Free Won't

A curious engineer attempts to deconstruct everything

How Shazam Works

There is a cool service called Shazam (http://www.shazam.com/music/web/home.html), which take a short sample of music, and identifies the song. There are couple ways to use it, but one of the more convenient is to install their free app onto an iPhone. Just hit the "tag now" button, hold the phone's mic up to a speaker, and it will usually identify the song and provide artist information, as well as a link to purchase the album.

What is so remarkable about the service, is that it works on very obscure songs and will do so even with extraneous background noise. I've gotten it to work sitting down in a crowded coffee shop (http://ritualroasters.com/) and pizzeria (http://www.pizzeriadelfina.com/).

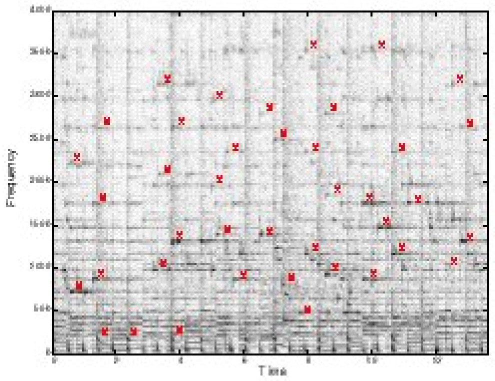
So I was curious how it worked, and luckily there is a paper written by one of the developers (http://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf) explaining just that. Of course they leave out some of the details, but the basic idea is exactly what you would expect: it relies on fingerprinting music based on the spectrogram (http://en.wikipedia.org/wiki/Spectrogram).

Here are the basic steps:

1. Beforehand, Shazam fingerprints a comprehensive catalog of music, and stores the fingerprints in a database.
2. A user "tags" a song they hear, which fingerprints a 10 second sample of audio.
3. The Shazam app uploads the fingerprint to Shazam's service, which runs a search for a matching fingerprint in their database.
4. If a match is found, the song info is returned to the user, otherwise an error is returned.

Here's how the fingerprinting works:

You can think of any piece of music as a time-frequency graph called a spectrogram. On one axis is time, on another is frequency, and on the 3rd is intensity. Each point on the graph represents the intensity of a given frequency at a specific point in time. Assuming time is on the x-axis and frequency is on the y-axis, a horizontal line would represent a continuous pure tone (http://en.wikipedia.org/wiki/Pure_tone) and a vertical line would represent an instantaneous burst of white noise (http://en.wikipedia.org/wiki/White_noise). Here's one example of how a song might look:



Spectrogram of a song sample with peak intensities marked in red. Wang, Avery Li-Chun. An Industrial-Strength Audio Search Algorithm. Shazam Entertainment, 2003. Fig. 1A,B.

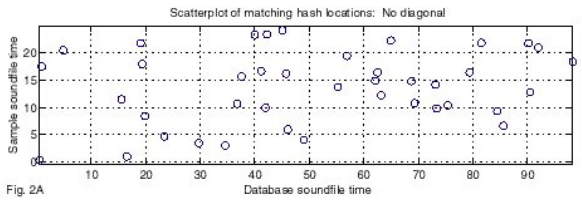
The Shazam algorithm fingerprints a song by generating this 3d graph, and identifying frequencies of "peak intensity." For each of these peak points it keeps track of the frequency and the amount of time from the beginning of the track. Based on the paper's examples, I'm guessing they find about 3 of these points per second. [Update: A commenter below notes that in his own implementation he needed more like 30 points/sec.] So an example of a fingerprint for a 10 seconds sample might be:

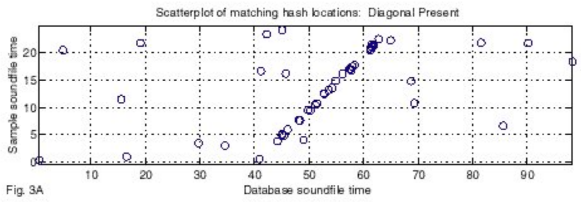
Frequency in Hz	Time in seconds
823.44	1.054
1892.31	1.321
712.84	1.703
. . .	. . .
819.71	9.943

Shazam builds their fingerprint catalog out as a hash table (http://en.wikipedia.org/wiki/Hash_table), where the key is the frequency. When Shazam receives a fingerprint like the one above, it uses the first key (in this case 823.44), and it searches for all matching songs. Their hash table might look like the following:

Frequency in Hz	Time in seconds, song information
823.43	53.352, "Song A" by Artist 1
823.44	34.678, "Song B" by Artist 2
823.45	108.65, "Song C' by Artist 3
. . .	. . .
1892.31	34.945, "Song B" by Artist 2

[Some extra detail: They do not just mark a single point in the spectrogram, rather they mark a pair of points: the "peak intensity" plus a second "anchor point". So their key is not just a single frequency, it is a hash (http://en.wikipedia.org/wiki/Hash_function) of the frequencies of both points. This leads to less hash collisions (http://en.wikipedia.org/wiki/Hash_collisions) which in turn speeds up catalog searching by several orders of magnitude by allowing them to take greater advantage of the table's constant (O(1)) (http://en.wikipedia.org/wiki/Constant_time) look-up time. There's many interesting things to say about hashing, but I'm not going to go into them here, so just read around the links in this paragraph if you're interested.]





Top graph: Songs and sample have many frequency matches, but they do not align in time, so there is no match. Bottom Graph: frequency matches occur at the same time, so the song and sample are a match. Wang, Avery Li-Chun. An Industrial-Strength Audio Search Algorithm. Shazam Entertainment, 2003. Fig. 2B.

If a specific song is hit multiple times (based on examples in the paper I think it needs about 1 frequency hit per second), it then checks to see if these frequencies correspond in time. They actually have a clever way of doing this They create a 2d plot of frequency hits, on one axis is the time from the beginning of the track those frequencies appear in the song, on the other axis is the time those frequencies appear in the sample. If there is a temporal relation between the sets of points, then the points will align along a diagonal. They use another signal processing method to find this line, and if it exists with some certainty, then they label the song a match.