Report on Shazam

By: Ankesh Gupta (2015CS10435)

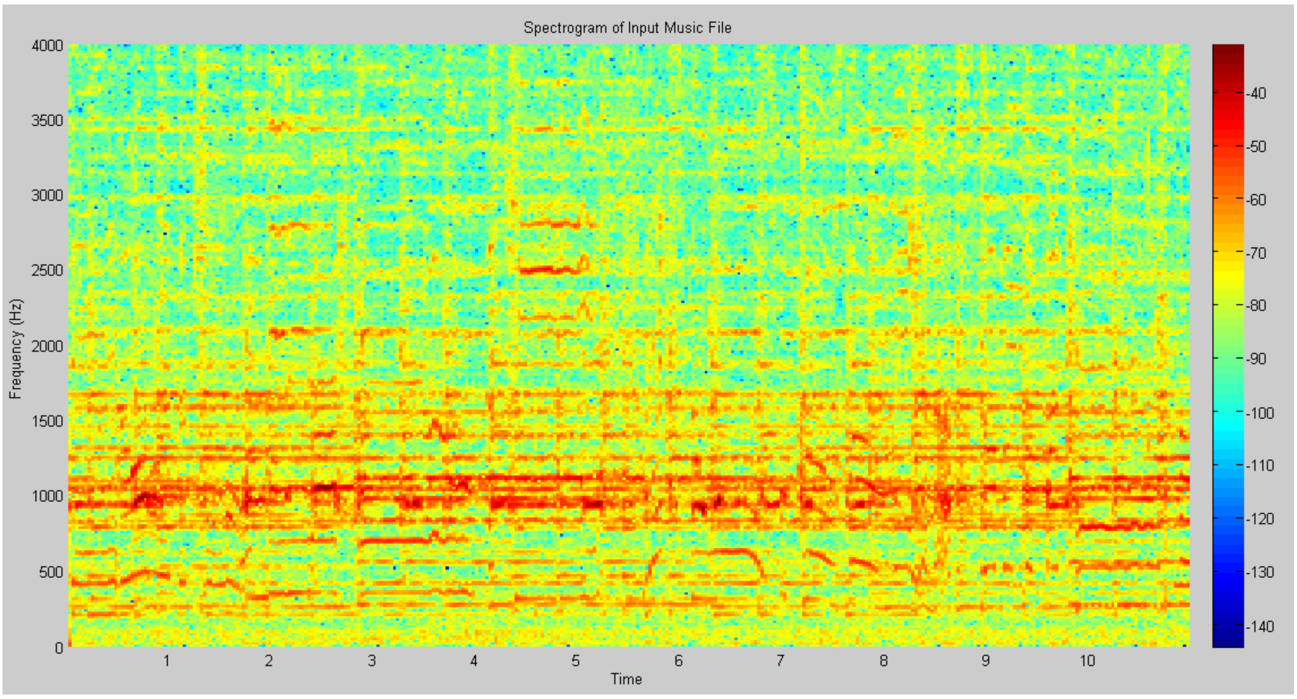
Shazam is a service that can in general, identify music, movies, advertising and television shows based on a short sample played and using the microphone on the device. It creates an **acoustic fingerprint** sample and compares it against a central database for a match.

Working:

1. A function that takes input a song and outputs table of information.

- Read the song (fs=44100 Hz) and down sample it to a frequency of 8000 Hz.
  Explanation: The input song contains too many samples making the whole process inefficient. Thus we resample it, so much so that we speed up, at the same time we don't lose much information.

- Compute the spectrogram of resampled song. **Spectrogram** is a visual representation of the spectrum of frequencies of sound or other signal as they vary with time or some other variable.

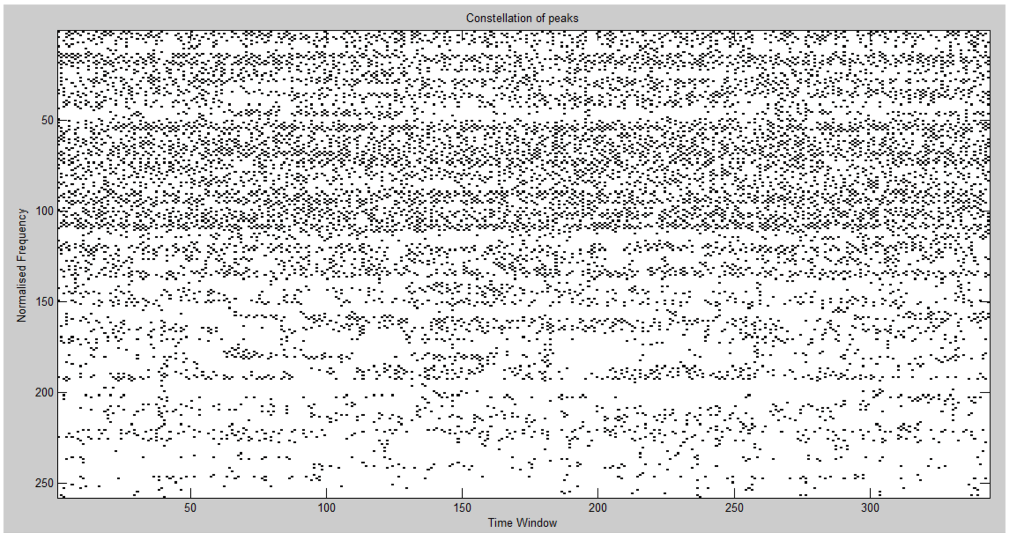


Spectrogram of Input Song 'viva.mp3' was plotted with ColorMap at side

- The song is divided into windows of 64 milliseconds and Fourier Transform of each interval is taken and plotted against frequency to give the desired heatmap as above.
  Windows are used in Spectrogram to prevent spectral leakages. 50% overlap (32 ms) was used.

- Local peaks are filtered from the above plot and a technique of **adaptive thresholding** is used where we threshold each time block so that we are left with ~30 peaks.
  Explanation: This step is necessary as to perform fingerprinting of input in the next step. This step helps further refine our characteristic points of the song so that we can perform efficient matching with less data, when given a song.

- The peaks then left are mapped as f(i)= { 1 if i!=0 , 0, if i=0 }, giving a Boolean matrix with 1's and 0's.



- **<u>Fingerprinting Algorithm</u>** is run, whereby a window is slided and local peaks within each window are paired. Each peak has belong to a corresponding time window(T_i) and some frequency window(F_i). We tap pair information in a table as

$$\left|\begin{array}{c|c|c|c} f_1 & f_2 & t_1^c & t_2^c - t_1^c \\ \vdots & \vdots & \vdots & \vdots \\ f_j & f_k & t_j^c & t_k^c - t_j^c \\ \vdots & \vdots & \vdots & \vdots \\ f_m & f_n & t_m^c & t_n^c - t_m^c \end{array}\right|$$

Where t_2-t_1 is time difference between 2 local peaks of sliding window.

Explanation: Storing t_2-t_1 helps make our system time shift invariant. As our clip may begin and end at any instant of the song, storing the difference will help us match feature of input clip with song database. We create a **hash table** where the key is a tuple **(f_1, f_2, t_2-t_1)** and value it stores in **t_1**. This table helps in efficient retrieval of data from database.

This hash table along with above mentioned table is returned by the function.

2. Each song in our database is passed through the above **black box** and their **fingerprint**, in the form of hash table is stored for future matching. This comprises our database.

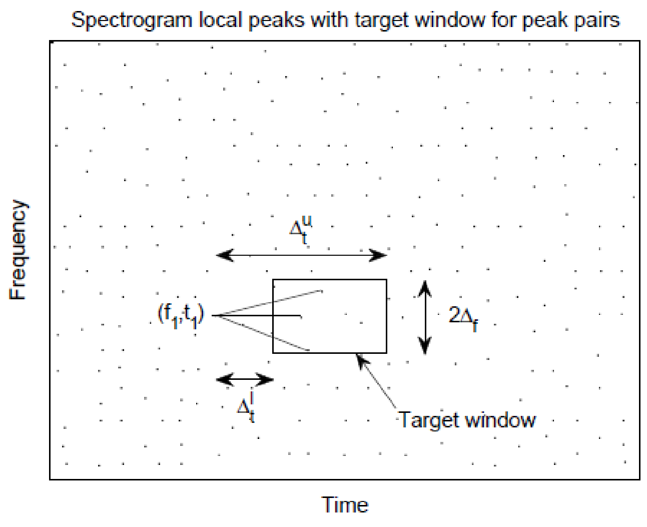
3. What happens when a clip (is played | given as input)?

- The input song is passed through our black box and a table of features is returned.
- For each feature returned, all songs in database are iterated through and a match is performed.
- We index into hash table of songs with key feature of clip as key for the table and get corresponding time value.
  Explanation: Here is where the fingerprinting and using the time difference comes handy. For a peak pair, all the 3 quantities – f1, f2 and t2-t1 are same in both the song and clip as taking t2-t1 removes the shift that the clip may have been played with. Hence **time shift invariance** helps tackle the offset problem.

- For each song, we count the number of matches of the input song and the one with the most number of matches is declared as the output song. The name is displayed accordingly.

Improvements:

The whole process needs to be made much more efficient. Also, currently we feed a song of fixed length as input (fixed but arbitrary). This can be made to receive a song from a microphone and perform real-time resolution like a modern Shazam.



<u>The fingerprinting algorithm where we slide a window and store peak pair information.</u>

References:

1. *A. Wang, An Industrial Strength Audio Search Algorithm, in Proc .ISMIR, Baltimore, USA, 2003*
2. *https://www.princeton.edu/~cuff/ele301/labs.html*
3. *https://en.wikipedia.org/wiki/Spectrogram*

Thank You.