Assignment 5 Software Design Document

CS2300 Section 1 Spring 2022

Landon Bunker

# Project Description

Part 1) Create the algorithm for the simple page rank and output the eigenvector and rank of the pages by using the power method.

Part 2) Perform linear binary classification by using the training input file to set the weights and then test an input file to determine whether a feature is recommended or not.

# Approach

For part A, I first put all the input data into a NumPy array so that I could get the transpose of it. Once I got the transpose I would make sure all the columns would add up to 1 and check for negative numbers in the input stream. I then would implement the power method and after this would print the eigenvector and the ranks to the output file.

For Part B, I also put all the input data from the training file into a NumPy array. I then used multiple for loops to index into input and calculate the error. From the error I would then get the weight and in turn get F. Once we got F we could test f with the weight array and then put a 1 or 0 in the classification array. After this we need to output the classification array and the weight array to a file.

# Detailed Design

## Programming Language

The programming language I used was Python because it has lists that can be dynamically grown or shrunk and can contain any data type. Python is also helpful because of the NumPy library which is helped to store arrays and perform rounding on them.

## Modules

I only used two modules in the program which broke the program up into Part A and Part B. Part A has all the calculations needed for the page rank algorithm and Part B has the needed calculations to determine the classification array and the weight array.

## Flowcharts

Diagram

Description automatically generated

## Key Data Structures

The key data structures that I used were lists to hold the various guesses and weights. I also used arrays in the NumPy library so that you could round the output when you write it to the file.

## Test Description

For part A, I used three different test files so that I could test to make sure that the columns had to add up to 1. I used another file to test and see if negative numbers would also trigger the invalid input. Lastly, for Part A, I used the input file that was given to use to make sure that the program would run correctly. For Part B, I used the original training input so that we could get the correct weight array. I then created a second Part B test file to test that all the parts were running properly.