Technical Report

Sequential Page Rank

SequentialPageRank takes input in the form of a file containing URLs represented as integers and their outbound URL-integers, assigns them a pagerank value, and writes the URLs with their pagerank values in decending order to an output file. To do this, it uses four main methods:

1. parseArgs(String[] args): this method parses the command line arguments and updates the corresponding instance variables. These are the input file to be read from, the name of the output file to write to, the number of times the algorithm iterates over the pagerank values, and the damping factor. If the user doesn’t provide the correct number of arguments, the program terminates, informing the user they provided the incorrect number of arguments.
2. loadInput(): this method reads the input from inputFile and populates the adjacency matrix. An error is thrown if the inputFile cannot be traced or does not exist. It first constructs a Scanner-BufferedReader-FileReader that takes inputFile as its parameter. For each line read, it creates a new key value from the first number in the line and puts it in an intermediary hashMap *outbounds* that represents the outbound link adjacency matrix. Each successive number in the line is stored as the value of the key in an arrayList. Then the entries in *outbounds* are looped over to fill *adjMatrix* which maps a URL to the URLs that link to it. Each integer in the current iteration’s key points to the key in *adjMatrix* where the current key is added as a value into the arrayList of the *adjMatrix*’s key. If the current key has no outbound links (signified by an empty values arrayList), then that key is added as a value to every key within *adjMatrix.*
3. calculatePageRank(): this method does a fixed number of iterations and calculates the page rank values of each URL stored in *adjMatrix*. The formula used to do this calculation is

where d is the damping factor, N is the number of URLs in *adjMatrix* (stored as the local variable *N*), PR(v) is the current iteration’s pagerank value of a given inbound URL, and L(v) is number of outbound links of the given URL. Initially, all URL’s pagerank values are set to 1/N and stored in an intermediary hashMap *interValues*. For each iteration of *calculatePageRank*, the entries in *adjMatrix* are looped over. For each entry, the arrayList stored as the value of each entry is looped over, and the ratio of each inbound URL’s PR(v) (obtained from *interValues*) to its L(v) (obtained from the length of the arrayList in *outbounds* for the given key) is calculated and added to that entry’s *sum*. The rest of the pagerank value is calculated using *sum* and is then stored as a value in the hashMap *rankValues.* Once each URL’s pagerank value is calculated, all values in *rankValues* are transferred to *tempValues*, *rankValues* is cleared, and the process repeats. Only on the last iteration does this final transfer and clearing not take place.

1. printValues(): this method sorts all URLs according to their pagerank values in decreasing order, writes the pagerank values to an output file, and prints the first ten values to the console. The pagerank values are sorted using an implementation of mergesort. This private method converts all values in *rankValues* to an arrayList of length-two arrays that contain the URL and its pagerank value. This data structure is then passed through mergesort, and a new arrayList *combined* stored the result of this sorting. Another private method *print* converts the values in *combined* a String *str.* Next, a new file is created using *outputFile*, and a BufferedWriter-FileWriter writes *str* into the new file. Finally, *combined* is iterated over ten times, each iteration printing out a URL-pagerank value string.