**FINAL PROJECT 591 L1 - CHECKING YOUR FAVORITE ARTIST'S POPULARITY ON REDDIT**

*by Julia Bighetto and Kyle Bryant*

**PURPOSE:**

To restate what we said during our presentation, we wanted to find out how popular a given artist was in a certain subreddit or in general (through the most popular subreddits). Our main idea was to use Twitter API to see which regions were "more beneficial" for each artist, but since it was hard to get the license for that API we migrated to Reddit and changed our project accordingly.

**METHODS:**

To create this project, we mainly used four libraries: *mwparserfromhell*, *pysimpleGUI, Tkinter,* and the reddit API itself. We then created an algorithm that parses through each comment (gathered by the Reddit API) and checks for the key words we're looking for, each with a specific value. The algorithm then returns either a popularity index [-1,1] or a list of the top 10 artists in a certain genre.

**ANALYSIS OF LIBRARIES:**

* **Tkinter:** tkinter was the easiest library in terms of UI in my opinion. The fun part of tkinter is that it is very straightforward, and you can implement functions within functions (so when you click on a button you open a new window, for example). The bad part of tkinter is it is lengthy and very repetitive. You have to be careful on where and how you declare things, especially if you have more complex windows (like the button that generates another window). It can get tricky. In terms of analysis with this class, it is binding Python to the GUI toolkit without modifying python itself.
* **PySimpleGUI:** PySimpleGUI was harder to work with. It is not as functional as Tkinter, so we couldn't derive a new window from a button, for example. However, it is simpler and easier to use than Tkinter. It's less repetitive and you can see everything in an easier way. It also makes the window prettier (but that's just a plus).
* **Mwparserfromhell:** Mwparserfromhell is a powerful parser that is essential a more straightforward and easy-to-use version of other mediawiki parsers. This allows us to gather strings in the web content easier, which is why it was selected over other Wiki parsers. Using this, we were able to pull just the body of text needed from each Wikipedia page. This is a deep embedding as it has classes specific to MediaWiki but is only compatible with Python.
* **Reddit API:** PRAW is used to scrape reddit.com. This is a relatively straightforward parsing tool for the site which made it possible to search for posts relating to the artist, and then pulling all comments from the post. This requires authorization via creating an app on reddit’s website. This is a deep embedding as it has classes that make up the different components that Reddit has and functions within.
* **Regents:**  This was used to clean up the strings gathered from Wikipedia. This library makes it easy to parse specific values out of strings, in a way that is efficient, and doesn’t require a lot of parsing. This is a deep embedding I believe because it also has classes (but it could be shallow it is unclear from the API).
* **Requests:** Requests library allows you to pull all data from a webpage, specifically for us, Wikipedia. This is a simple library that we only used to pull webpage html. This is a deep embedding as its functions are based off classes specific to the library.
* **Itertools:** Itertools is a library focused specifically on list functions requiring lots of iteration. This was used to chain together the artists name with the list of comments, which was needed for a few of our methods. This is a deep embedding as it has classes of functions.
* **CSV:** CSV was used to store the artist names and word association scores, and then later used to fetch this info to use in the code. This was mostly used with transferring of information to other files. This is an external embedding as it is outside of python, this library simply makes it compatible.
* **JSON:** JSON was used to pull the website info from wikipedia in a specific way (according to parameters). JSON is used for querying and database scraping, which makes sense for it to be used here. This is an external embedding as it is a unique thing, and the library being used allows it to be used within Python.

**HOW TO RUN THE CODE:**

There are a lot of files for this project, but the main one is called final\_project.py located on the folder called python\_scripts. To run the code, just keep all the files from this folder in the same directory, since one relies on the other.

**FURTHER COMMENTS:**

* We rely on good grammar from the user and reddit in general. Our code won't recognize a misspelled word.
* We decided to hard code a few things to make it easier for others to run on their computer. For example, we could have made the dictionary of positive/negative words a dynamic process, where the program takes a spreadsheet (in which the user can alter) and read it. However, to make it less error prone, we just hardcoded the dictionary in a python file.
* For part 3 and 4 it takes a long while to run, since we have a lot of artists per genre and each artist has its own iteration through the algorithm with hundreds of comments. The iteration is there and the algorithm is constructed, but we couldn’t test that much. A solution for this would be to, in the future, scrape other websites (like <https://www.billboard.com/>) to find dynamically the top 20 artists of a certain genre and search those artists up. But it would still take a long time to make this calculation.