Proc. of Int. Conf. on Current Trends in Eng., Science and Technology, ICCTEST

Encoding the Features of a Social Network into an Audio

PrasannaLakkurSubramanyam
Department of Computer Science, EWIT, Bengaluru, India
Email: prasanna.lakkur@gmail.com

***Abstract*—Explores encoding and communication of information on a social media profile through sound waves. Gives a concrete implementation of this process and also explores a general model using which we could do the same with any information source. Gives the reader insight into novel ways of encoding, compressing and communicating information using different medium, encoders and decoders.**

***Index Terms*—audio synthesis, social media analysis, modulation, encoding data.**

I. INTRODUCTION

When we look at how we receive information today, wemight notice that the medium is overwhelmingly visual. I'm not sure if people even realize that they could receive the same information through other senses. It's an alien concept to many. What is essential for communication is a sender, a receiver, a message, a transmission medium and a protocol. We often seem to forget that we have not one, but five senses. Five inputs through which we can receive information. We also seem to forget that all our languages are just protocols. We don't have to write “I saw a cat” in English language to communicate the message that we saw a cat. We could just say it out loud. That's another protocol. We could also perhaps think about generating some vibrations on a belt which could communicate this message. Another protocol. And yet, the vast majority of the time, especially on the internet, the information seems to be visual. On social media websites, we visit someone's profile and what do we do? We SEE his profile. Why not hear his profile instead? Why not smell or taste his profile? That's exactly what I've tried to do in this paper. I've implemented a system which lets you listen to a profile on Facebook. But that's not the aim of this paper. I don't want to talk about my specific implementation as much as I want to talk about the general idea that information on the Internet can be presented in different ways and using different communication protocols.

II. THE IDEA

At this point, you might be thinking that what I'm trying to do has already been done long ago and it is called a recording and playing audio. If I were doing something similar, the audio of a person's Facebook profile might just read out all of his profile details one by one. That is not what I'm after. I want to generate a sound which gives an intuitive understanding of the person's profile. Just as a person can intuitively tell how far a train is just by listening to the sound it creates, he should also intuitively be able to understand a person's

Grenze ID: 02.ICCTEST.2017.1.142

profile based on the sound. More concretely, the generated sound should have the following requirements –

- In the range of acceptable inputs, the audio should be unique to the input(one to one mapping).
- The audio should be easy to understand and interpret.
- A large change in the input should also change the audio in a noticeable way.

To give more insight into how this might be used, consider a few example applications which can be built using this technology–

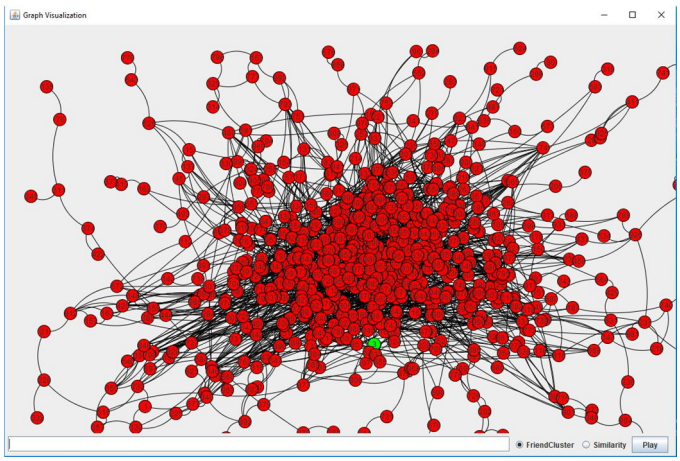

Figure 1. GUI demonstration for this project

- An app which lets the user browse social media through audio. A sort of a “social media radio” if you may. You can think of this “radio” as having different channels which the user can tap into. One channel might play an audio which gives you a sense of how many friends a user has, another might tell you how close his group of friends are, another might tell you how similar your profile is to his/her profile and so on. The radio keeps going through different profiles in the social media and depending on the channel that the user has selected, it'll play an audio. The interesting aspect of this technology is that the person using this app can just lie down and close his eyes. Just by listening to the audio, he'll be able to tell whether he wants to know more about the current profile.
- With the advent of Internet of Things, one might even build an app for something similar to Google Glass. An app which scans the people that the user of the glass encounters and plays a sound as and when he encounters them. Therefore, the user of the glass is not just seeing the person, he is listening to the person's profile at the same time! He's getting more insight into the person. One might argue that we could achieve the same thing using visual aids such as colors or graphs. Sure, you could. But you'd be using the same sense organ for everything. I'm speculating that it'll be a more engaging experience for the user if multiple senses are stimulated at the same time.

III. Quick Demo

Before getting into the details of the implementation, let's have a look at a GUI (Figure 1)that I've developed for my version of the “social media radio” [1]. I believe that seeing this before, rather than after the detailed explanation will offer valuable context which will help us understand the system better. The graph visualization at the center represents the social network that we're currently in. The visualization is done using JUNG library[2]. Each node in the network has an ID which is shown in the center of the circle representing the node. To hear the sound of any node, we need to enter the ID of the node in the input box given at the bottom and then, we need to select the “channel” of the “radio”. The audio starts playing once we click on the “Play” button. The node which represents the user of the software on the network is filled in green. If we select the “Similarity channel” and click on Play, an audio starts playing. This audio will be a function of how similar the green node's network is to the network of the node entered in the input box. If we instead select the “FriendCluster” channel, it'll play an audio which is a function of how many friends the entered node has and how clustered the node's friendship network is.

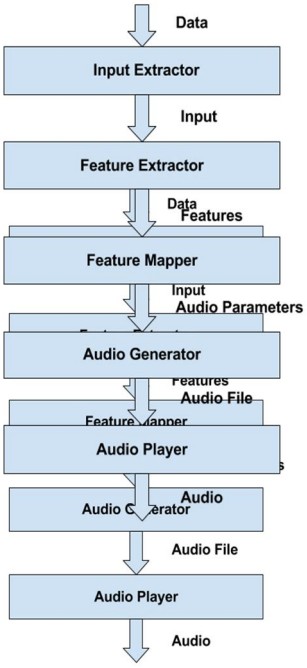

Figure 2. Layers in the model

IV. Design and Implementation

The implementation can be broken down into different layers with specific interfaces defining each layer. This makes it flexible and easy to change each layer without affecting the other layers too much. As I go through each layer, I shall describe, firstly, the functionality of the layer at a high level and then I shall present my implementation of the layer

Input Extractor

This layer insulates the rest of our software from changes in the format of data in the input source we're interested in and changes in the location of the input source. It is expected to take as input, any generic stream of bytes and produce as output, a set of features in a format that is agreed upon during the design of the software. The data set that I have used in my implementation is the USCD Facebook data set[3]. There are 14,948 vertices and 443,221 edges in this network when it is represented as a graph. The expected output in my implementation is an undirected graph representing the input network.

Feature Extractor

This layer takes as input, the output of the Input Extractor layer. It produces as output, a set of features based on which the audio will be generated. This set of features will be a function of the set of inputs. The audio will be a function of the set of features. The purpose of this layer is to insulate the lower layers from any features which are added and to separate the feature mapping from the feature extraction(i.e. same features can have different implementations which map them to different audio). My implementation keeps it slightly simple and uses the number of friends, the Clustering Coefficient and the Jaccard Similarity as the features of a given node.

Number of Friends

Number of friends can be obtained by looking at the number of outgoing edges from the given node.

Clustering Coefficient

“The clustering coefficient of a node N is defined as the probability that two randomly selected friends of N are friends with each other. In other words, it is the fraction of pairs of N's friends that are connected to each

other by edges"[4]. A simple analysis of the equation will reveal to us that the clustering coefficient will always be a value between 0 and 1. The higher the value, the more clustered the network is.

Jaccard Similarity

Jaccard Similarity gives us a sense of how similar two nodes in the graph, say, n1 and n2 are. Let's say that n1 and n2 have a set of friends f1 and f2 respectively. The Jaccard Similarity between n1 and n2 will be equal to the size of the intersection between f1 and f2 divided by the size of the union between f1 and f2. This will result in a value between 0 and 1. The higher the value, the more similar n1 and n2 are[5].

Feature Mapper

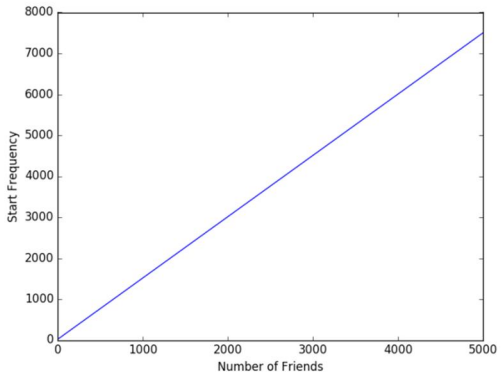

Figure 3. Relation between the "base frequency"

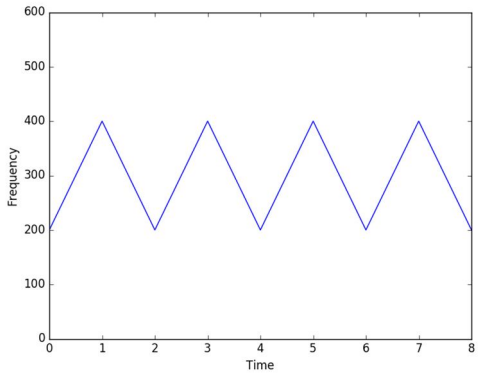

Figure 4. The periodic function generated for a node with number of friends.base frequency 200 and a clustering coefficient of 1

In this layer, we map the input features to parameters which describe the audio to be generated. Before this, however, during the design of the software, we have to decide upon the parameters that describe the audio to be generated. The parameters might be dependent on the type of software/library we use to generate the audio. In my implementation, the output, i.e. the audio parameters, are actually periodic functions. The output of the function represents the frequency of the sound wave. We have two "channels" - FriendCluster and Similarity. We have a periodic function for each channel.

FriendClusterFunction:

This depends on two features - number of friends and clustering coefficient. The frequency of the sound wave increases and decreases periodically depending on the above features. The lowest frequency(the frequency at the start, let's call this as "baseFreq" henceforth) is determined by the number of friends. I have picked a lower bound("minFreq") and an upper bound("maxFreq") for the frequency at which the audio is allowed to play. Since I'm dealing with Facebook data and I know that there can be anywhere from 0 to 5000 friends for any given node, I've scaled the base frequency linearly from "minFreq" to "maxFreq"/2 depending on the number of friends(refer Fig. 3). As I've mentioned before, the frequency goes up and down. This change in frequency depends upon the Clustering Coefficient. The audio starts playing at a "baseFreq" and increased upto some limit and then drops back to "baseFreq". This upper bound for the increase is given by the equation

$$upperBound = baseFreq + (clusteringCoEfficient * baseFreq)$$

Therefore, the audio will start at "baseFreq" and may go upto two times "baseFreq" (if the clustering coefficient is 1. This is why we choose the upper limit for "baseFreq" as "maxFreq"/2(Refer Fig. 4). If the network has a Clustering Coefficient of 0, the audio just keeps playing at the base frequency with no change.

Similarity Function:

Similarity encoding is also based on frequency and the change in frequency. There will be two periodic functions here. Let's call them f1 and f2. The output of f1 gives us the frequency of the output audio wave. f2 varies from 1 to -1 periodically. f1 function will start off at a set frequency(200 Hz in Figure 6). It will also

have a parameter called “displacement” which will specify the maximum allowed displacement of the value from the initial value(50 Hz in Figure 6). The output of f1 at a point x is calculated by

$$f1(x) = initialFrequency + (f2(x) * displacement)$$

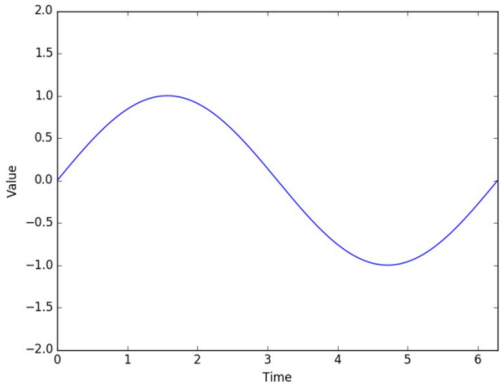

Figure 5.f2 function goes from -1 to 1 periodically

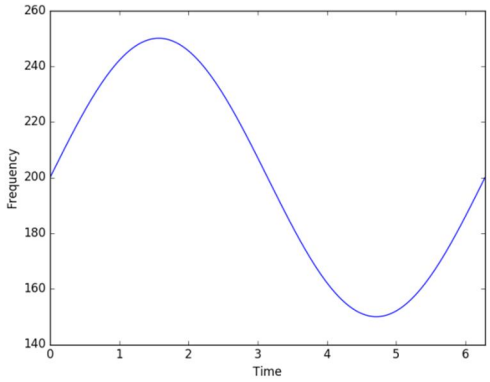

Figure 6.f1, Initial frequency = 200 Hz, displacement = 50 Hz

A simple analysis will reveal that f1 will vary between (initialFrequency - displacement) and (initialFrequency + displacement) periodically. Therefore, the output of f1 depends on the output of f2 as shown in figure 5 and 6. The frequency of f2 in turn depends on the Jaccard Similarity value. It goes from 0 to a specified maximum frequency depending on the value of the Jaccard Similarity. The frequency of f2 is given by

$$f = jaccardSimilarity * maxAllowedFrequency$$

Therefore, the rate of change in frequency of the sound wave will be directly proportional to the similarity between two nodes.

Audio Generator

This layer is used to generate the audio file based on the audio parameters. The output is an audio file. The format of the audio file is, again, something that needs to be decided during the design of the software. We need to ensure that the Audio Player works with the format chosen. This layer decouples the process of generation of the audio and the process of actually playing the audio. We might even export the audio file and play it on different Audio Players. We have some level of flexibility here. The Audio Generator that I've used is provided by the Beads library[6].

Audio Player

Finally, using the audio player, we play the audio file that we've generated. This layer is, again, very flexible. We can think of a wide variety of choices to fill this role. The Audio Player might be a software interface or a hardware device. The only condition is that it should accept and understand the format of the generated audio file. The Audio Player that I've used is provided by the Beads library.

V. Key Trends and Reflection

There are two important trends here that I'd like the reader to notice.

Encoding information in different forms

We are basically trying to communicate. And for communication, we need to encode the information using some protocol. The protocol can be anything we decide upon. It's important to note the parameters which may vary and parameters which will be constant.

The aspects of communication are - there *has* to be a sender, a receiver, a message, a protocol and a transmission medium. There has to be at least one of each of the above mentioned entities. But we're not too concerned about the specifics as we shall see below. I think it's safe to keep the sender, the receiver and the message constant since the goal is to send a given message from a sender to a receiver and it won't make sense to change those aspects.
The aspects which may vary are –

1. The Protocol: We may define our own new protocol.
2. The Transmission Medium: It doesn't matter what the transmission medium is as long as we have one and it can transmit our message.
3. The Encoder and the Decoder used: For example, we could encode the information in light waves and our eyes would decode it. Or we could encode it in sound waves and our ears would decode it. We can choose any encoder and decoder we want as long as they are available at the sender's and the receiver's end respectively.

Compressing information

The aim of my paper is not to produce an audio which just reads everything out loud to the receiver. The aim is to produce an audio which gives some level of insight and intuition into some topic. By listening to the sound on the FriendCluster channel, we will be able to get a good idea as to how many friends a person has and how clustered his network is. But we won't be able to tell exactly(unless you're a computer) how many friends he has or what the clustering coefficient of his network is. We lose some specifics in order to keep the information simple and easy to understand. You can think of this as "compressing" the information. An important decision in designing a system such as this would be to decide upon how much compression of information we need. This will be highly dependent on the context in which the system is being used.

VI. Scope for improvement

I didn't focus much on producing beautiful and amazing sounds as much as I did trying to understand the overall design of the system. Hence, Audio Synthesis could definitely use some work.
My implementation works fine with undirected graphs. This might be okay if you're working with social networks such as Facebook and LinkedIn where person A cannot be connected to person B without person B being connected to person A. But if we're working with sites like Twitter or Google+, we will need to make some changes for the algorithm to work with networks where the underlying structure is a directed graph.
I'm working with a network where there is a limit to the number of connections a node can have(i.e. 5000). If the network doesn't have a limit to the number of connections, a linear function mapping to decide the "base frequency" might not be feasible in some circumstances. We might have to think of mapping it using other types of functions.

VI. Conclusions

Thus, we've managed to synthesize an audio which is unique for a given input and is easy to understand because of the compression of information involved. If you've played around with the GUI demo for this project, you would have also noticed that a large change in the input will change the audio in a way that catches the attention of people(might not be in the most pleasant way. But it does catch the attention of the user).
From the response that I've got from my colleagues and friends to whom I've demonstrated this project, I can understand that they're definitely excited about it and think that it's something cool and completely new. Of course, this technology is still in its infancy. Producing a really revolutionary product using the idea presented in this paper would require an interdisciplinary collaboration. As I see it, in this world where people constantly want new and exciting technologies, implementing this idea on a large scale is worth the trouble. Definitely something to think about, at the very least.

Acknowledgment

This project uses the following libraries/tutorials and I would like to thank the developers of the following libraries/tutorials for providing such excellent resources.

- Java Universal Network/Graph(JUNG) Framework - http://jung.sourceforge.net/

- Beads library - http://www.beadsproject.net/
- Evan X. Merz for writing the book Sonifying Processing: The Beads Tutorial and making it available online for free(as a pdf) http://computermusicblog.com/blog/sonifyingprocessing/
- Christine Alvarado, Mia Minnes and Leo Porter for, firstly, putting together an amazing series of courses at Coursera and secondly, providing the UCSD facebook data that this project uses - https://www.coursera.org/specializations/java-object-oriented

REFERENCES

[1] Link to GitHub repository for the project - https://github.com/prasannals/Echo
[2] Java Universal Network/Graph(JUNG) Framework - http://jung.sourceforge.net
[3] Facebook data scrape related to paper "The Social Structure of Facebook Networks", by Amanda L. Traud, Peter J. Mucha, Mason A. Porter. Source: https://archive.org/details/oxford-2005-facebook-matrix
[4] David Easley and Jon Kleinberg, “Networks, Crowds, and Markets:Reasoning about a Highly Connected World”
[5] Jaccard Index: https://en.wikipedia.org/wiki/Jaccard_index
[6] Oliver Bown,, “Experiments in Modular Design for the Creative Composition of Live Algorithms”, *Computer Music Journal*, Fall 2011, Vol. 35, No. 3, Pages 73-85.