Dave Crosby

IS 520 – Final Project

VBA Final Project: Cover Song Finder

Executive Summary:

These days, the best way for an independent musician to get noticed is on YouTube. On YouTube, a musician can gain subscribers and build an audience to release content to and eventually profit from.

Most musicians want to write and release their own content. The problem is doing this doesn't attract an audience. A musician must first "cover" other popular songs and hope they will then show up in search results of users searching for popular artists. Once an artist is randomly found, if their content is good, a user will subscribe to their channel. Now when the artist wants to release original music they have an audience to receive it.

By analyzing data found on billboard.com, YouTube.com, and other sources, this project is a tool for musicians to quickly identify the songs to cover that will bring the most exposure to their YouTube channel.

The Cover Song Finder

Making a high quality video to put on YouTube can take a huge amount of time and resources. Musicians spend time learning to play the song in a unique way, which can sometimes lead to multiple participants or band members. Then they find a location to film their performance. Professional locations and film crews can be very expensive. Most musicians randomly pick a song to cover only to find out they have wasted time and resources on a song that has brought them no additional exposure.

The best songs to cover are songs that *will* be very popular, but are not *currently* popular, because these songs have increasing search traffic but low competition from other YouTube musicians. This project aims to take the guesswork out of choosing cover songs by analyzing online data and looking at four criteria:

1. How long it's been since the song has been released, or well-known. (30%)
2. How many spots up the song has jumped in the Billboard Top 100 since the previous week. (30%)
3. Amount of competition for YouTube search terms related to the song. (30%)
4. Current popularity of the song. (10%)

VBA Code

The VBA to perform these tasks is one sub procedure that calls 12 separate sub procedures. Each sub procedure is detailed below:

ClearData – Removes all sheets and data from last import in preparation to import new list.

GrabHotOneHundred – Pulls songs from the Hot 100 list from billboard.com and places them in a sheet called "Hot 100 Raw Data" to be analyzed.

JustSongs – Removes all unnecessary data in the "Hot 100 Raw Data" sheet and organizes all the information to insert into tables.

BuildHotOneHundred – Builds a table with hot 100 data, including current rank, last week rank, song name, artist name, peak position, weeks on chart, and a number of other formatting and data inputs.

RisingSongs – Additional data analysis, adds data for how much a song has jumped in a given week

SongComp – External web query using Agent that goes to YouTube for songs 1 to 50, and searches for the Artist and Song Name. The agent then grabs the number of results returned for the search, as well as skips the ads and gets the URL for the actual video.

SongComp2nd – Same as SongComp but for songs 51 to 100. I kept getting an error that said "Out of Memory" or "Automation Error." The errors were not consistent. After much research trying to solve the problem, I finally ended up breaking this step into two sub procedures and I no longer got these errors. I believe so many web queries were overwhelming the program.

CalculateBestSongs – Uses an algorithm to give a score to each song. It assigns 4 scores to each song and adds them up for a final score. The score is based on a 100 point system. There are 30 points allocated for time on Top 100 list (less is better), 30 points for number of spots song has jumped in a week (more is better), 30 points for competition on YouTube, measured by total search results when song is searched (less is better), and 10 points for current rank (Higher is better). Some of these points are allocated as negative and lowers the total score. Each score is calculated as shown below:

$$\frac{\text{Individual Score}}{\text{Total Score}} \times \text{Points} \qquad \textbf{For Example} \qquad \frac{\text{Individual Search Results} - 400{,}000}{\text{Total Results All Songs on List} - 50{,}000{,}000} = 0.008$$

It then takes the result and multiplies by a (-30), so the competition score for this particular song would be –(0.24). A score would then be added in similar format for release date (-30), song jump (+30), and current rank (-10), and then the song would be assigned a final rank. Assuming the impossible scenario that the song had zero search results (no competition) jumped 100 spots while no other songs had jumped any spots, and had just been released last week while other songs had been released for multiple weeks, the highest score a song could get would be 30, but this would be incredibly unlikely. Anything above 5 points would be very good.

TrueRank – Creates an array to assign the final score to each song and then gives it a "True Rank".

BuildCoverOneHundred – Reconstructs the Top 100 based on True Rank, and pulls all data into table into final format, including getting URL's for each YouTube video.

HyperLinks – Makes all YouTube links hyperlinks

CleanData – Gets rid of all sheets that are no longer needed.

Conceptual Difficulties

This project was incredibly difficult for me. I started early after the project was approved and worked many hours to get it done on time. Having no coding experience whatsoever before this class, there

were a number of things that I had to do extra research on or visit with the TA's. The concepts I felt I really mastered while building this are:

1. Loops – This project has tons of loops, and finally figured out how they work. I did so many in this project I don't think I could ever forget.
2. Using the Agent – One of the biggest headaches was trying to use the agent to grab search results and URL. This seems pretty simple, but I spent many hours trying to get this to perform correctly.
3. Variables – I used many types of variables, and feel I much better understand different types of variables and when they are used.
4. Debugging – Many hours were spent debugging each sub procedure. I used all types of procedures to debug, mostly debug.print and using stops to slowly walk through sub procedures.
5. If Statements – Lots of If Statements built into this project, including nested If Statements.

This is certainly not all, but I think I've really gone a long way to master these basic principles.

Missed Elements

There were several elements I wanted to include but was unable to.

1. **Photos** – I really wanted the thumbnail photo for each YouTube video to be used as the hyperlink for the video I tried for many hours and spent lots of time with TA's to figure out how to do this, but it got too complicated. When using the agent to get the URL for each thumbnail, it would return "____Thumbnail____" instead of the actual URL for the thumbnail. I know it has something to do with the tag, but I couldn't figure it out. It was very frustrating and I ultimately gave up trying to include this feature.
2. **Faster Processing** – This is a very slow processing project, mainly because it is doing so many web queries using Agent. I first built it using just the excel web query, but it was so slow I had to go back and redo everything using the agent to make the time to wait bearable. Otherwise the project would be useless to me or anybody else. Even after going back to use Agent, I still wish I could find ways to make it go faster.
3. **Efficiency** - At first I was trying to be very efficient with my variable types, i.e. always using byte instead of integer when applicable, but I kept running into so many errors I just started using standard variables like integer and string for variables for everything. I know if I went back through and really looked at this I could make the model more efficient and have it run faster.
4. **Google Trends** – I tried to include a separate element for the algorithm using Google Trends, but the data was very difficult to get to using web queries, and the project already had so many web queries that I decided not to include anymore.

I had occasional help from the TA's on this, but all the work is my own. The TA wrote some code for me to get URL's, but it didn't end up working out so I started from scratch and figured out how to grab URLs using a different method. Ultimately, this was a great experience and I feel it was a great learning opportunity to master many of the principles we've discussed over the semester.