Алгоритм поиска звука промышленного уровня

http://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf

Avery Li-Chun Wang
avery@shazamteam.com
Shazam Entertainment, Ltd.

USA:
2925 Ross Road
Palo Alto, CA 94303
United Kingdom:
375 Kensington High Street
4th Floor Block F
London W14 8Q

Мы разработали и запустили в коммерческую эксплуатацию гибкий движок поиска звука. Алгоритм стоек к шуму и искажениям, вычислительно эффективен и хорошо масштабируем, способен быстро выявлять короткий отрезок музыки, захваченный через микрофон мобильного телефона в присутствии на переднем плане голоса и других доминирующих шумов, и прошедший через кодек сжатия голоса, в базе данных из более миллиона треков. Алгоритм использует комбинаторно хэшированный анализ частотно-временного признака звука, что приводит к необычным свойствам, таким как прозрачность, когда могут быть определены несколько дорожек, смешанные вместе. Кроме того, для таких приложений, как мониторинг радио, достигается время поиска порядка нескольких миллисекунд на запрос, даже на обширной музыкальной базе данных.

1 Введение

Shazam Entertainment, Ltd. стартовала в 2000 году с идеей предоставления услуги, которая могла бы соединить людей с музыкой, распознавая музыку в окружающей обстановке, используя для прямого распознавания музыки их мобильные телефоны. Алгоритм должен был быть в состоянии распознать короткий звуковой образец музыки, который был передан, смешанный с сильным окружающим шумом, в условиях реверберации и другой обработки, захваченный маленьким микрофоном мобильного телефона, подвергнутый компрессии голосового кодека и выпадению пакетов в сети, всему этому до поступления на наши серверы. Алгоритм также должен был выполнять быстрое распознавание по большой базе данных музыки с почти 2 млн. треков и кроме того иметь низкое количество ложных срабатываний, имея высокий уровень распознавания.

Это была нелёгкая задача и в то время не было никаких известных нам алгоритмов, которые могли бы удовлетворить всем этим ограничениям. В конце концов мы разработали нашу собственную технику, отвечающую всем эксплуатационным ограничениям [1].

Мы развернули алгоритм для измерений в нашей коммерческой службе распознавания музыки с более чем 1.8 млн. треков в базе данных. Услуга в настоящее время работает в Германии, Финляндии и Великобритании с более чем с полумиллионом пользователей, а вскоре будет доступна в других странах в Европе, Азии и Америке. Использование выглядит следующим образом: пользователь слышит музыку в окружающей обстановке. Он связывается с нашим сервисом с помощью своего мобильного телефона и передаёт образцы до 15 секунд звука. Идентификация по образцу осуществляется на нашем сервере, затем название

композиции и исполнитель отправляются обратно пользователю с помощью текстового SMS сообщения. Информация также доступна на веб-сайте, где пользователь может зарегистрироваться и войти в неё с помощью номера мобильного телефона и пароля. На веб-сайте или на смартфоне пользователь может посмотреть свой список отмеченных треков и купить компакт-диск. Пользователь может также скачать рингтон, соответствующий помеченному треку, если это возможно. Пользователь также может отправить 30-секундный кусочек песни другу. В ближайшем будущем могут стать доступными другие услуги, такие как приобретение загрузки MP3.

В последнее время появилось множество похожих потребительских услуг. Musiwave развернула подобный сервис идентификации музыки с помощью мобильного телефона на испанском операторе мобильной связи Amena с использованием надёжного алгоритма хэширования Philips [2-4]. Используя алгоритм от Relatable, Neuros включил функцию отбора образцов в свой MP3 плеер, который позволяет пользователю собрать 30-ти секундный фрагмент со встроенного радио, а потом подключиться к интернет-серверу для определения музыки [5,6]. Audible Magic использует алгоритм Muscle Fish, предлагая сервис Clango для идентификации потокового звука от Интернет-радиостанции [7-9].

Алгоритм Shazam может быть использован во многих приложениях, а не только распознавания музыки с помощью мобильного телефона. В связи с возможностью работы при сильном шуме, мы можем определить музыку, скрытую за громким голосом на переднем плане, например, в радио-рекламе. С другой стороны, алгоритм также очень быстр и может быть использован для мониторинга авторских прав при скорости поиска больше реального времени в более 1000 раз, что позволяет скромному серверу мониторить значительное число медиа-потоков. Этот алгоритм также подходит для разметки на основе содержания и индексирования для библиотечных и архивных целей.

2 Основной принцип работы

У каждого аудиофайла берётся "отпечаток", процесс, в котором извлекаются воспроизводимые маркеры хэша. Оба звуковых файла "в базе данных" и "образец" подвергаются одинаковому анализу. Отпечатки от неизвестного образца сравнивается с большим набором отпечатков, полученных из музыкальной базы данных. Кандидаты на соответствие впоследствии оцениваются на точное совпадение. Руководящими принципами для атрибутов для использования в качестве отпечатков являются такие, которые должны быть локализованы во времени, не подвергаться преобразованиям, надежными и иметь достаточно хорошее распределение (энтропию). Основной подход к определению участка во времени предполагает, что каждый хэш отпечатка рассчитывается с использованием сэмплов звука вблизи соответствующей точки во времени, так что далёкие события не влияют на хэш. Аспект неподверженности преобразованиям означает, что хэши отпечатков, полученные из соответствующего контента, воспроизводимы независимо от позиции в звуковом файле до тех пор, пока временной участок, содержащий данные, из которых вычисляется хэш, содержится в файле. Это имеет смысл, поскольку неизвестный образец может прийти из любой части оригинальной звуковой дорожки. Надёжность означает, что хэши, сгенерированные из оригинального чистого трека базы данных, должны быть воспроизводимы из ухудшенной копии звука. Кроме того, маркеры отпечатков должны иметь достаточно высокую энтропию (хорошее распределение) в целях сведения к минимуму вероятности ложных совпадений в несоответствующих местах между неизвестным образцом и треками в базе данных. Недостаточная энтропия приводит к чрезмерным и ложным совпадениям в несоответствующих местах, требуя больше вычислительной мощности, чтобы отбирать результаты, а слишком большая энтропия обычно приводит к хрупкости и невоспроизводимости маркеров отпечатков

в присутствии шумов и искажений.

В следующих разделах представлены 3 основных компонента.

2.1 Надёжные оценки

В целях решения проблемы надёжной идентификации в присутствии весьма значительного шума и искажений, мы экспериментировали с различными характеристиками кандидата, которые могут выжить при GSM кодировании в присутствии шума. Мы остановились на пиках спектрограммы из-за их надёжности в присутствии шума и приблизительно линейной совпадаемости (superposability) [1]. Частотно-временная точка является пиком кандидата, если она имеет более высокое содержание энергии, чем все её соседи в области вокруг данной точки. Пики кандидата выбираются в зависимости от критерия плотности, чтобы гарантировать, что частотно-временная полоса для звукового файла имеет достаточно равномерное покрытие. Пики в каждой частотно-временной области также выбираются в зависимости амплитуды, на основании того, что самые высокие по амплитуде пики, скорее всего, переживут вышеперечисленные искажения.

Таким образом, сложная спектрограмма, такая, как показанная на Рисунке 1A, может быть уменьшена до редкого набора координат, как показано на Рисунке 1B. Обратите внимание, что в этой точке амплитудная компонента была устранена. Это сокращение имеет преимущество достаточной нечувствительности к эквалайзеру, так как обычно пик в спектре по-прежнему остаётся пиком с теми же координатами в профильтрованном спектре (при условии, что производная функции передачи фильтра разумно мала - пики в окрестности резкого перехода в передаточной функции имеют небольшой сдвиг по частоте). Мы называем список разрежённых координат "картами созвездий", так как разбросанные координатные точки часто напоминают звёздное поле.

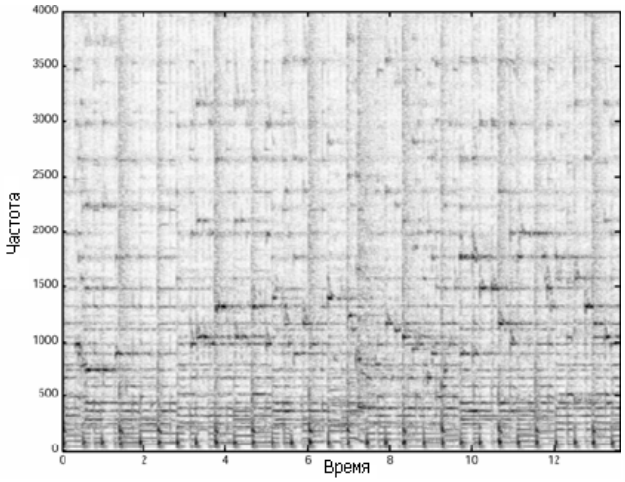

Рис. 1A, Спектрограмма

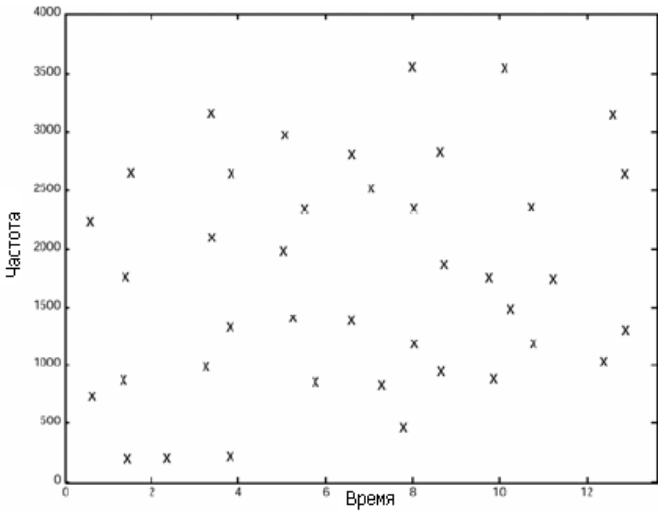

Рис. 1B, Карта созвездий

Узор из точек должен быть одинаковым для совпадающих сегментов звука. Если вы поместите карту созвездий песни в базе данных на ленточную диаграмму, а карту созвездий короткого соответствующего образца звука длиной несколько секунд на прозрачный кусок пластика, то при перемещении последнего над первым в какой-то момент при соответствующем смещении во времени и если обе карты созвездий выровнены по диапазону, значительное число точек совпадёт.

Количество совпадающих точек будет значительным в присутствии ложных пиков, возникающих из-за шума, поскольку позиции пиков относительно независимы; более того, количество совпадений может быть значительным, даже если многие из правильных точек были удалены. Регистрация карт созвездий, таким образом, мощный способ согласования в присутствии шума и/или удаления деталей. Эта процедура уменьшает проблему поиска до своего рода "астронавигации", в которой небольшой участок частотно-временных точек созвездий должен быть быстро обнаружен в большой вселенной точек на ленточной диаграмме вселенной, размеры ограниченного диапазона частот в сравнении с почти миллиардом секунд в базе данных.

Yang также рассматривал использование пиков спектрограмм, но использовал их по-другому [10].

2.2 Быстрое комбинаторное хэширование

Поиск правильной регистрации смещения непосредственно из карт созвездий может быть довольно медленным из-за необработанных точек созвездий, имеющих низкую энтропию. Например, ось частот из 1024 значений даёт лишь не более 10 бит данных о частоте для пика. Мы разработали быстрый способ индексации карт созвездий.

Хэши отпечатков формируются из карты созвездий, в которой пара частотно-временных точек комбинаторно связаны. Выбираются опорные точки, каждая опорная точка имеет целевую зону, связанную с ней. Каждая опорная точка последовательно спарена с точками в пределах своей целевой зоны, каждая пара даёт две частотных компоненты плюс разница во времени между точками (Рисунки 1C и 1D). Такие хэши вполне воспроизводимы даже в присутствии шума и при сжатии голосовым кодеком. Кроме того, каждый хэш может быть упакован в 32-х разрядное беззнаковое целое число. Каждый хэш также связан со смещением во времени от начала соответствующего файла до опорной точки, хотя абсолютное значение времени само по себе не является частью хэша.

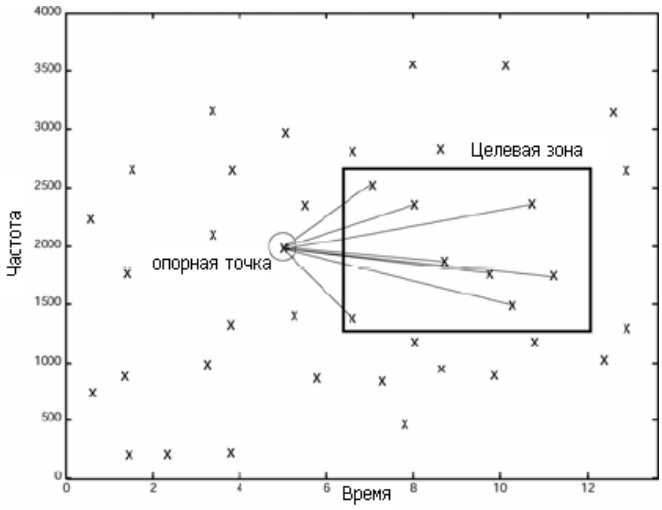

Рис. 1C Генерация комбинаторного хэша

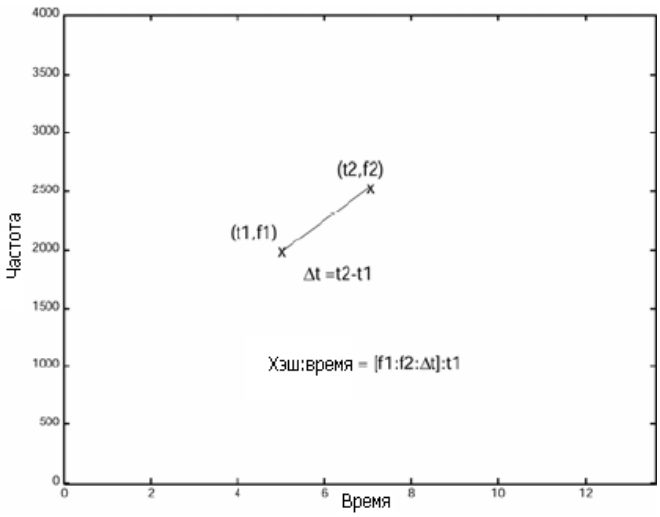

Рис. 1D Детали хэша

Для создания индекса базы данных вышеописанная операция проводится над каждым треком в базе данных, чтобы создать соответствующий список хэшей и связанных с ними смещений времени. Идентификаторы треков могут быть дополнены небольшими структурами

данных, приводя к общей 64-х разрядной структуре, 32 бита для хэша и 32 бита для смещения времени и идентификатора трека. Чтобы облегчить быструю обработку, 64-х разрядные структуры сортируются в соответствии со значением маркера хэша.

Количество обработанных хэшей аудио-записи в секунду примерно равно плотности точек созвездий в секунду умноженному на коэффициент разветвления в целевой зоне. Например, если каждая точка созвездия берётся как базовая точка и если целевая зона имеет коэффициент разветвления равный F=10, то количество хэшей примерно равно F=10 умножить на число точек созвездия, извлечённых из файла. Ограничивая число точек, выбранных в каждой целевой зоне, мы стремимся ограничить быстрый комбинаторный рост числа пар. Коэффициент разветвления приводит непосредственно к фактору стоимости с точки зрения места для хранения.

Формируя пары вместо поиска совпадений с отдельными точками созвездия мы получаем огромное ускорение процесса поиска. Например, если каждая частотная компонента имеет 10 бит и компонент Δt также 10 бит, то соответствующая пара точек даёт 30 бит информации, по сравнению с только 10 для одной точки. Тогда отличие такого хэша было бы примерно в миллион раз больше, в связи с 20-ю дополнительными битами, и, следовательно, скорость поиска для одного маркера хэша увеличена аналогично. С другой стороны, в связи с комбинаторной генерацией хэшей, предполагая симметричную плотность и коэффициент разветвления для генерации хэша базы данных и образца, есть в F раз больше комбинаций маркеров неизвестного образца для поиска и в F раз больше маркеров в базе данных, таким образом, общее ускорение является фактором около $1000000/F^2$, или около 10000, выше скорости поиска маркера на основе отдельных точек созвездия.

Обратите внимание, что комбинаторное хэширование это квадрат вероятности выживания точки, то есть если p является вероятностью выживания пика спектрограммы на пути от оригинального исходного материала до захваченного записанного образца, то вероятность хеша из пары выживших точек приблизительно p^2. Это уменьшение живучести хэша является компромиссом в сравнении с огромным предоставляемым увеличением скорости. Снижение вероятности выживания отдельного хэша смягчено комбинаторной генерацией большего количества хэшей, чем исходных точек созвездия. Например, если F=10, то вероятность, что по крайней мере один хэш выживет для данной точки привязки была бы объединённой вероятностью такой опорной точки и по меньшей мере одной целевой точки в своей целевой зоне выживания. Если упрощённо посчитать вероятность выживания p для IID (уникального идентификатора) для всех участвующих точек, то вероятность выживания по крайней мере одного хэша в опорной точке равна $p*[1-(1-p)^F]$. Для достаточно больших значений F, например, F>10, и разумных значений p, например, p>0.1, получаем примерно

$$p \approx p*[1-(1-p)^F]$$

так что мы на самом деле не сильно проигрываем тому, что было до этого.

Видно, что с помощью комбинаторного хэширования мы имеем компромисс с примерно в 10 раз большим местом для хранения и примерно в 10000 раз улучшение в скорости, и небольшую потерю в вероятности обнаружения сигнала.

Для разных условий сигнала могут быть выбраны разные коэффициенты разветвления и плотности. Для относительно чистого звука, например, для приложений радиоконтроля, F может быть выбран достаточно малым и плотность также может быть выбрана малой по сравнению с несколько более сложным приложением для мобильного телефона потребителя. Таким

образом, разница в требованиях к обработке может отличаться на много порядков.

2.3 Поиск и подсчёт

Чтобы выполнять поиск, вышеописанный шаг создания отпечатка выполняется на захваченном кусочке звукового файле для создания набора записей хэш:смещение во времени. Каждый хэш этого кусочка используется для поиска в базе данных совпадающих хэшей. Для каждого совпадающего хэша, найденного в базе данных, соответствующие смещения времени от начала кусочка и файлов базы данных связываются во временнЫе пары. ВременнЫе пары распределяются по коробочкам, в соответствии с индентификатором трека, связанным с соответствующим хэшем базы данных.

После того, как все хэши кусочка были использованы для поиска в базе данных, чтобы сформировать совпадающие временнЫе пары, коробочки проверяются на наличие совпадений. Внутри каждой коробочки такой набор временнЫх пар представляет график рассеяния, связанный с этим кусочком и звуковыми файлами базы данных. Если эти файлы совпадают, соответствующие характерные моменты должны находиться в аналогичных относительных смещениях от начала файла, то есть последовательность хэшей в таком файле должна также встречаться и в совпадающем файле с такой же последовательностью относительного времени. Проблема принятия решения о том, было ли найдено совпадение, сводится к выявлению значительного скопления точек, образующих диагональную линию в пределах графика рассеяния. Для выполнения обнаружения могут быть использованы разные методы, например, преобразование Хафа (Hough transform) или другие надёжные регрессионные методы. Такие методы являются слишком общими, требующими больших вычислительных ресурсов и восприимчивыми к выбросам.

Из-за жёстких ограничений задачи, следующая методика решает эту проблему примерно за время N*log(N), где N - число точек, появляющихся на диаграмме рассеяния. Для целей этого обсуждения можно считать, что наклон диагональной линии имеет 1.0. Тогда соответствующие моменты времени совпадающих характерных моментов между совпадающими файлами находятся в отношении

$$tk'=tk+offset,$$

где tk’ является координатой времени функции в совпадающем (чистом) звуковом файле базы данных, а tk является координатой времени соответствующего момента в звуковом файле кусочка, который должен быть идентифицирован. Для каждой координаты (tk’,tk) в рассеянии вычислим

$$\delta tk=tk'-tk.$$

Затем вычисляем гистограмму этих значений δtk и сканируем на пик. Это может быть сделано сортировкой множества значений δtk и быстрым сканированием кластера значений. Диаграммы рассеяния обычно очень редкие из-за специфики хэшей, использующих комбинаторный метод генерации, как говорилось выше. Так как количество пар времени в каждой коробочке мало, процесс сканирования занимает порядка нескольких микросекунд на коробочку или меньше. Число баллов совпадения представляет собой количество совпадающих точек на пике гистограммы. Наличие статистически значимого кластера указывает соответствие. Рисунок 2A показывает рассеяние в базе данных времени при сравнении со временем кусочка для трека, который не соответствует этому кусочку. Есть несколько случайных совпадений, но линейное соответствие не появляется. Рисунок 3A

показывает случай, когда на диагональной линии появляется значительное число совпадающих пар времени. Рисунки 2B и 3B показывают гистограммы значений δtk, соответствующих Рисункам 2A и 3A.

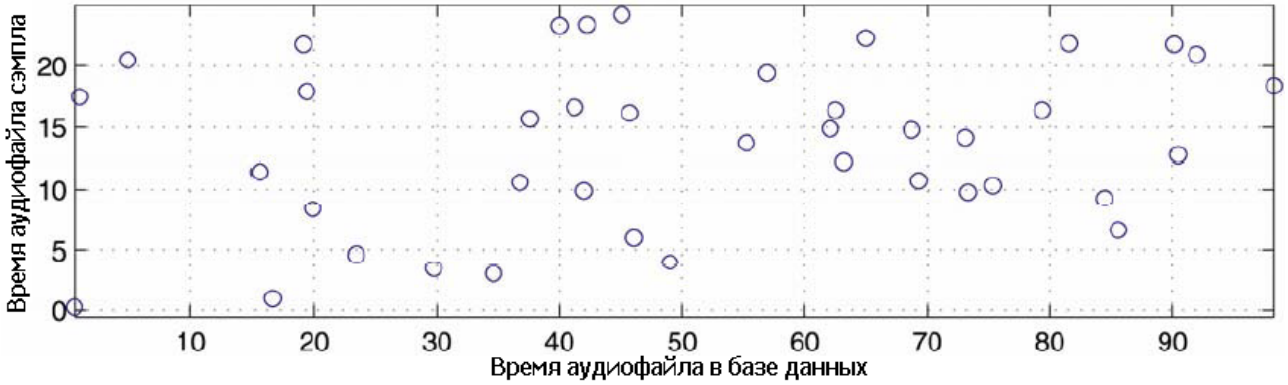

Рис. 2A Рассеяние положений совпадающих хэшей: Диагонали нет

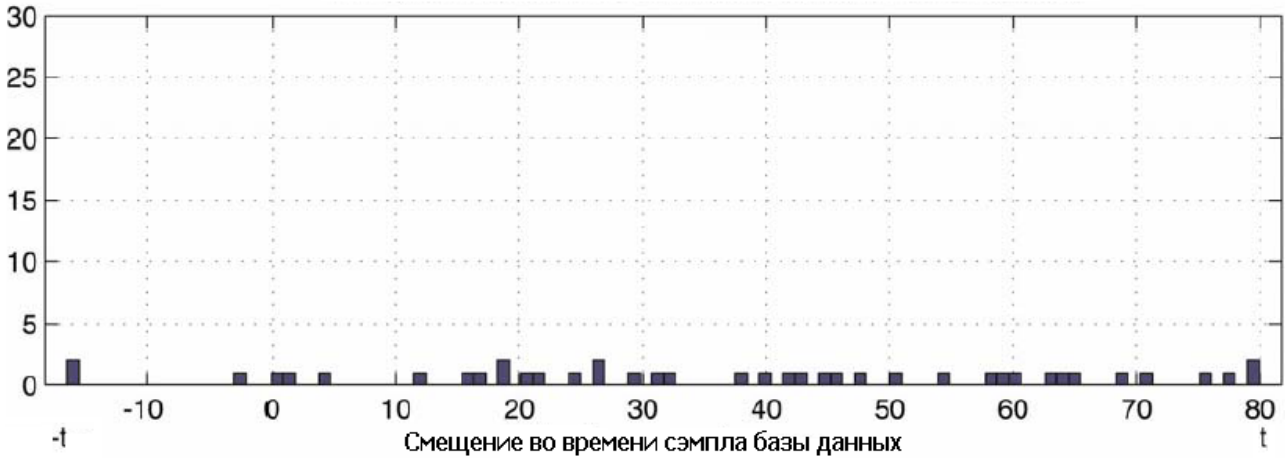

Рис. 2B Гистограмма разницы смещений во времени: сигналы совпадают

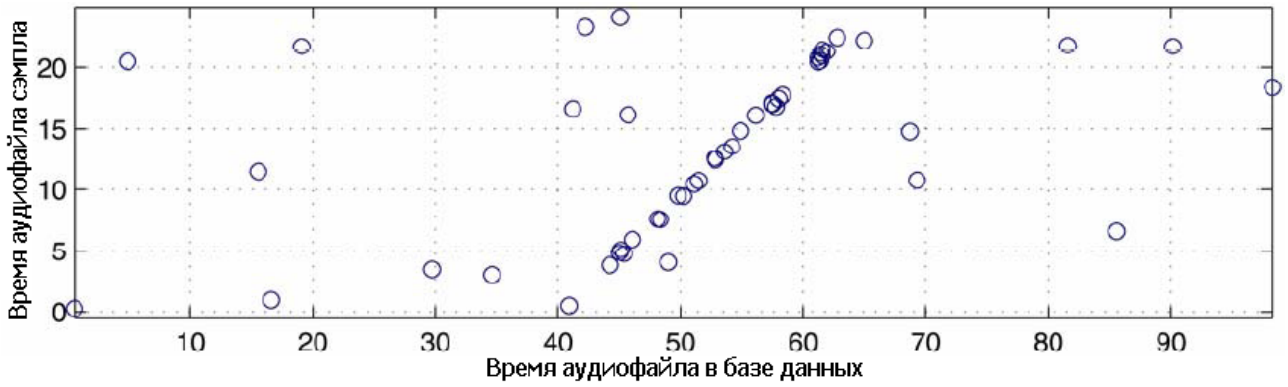

Рис. 3A Рассеяние положений совпадающих хэшей: Диагональ присутствует

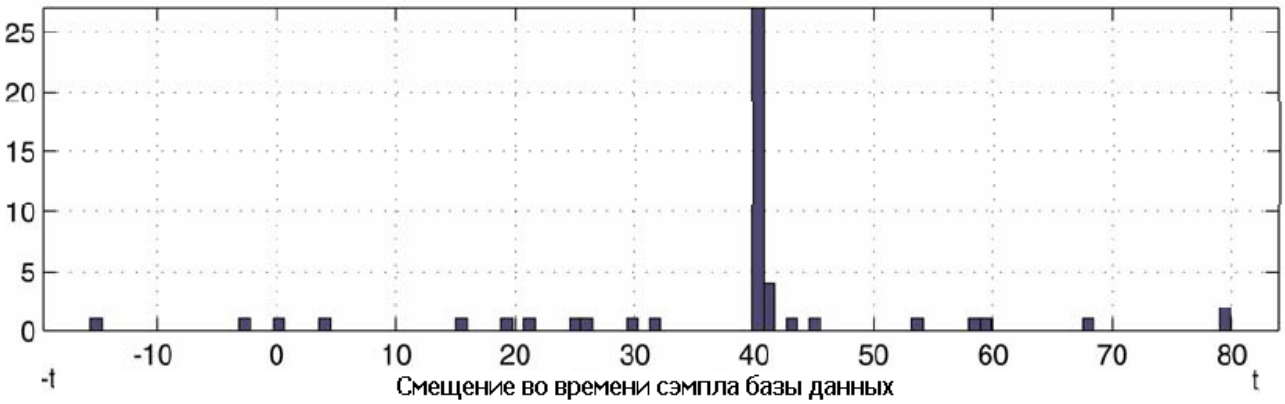

Рис. 3B Гистограмма разницы смещений во времени: сигналы совпадают

Этот процесс сканирования коробочек повторяется для каждого трека в базе данных пока не найдено значительное совпадение.

Обратите внимание, что фазы сравнения и сканирования не делают каких-то специальных предположений о формате хэшей. В самом деле, хэши должны иметь только свойства, имеющие достаточно энтропии, чтобы избежать слишком большого числа ложных совпадений, а так же быть воспроизводимыми. В фазе сканирования главное, что имеет значение, чтобы соответствующие хэши были выровнены во времени.

2.3.1 Значимость

Как описано выше, оценка - просто количество совпадающих и выровненных во времени маркеров хэша. Распределение баллов неправильно совпадающих треков представляет интерес в определении доли ложных срабатываний, а также долю корректного распознавания. Коротко говоря, собирается гистограмма оценок неправильно совпадающих треков. Учитывается количество треков в базе данных и создаётся функция плотности вероятности наивысших оценок неправильно сопоставленных треков. Затем выбирается приемлемая вероятность ложных срабатываний (например, 0.1% ложных срабатываний или 0.01%, в зависимости от приложения), затем выбирается порог баллов, соответствующий или превосходящий критерий ложного срабатывания.

3 Характеристики

3.1 Стойкость к шуму

Алгоритм хорошо работает при значительных уровнях шума и даже нелинейных искажениях. Он может правильно определить музыку в присутствии голоса, шума транспорта, выпадениях и даже другой музыки. Чтобы дать представление о силе этой техники, из сильно повреждённого 15-ти секундного сэмпла статистически значимое совпадение может быть определено с помощью выживших на самом деле только лишь около 1-2% порождённых хэш-маркеров и способствующих созданию смещения кластера. Свойство метода гистограммирования рассеяния в том, что разрывы не имеют значения, что позволяет иметь невосприимчивость к выпадениям и маскировке из-за помех. Один несколько неожиданный результат состоит в том, что даже с большой базой данных мы можем правильно определить каждый из нескольких смешанных вместе треков, в том числе нескольких версий одного и того же фрагмента, свойство, которое мы называем "прозрачность".

На Рисунке 4 показан результат выполнения распознавания 250 сэмплов различной длительности и разным уровнем шума с помощью тестовой базы данных из 10000 треков, содержащей популярную музыку. Сэмпл шума был записан в шумном пабе для имитации условий "реальной жизни". Отрывки звука длительностью 15, 10 и 5 секунд были взяты из середины каждого тестового трека, каждый из которых был взят из тестовой базы данных. Относительная мощность шума каждого тестового кусочка была нормирована до требуемого отношения сигнал/шум, а затем линейно добавлена к сэмплу. Видно, что скорость распознавания падает до 50% для 15, 10 и 5 секундных сэмплов при отношении сигнал/шум примерно -9, -6 и -3 дБ, соответственно. Рисунок 5 показывает тот же анализ, за исключением того, что результирующая смесь музыка + шум далее подвергнута сжатию GSM 6.10, а затем снова сконвертирована в звук с PCM. В этом случае 50% уровень скорости распознавания для 15, 10 и 5 секундных сэмплов имеет место примерно при отношении сигнал/шум -3, 0 и +4 дБ. Выборка звука и обработка проводились с использованием сэмплов 8 кГц, моно, 16-бит.

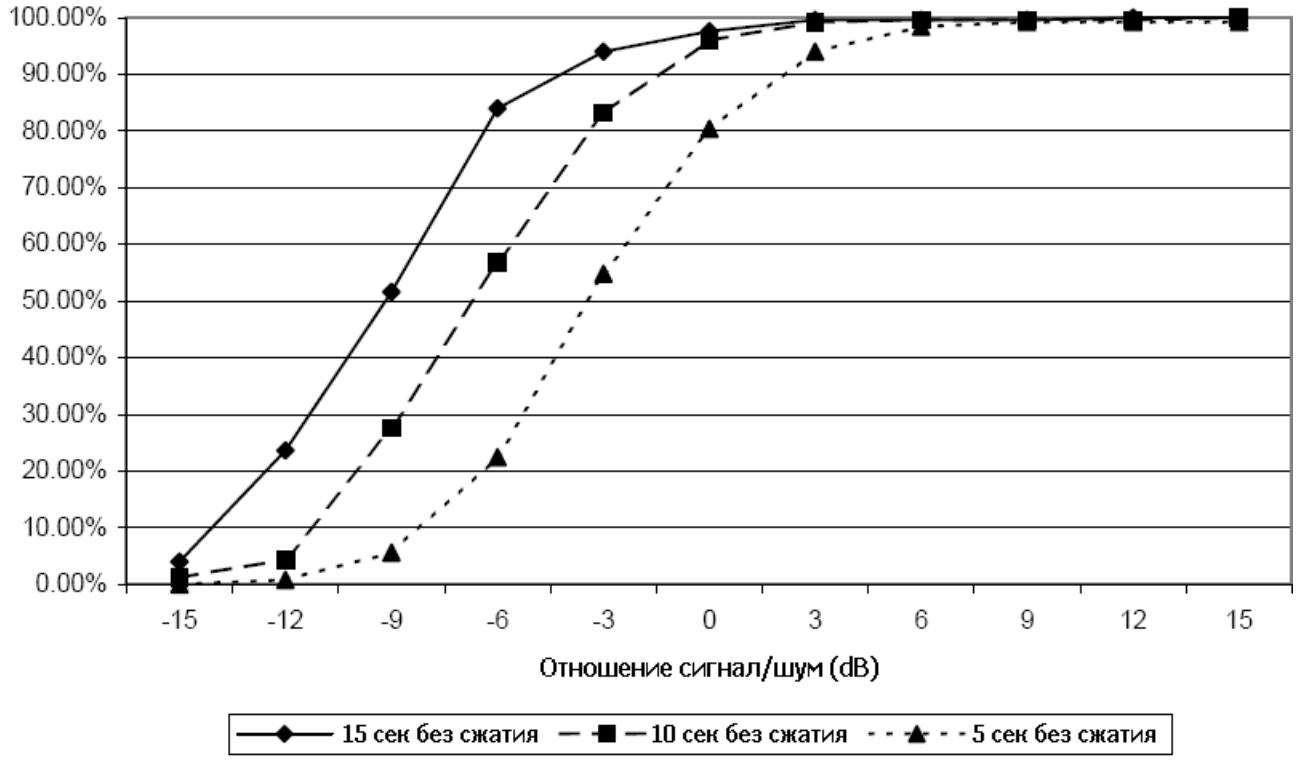

Рис. 4 Скорость распознавания - добавлен шум

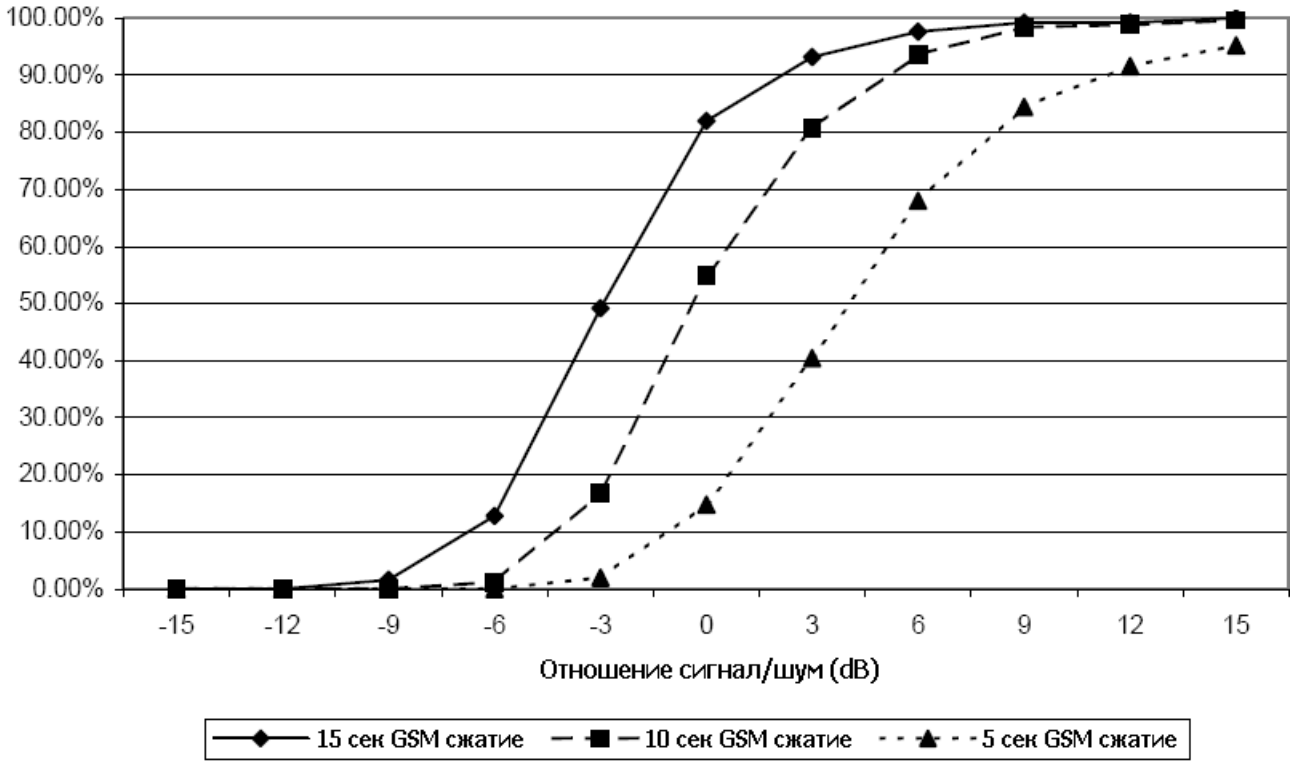

Рис. 5 Скорость распознавания - добавлен шум + GSM компрессия

3.2 Скорость

Для базы данных имеющей около 20 тысяч треков, реализованной на ПК, время поиска имеет порядок 5 - 500 миллисекунд, в зависимости от параметров настройки и приложения. Служба может найти соответствующий трек для сильно повреждённых звуковых сэмплов в течение нескольких сотен миллисекунд основного времени поиска. Для звука с "качеством радио" можно найти совпадение менее чем за 10 миллисекунд, а при оптимизации, вероятно, уменьшить до 1 мс на запрос.

3.3 Особенности и ложные срабатывания

Алгоритм был разработан специально с целью распознавания звуковых файлов, которые уже присутствуют в базе данных. Не ожидалось распространять на концертные записи. Тем не менее, мы случайно обнаружили на концертах несколько исполнителей, которые очевидно или чрезвычайно точны и воспроизводимы во времени (с точностью до миллисекунд), или более правдоподобно синхронизируют губы.

Алгоритм наоборот очень чувствителен к той именно версии трека, из которого был взят кусок. Учитывая множество различных версий одной и той же песни исполнителя, этот алгоритм может выбрать правильный, даже если они практически неотличимы для человеческого уха.

Мы время от времени получаем сообщения о ложных срабатываниях. Часто мы находим, что алгоритм на самом деле не был неверен, так как он нашёл пример "сэмплирования", или плагиат. Как уже упоминалось выше, существует компромисс между верными и ложными срабатываниями и, таким образом, максимально допустимый процент ложных срабатываний

является параметром при проектировании, который выбирается подходящим для данного приложения.

4 Благодарности

Особая благодарность Julius O. Smith, III и Daniel P. W. Ellis за обеспечение руководства. Спасибо также Chris, Philip, Dhiraj, Claus, Ajay, Jerry, Matt, Mike, Rahul, Beth и всем другим замечательным людям в Shazam, и моей Katja.

5 Литература

[1] Avery Li-Chun Wang and Julius O. Smith, III., WIPO publication WO 02/11123A2, 7 February 2002, (Priority 31 July 2000).

[2] http://www.musiwave.com

[3] Jaap Haitsma, Antonius Kalker, Constant Baggen, and Job Oostveen., WIPO publication WO 02/065782A1, 22 August 2002, (Priority 12 February, 2001).

[4] Jaap Haitsma, Antonius Kalker, “A Highly Robust Audio Fingerprinting System”, International Symposium on Music Information Retrieval (ISMIR) 2002, pp. 107-115.

[5] Neuros Audio web site: http://www.neurosaudio.com/

[6] Sean Ward and Isaac Richards, WIPO publication WO 02/073520A1, 19 September 2002, (Priority 13 March 2001)

[7] Audible Magic web site: http://www.audiblemagic.com/

[8] Erling Wold, Thom Blum, Douglas Keislar, James Wheaton, “Content-Based Classification, Search, and Retrieval of Audio”, in IEEE Multimedia, Vol. 3, No. 3: FALL 1996, pp. 27-36

[9] Clango web site: http://www.clango.com/

[10] Cheng Yang, “MACS: Music Audio Characteristic Sequence Indexing For Similarity Retrieval”, in IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2001