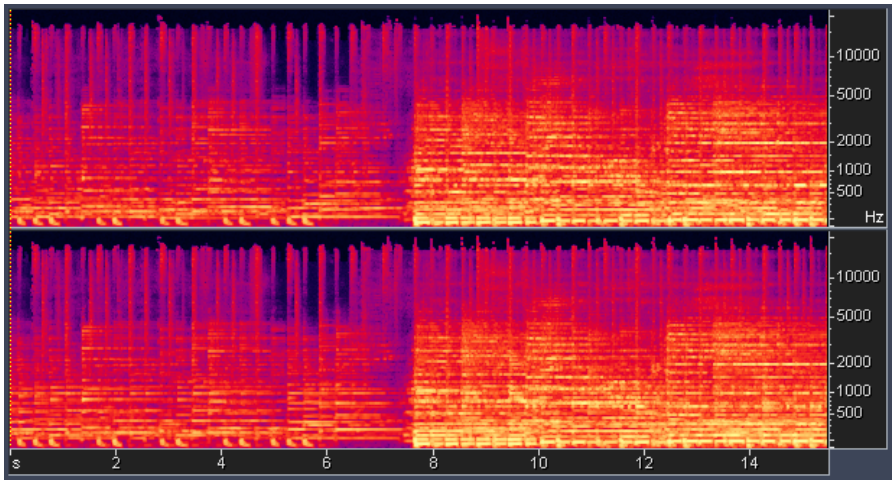

Рис. 1. Пример спектограммы

Данный 3D-график создается алгоритмом Shazam при получении «отпечатков» песни, извлекается частота "наибольшей интенсивности" («peak intensity»). Пики используют по причине их надежности при присутствии шума. Для каждой из этих точек пика он определяет значение частоты и временной промежуток от начала дорожки (таблица 1). Таким образом «отпечаток» есть не что иное, как отмеченные пики на этой спектограмме. Предполагается, что находят около трех из этих точек в секунду, обеспечивая этим неплохую скорость работы приложения.

Таблица 1

Пример «отпечатка» для образца звука в течение 10 секунд.

Частота, Гц	Время, с
823.44	1.054
1894.31	1.247
727.84	1.671
. . .	. . .
822.71	9.852

У Shazam есть огромная библиотека отпечатков, к которой имеют доступ ее приложения. Данные заносятся в таблицу: в одной колонке будет частота этого пика, а в другой время, когда этот пик произошел, например пик произошел на 1 с 30 мс , а частота была 920 Гц. И так далее до конца дорожки. Программа на устройстве делает точно такой же отпечаток и сверяет его с базой.

Хэши отпечатков получаются из пиков интенсивности. Опорные точки имеют некоторую область, связанную с ней. Shazam создает перечень меток, как хэш-таблицу, где ключом является частота. Для поиска используется хэш – сдвиг по времени. В базе данных ищутся совпадающие хэши. Как только приложение получает отпечаток, применяется первый ключ (в данном случае 823,44) и ищет все соответствующие композиции с похожей записью (таблица 2).

Таблица 2

Хэш-таблица

Частота, Гц	Время в секундах и информация о песни
823.43	53.352, “Песня А” Артист 1
823.44	34.678, “Песня В” Артист 2
823.45	108.65, “ Песня С’ Артист 3
. . .	. . .
1892.31	34.945, “ Песня В” Артист 2

На спектрограмме отмечается не одна точка, а некоторая пара точек: наибольшей интенсивности и опорная. Т.е. ключ это не только одно значение частоты, это сочетание обоих значений. Это приводит к меньшим хэш-коллизиям.

Скорость поиска композиций может составлять около 500 миллисекунд, если в базе данных находится порядка 20 000 песен.

Благодаря этому мы можем получить название композиции и исполнителя песни путем записи звука приложением.

Рассмотрим другой вариант алгоритма распознавания треков. Первая российская компания, предложившая свою реализацию алгоритма распознавания композиций, стала «Яндекс», выпустившая новое мобильное приложение «Яндекс.Музыка». Небольшой музыкальный фрагмент (10 секунд) поступает на сервер в качестве запроса. Если точных совпадений по трекам не происходит, то ответ будет отрицательным. Благодаря пользователям Яндекс.Музыки формировалась большая часть базы записей. Исходя из

статистики наиболее интересных для слушателей треков, получилось 6 миллионов композиций.

Компания «Яндекс» сразу отбросила несколько малоподходящий алгоритмов, одним из которых являлся алгоритм сравнения спектограмм, которые, по сути, представляет собой картинки. У этого способа есть следующие минусы:

1. Операция является дорогой, т.к. изображения сравниваются с 6 млн. записями. Некоторые различия – показательнее, чем остальные.
2. Разработчики встали перед вопросом: «А какие характеристики записанных дорожек лучше всего сохраняются, например, под воздействием шума?».

Использование вариант алгоритма пиковых частот, по мнению сотрудников «Яндекс», не подходит: высоту пиков модифицирует АЧХ микрофона, но их положение на частотно-временных осях мало меняется. Таким образом, одна композиция имеет порядка 300 тысяч пиков. Такое количество сравнить с базой гораздо легче, чем всю спектограмму.

Но даже такой способ не является совершенным, ведь шум может добавлять пики или вовсе заглушать их, а при передаче частота пиков может смещаться. Следовательно, основными задачами становятся:

1. Поиск такого отрезка времени, где число совпадений по пикам будет максимально
2. Выбрать из множества записей трек с наибольшим количеством совпадений.

Для этих целей необходимо построить гистограмму. Ох – число совпадений, Оу – сдвиг по времени.

Пример гистограммы представлен на рисунке 2.

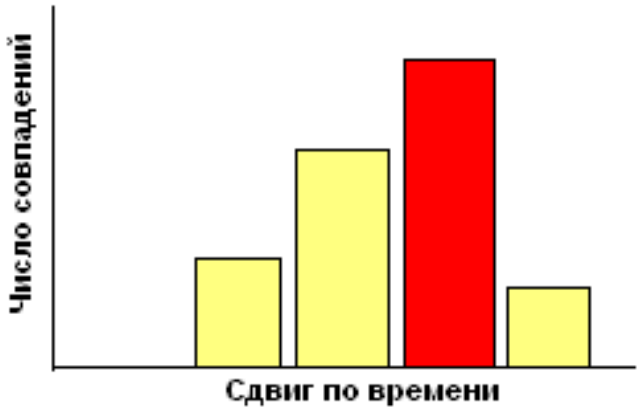

Рис. 2. Гистограмма

В результате самый высокий столбец и есть искомый результат.

Для ускорения поиска необходимо выбрать небольшое количество записей, с которыми будет происходить сравнение.

Если в качестве ключа взять одно значение частоты пика, то количество перебираемых вариантов все равно будет велико. Поэтому выгоднее рассматривать пары рядом стоящих пиков, ведь они будут встречаться гораздо реже. Значение каждого пика можно включить в несколько пар. В таком случае число просмотренных записей будет на 3 порядка ниже. Гистограмму можно применять не только к единичным пикам, но и к парам.

Для еще большего ускорения поиска ответа разработчики приняли решение использовать две фазы поиска:

Первая. Проводится отбор композиций по менее плотному набору различных пиков. Характеристики будут подбираться для снижения количества сравнений, но при этом для поиска наиболее соответствующих вариантов.

Вторая. Находится наиболее подходящий вариант, где индекс – это трек, определяющий пару частот и местоположение в треке.

Вышеперечисленные фазы ускорили поиск в десяток раз. В большинстве случаев результаты первой фазы совпадают с результатами второй.

Бывают случаи, когда в ответ на запрос нет подходящей композиции. Для принятия данного решения используется так называемая кривая релевантности.

На рисунке 3 изображен пример кривой релевантности.

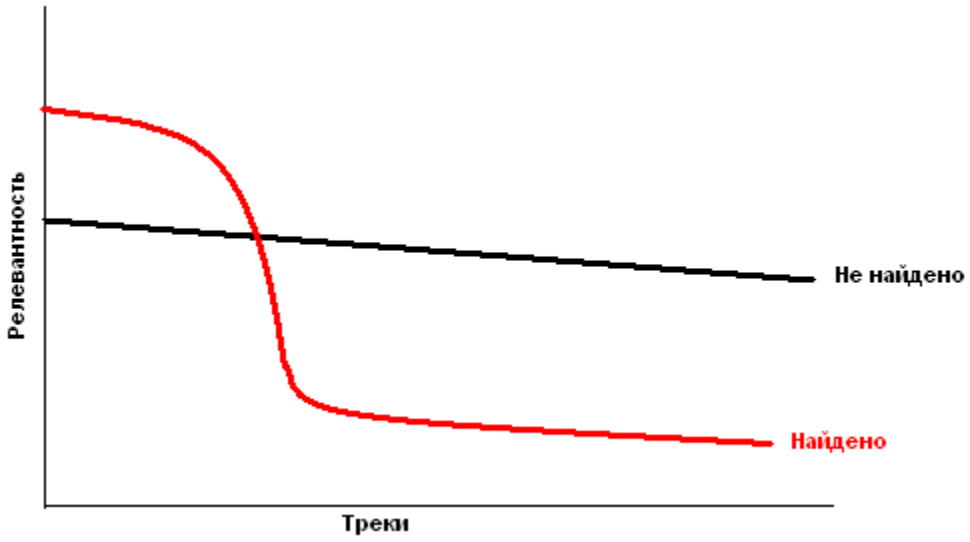

Рис. 3. Кривая релевантности

В ближайшем будущем планируется решить еще ряд интересных задач:

- Осуществить алгоритм поиска так называемых «неточных» совпадений, например, каверов.
- Поиск треков при искажении темпа
- Поиск по напетому фрагменту, где требуется кардинально другой подход, где вместо аудиозаписи должно использоваться нотное представление композиций

Таким образом, в статье были рассмотрены современные эффективные методы распознавания музыкальных треков. Также был проведен пошаговый разбор каждого алгоритма с наглядными примерами. Проанализированы способы оптимизации по скорости данных алгоритмов. Приведен план необходимых будущих доработок рассматриваемых методов.

Список литературы

1. Based Audio Fingerprinting. Available at: http://labrosa.ee.columbia.edu/matlab/fingerprint, accessed 20.02.2015 .
2. Haitsma J., Kalker T. A Highly Robust Audio Fingerprinting System // Artificial Intelligence. Vol. 129, Issue 1–2. 2001. P. 40–50.
3. How Shazam Works. Available at: https://laplacian.wordpress.com/2009/01/10/how-shazam-works, accessed 01.03.2015.
4. Галкин В.А., Григорьев Ю.А. Телекоммуникации и сети. М.: Издательство МГТУ им. Н.Э. Баумана, 2003. 609 с.
5. Цатурян Т. Ш., Гапанюк Ю. Е. Обзор способов распознавания посетителей веб ресурсов на основе получаемой или оставляемой ими информации // Молодежный научно-технический вестник. МГТУ им. Н.Э. Баумана. 2014. № 1. Электрон. журн. Режим доступа: http://sntbul.bmstu.ru/doc/688991.html (дата обращения 23.12.2014).
6. Фазылова Э. Ф. Системы генерации музыки или как автоматизировать искусство? // Молодежный научно-технический вестник. МГТУ им. Н.Э. Баумана. Электрон. журн. 2014. № 6. Режим доступа: http://sntbul.bmstu.ru/doc/723360.html (дата обращения 10.03.2015).
7. Галкин В.А. Алгоритм нечеткого фонетического поиска на основе простых чисел.// Молодежный научно-технический вестник. МГТУ им. Н.Э. Баумана. Электрон. журн. 2012. № 7. Режим доступа: http://sntbul.bmstu.ru/doc/457989.html (дата обращения 19.03.2015).