Karol Vargas
Austin Bristol
David Perez
Francisco Guzman

Final Report

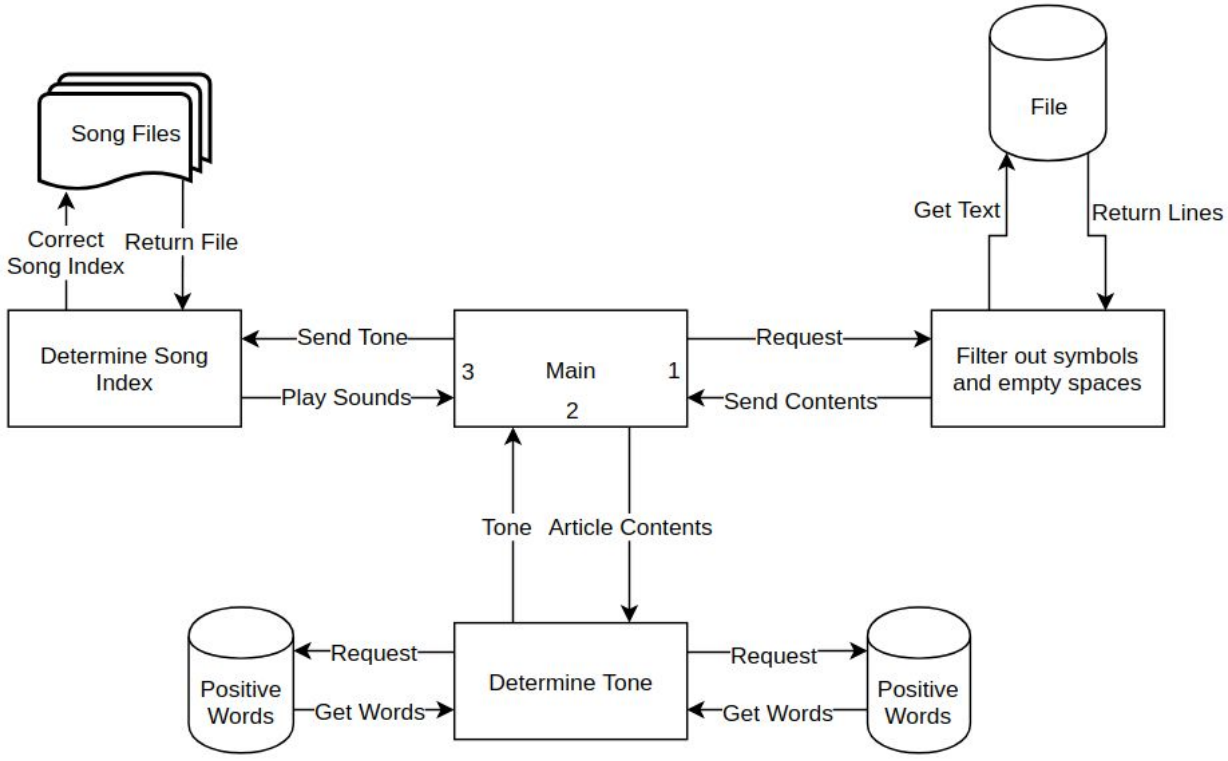
Our motivation was to create a project which could have an impact on how we read the news on a daily basis. A way to get an idea as to what kind of news something is before you read it all the way through and decide for yourself. Using Sentiment analysis was another motivation of ours as we thought it was interesting how you can analyze sentiments using python. This type of project would also give us an opportunity to work on using different methods we learned this semester in python while simultaneously making something useful. We all love to listen to music and what better way to mutate that love for music into something we can use in our day to day lives. This can also be a way for people to learn a new language as well. Our program performs sentiment analysis on a news file in txt format and if we were to apply those same concepts to an algorithm which would return a sound based on what an articles tone is in a different language. We can disrupt an industry of people learning new languages. Creating a tool which could be used to help make peoples lives easier is the goal of anybody in the computer science industry.

The reason why there is not a whole lot of output is that most of the programming done for this program is done on the back end. One of the first challenges we faced was finding a library which would allow us to play .wav files from the terminal. This proved to be much more challenging then we could've presumed but we decided to use pygame. It had a very easy to understand API which usually ends up being what holds anybody back from picking a library and we collectively decided this would be the best option. One of the next challenges we faced was how we were going to store the news files we planned to parse online. We decided that to avoid complication we would download the news file from the html document online as a .txt file. This would allow us to successfully parse through the file seamlessly. Next we had to decide on what words contained emotion or sentiment. From previous classes we decided we need to take out all stop words from the file.

Next, we will go over each part of the code and explain the process that the program goes through. The first thing we do is establish the total positive and negative tones that we will be adding to as the program runs. Then, we will actual get into a list, each word from the article. To do this, we call a function which will iterate over each line in the article's text file and split it by word. After we got each individual word, we do some transformations. We take the word and pass it through some regex code which filters out everything except the raw word. This raw word will then be passed into a function which determines whether or not it is a stop word. That is if it is a common word that doesn't hold any positive or negative meaning. If it is a stop word, we filter it out. After this portion is done, we return a list of words from the article that does not have any numbers or symbols as well as no words that are stop words. Next, we iterate over every

word in the article that we just collected. On each iteration, we will call two functions to get the positive and negative words. These positive and negative word objects are generators, so whenever we need to get a positive or negative word, we get it from the file rather than storing the thousands of words in a list.

After getting all of the positive and negative words, we check if the current word from the article is positive or negative. If it is, we increment the correct tone by one. After we have checked all words from the article, we have a numerical value for positive and negative words. Thus, we pass these values into a new function to actually play the sound. We take the tones we collected and determine the average positive tone for all of the words in the article. Then, we take this average tone and multiply it by the total amount of sounds we have to play. This means that this is an index of the song file that we want to play to represent the article's feeling. So, as we list our song files from left to right, they get increasingly happy. So if one hundred percent of the words are positive, the last index will be played which is the most happy. Finally, we play this song that we have picked using the Pygame library. This overall structure of our code can be seen in the flow chart below, first we get the filtered file contents. Then we determine the tone with positive and negative word files. Finally, we take the tone and play the song files. This process can be seen in the figure below



Our results may not look like a lot on paper at the moment simply because of how difficult it is to visualize the sound the program plays. The end result of the program is to basically take out all stop words from an inputted .txt file with words that can be parsed over

with sentiment analysis. After the code is done taking out the stop words for every negative word which can be matched to a .txt file with a ton of negative words the score will go down by 1 and for every positive word the score will go up by one. Depending on what the absolute value is of the final score the music player will use that score as the index then to choose a .wav file which will play a song of either positive or negative tone. We stored the files so that the higher the score the higher or more positive the score is and in turn the more positive the note was towards the end of the list. To choose the files we decided that it would be best to download chords online to use as little of the users time as possible. This would allow for a quick tone decide what the articles tone is and in that instant the user can decide what it is that they want to read. Below is our terminal output.

The biggest rewards I feel we felt as a team were when we heard the music play for the first time after installing the pygame library. We were all getting extremely frustrated each doing our own research to find out what we could play music with and having so many different libraries either have no documentation or just not being feasible to use. Another of our biggest rewards came from when we found out where we were going to pull our sound from. It was too difficult to try and find chords which were positive or negative from youtube since there was simply to much until we found sound.com. This ended up being or saving grace in this issue because the files were easy to download, edit, and implement as .wav files. The teamwork involved was as follows Austin was the most familiar with sentiment analysis and eliminated the stop words and parsed over the .txt files with the functions he made. Francisco and David found out how to turn the news pages into .txt files. Karol handled the sound of the program and setting up the lists which would be parsed over with the sentiment analysis functions.