CS 2302

Fall 2019

Lab Report #4

John Rodriguez

Due: October 21, 2019

Professor: Olac Fuentes

TA: Anindita Nath

Introduction

For lab 4 we were given the task to compare the running times of the two data structures, Binary Search Tree (BST) and a B-Tree when retrieving the word embeddings to compare two words. The purpose of this lab is to make us comfortable when working with BSTs and B-Trees and also learn how Natural Language Processing (NLP) works.

Proposed Solution Design and Implementation

I approached this lab by first creating the BST and B-Tree classes. I also included the insert function for a BST and the several functions needed for insertion in a B-Tree. Then I added the WordEmbedding class which allowed me to nodes of a WordEmbedding object which contains the word and it’s embedding which can both be accessed for the node.

The next step I took was implementing functions that can get the height of a BST, get the number of nodes in a BST, and finally search a BST for a specific word. After implementing these functions, I then created these same 3 functions but to work on a B-Tree. Then I created a function called OnlyLetters which is used to check if a string only contains letters so when I create the Trees using the glove.6B.50d.txt file only words will be stored and not any of the symbols included in the file. The finale two functions I created are called words\_diffrence and words\_diffrence\_BTree which both return a number showing the similarity between two words. I also discovered I had to make small modifications to all the functions so they would work with the nodes of WordEmbedding objects.

The next part of the program I worked on was the menu for the user which would output the message for the user to choose which tree they want constructed. The user could enter 1 to create a BST and 2 to create a B-Tree and I also added a else condition that would tell the user to enter 1 or 2 if they entered a different input.

Then the next part of the program I worked on was if the user selected option 1 to create a BST using the file glove.6B.50d.txt. The first thing I did was open the glove.6B.50d.txt. read the file line by line and separate the information on that line by the space between them. I then would store the first item in the row and store it in a variable called word and use the function OnlyLetters to check if the string stored only contained letters. If it did then I would use the Insert\_BST function to add the word and it’s embedding to the tree using the WordEmbedding class to create an object holding the word and it’s embedding. While doing this I measured the running time to construct the BST and stored it in the variable run\_time. I then output the number of nodes in the tree using the function Size\_BST, then output the height of the tree using the function Height\_BST, and finally output the running time it took to construct the BST.

I then worked on the part of the program I worked on was if the user selected option 2 to create a B-Tree using the file glove.6B.50d.txt. The first thing I did was open the glove.6B.50d.txt. read the file line by line and separate the information on that line by the space between them. I then would store the first item in the row and store it in a variable called word and use the function OnlyLetters to check if the string stored only contained letters. If it did then I would use the Insert function to add the word and it’s embedding to the tree using the WordEmbedding class to create an object holding the word and it’s embedding. While doing this I measured the running time to construct the B-Tree and stored it in the variable run\_time. I then output the number of nodes in the tree using the function NumItems, then output the height of the tree using the function Height, and finally output the running time it took to construct the B-Tree.

The next part of the program I worked on was to read from a text file I created called “word file.txt” which holds pairs words on each line separated by a space. I first read the file line by line separating the information on the row by the space between the information. I then stored the first string in the variable word1 and the second string in the variable word2. I then used the words\_diffrence\_Btree to output the similarity of the words if it was searching for the words using a B-Tree and words\_diffrence if it was a BST. I then recorded the running time it took for the trees query processing and outputted it. I then added this to both options that the user can select from.

Experimental Results

**Option 1: Binary Search Tree**

I tested my program by selecting option 1 to construct a BST using information from the “glove.6B.50d.txt” file and then search for words from the “word file.txt” in the BST to find the similarity between them.

A screenshot of a cell phone

Description automatically generated

A close up of text on a white background

Description automatically generated

A screenshot of a cell phone

Description automatically generated

**Option 2: B-Tree**

I tested my program by selecting option 2 to construct a B-Tree using information from the “glove.6B.50d.txt” file and then search for words from the “word file.txt” in the B-Tree to find the similarity between them.

A screenshot of a social media post

Description automatically generated

A close up of text on a white background

Description automatically generated

A screenshot of a cell phone

Description automatically generated

**Running Times Table:**

|  |  |  |
| --- | --- | --- |
| Tree Type | Running Time Tree Construction | Running Time Query Process |
| BST Test1 | 10.09720754623413 seconds | 0.031238079071044922 seconds |
| BST Test2 | 9.914376974105835 seconds | 0.017843961715698242seconds |
| B-Tree Test1 | 12.657171964645386 seconds | 0.015622138977050781 seconds |
| B-Tree Test2 | 15.335143327713013 seconds | 0.031242847442626953 seconds |

Conclusion

This lab helped me become more comfortable when using B-Trees and BSTs. I also found that the BST was able to be constructed faster than the B-Tree based off their running times. The query processing running time for both trees was roughly the same. I also learned a bit more on how Natural Language Processing works after using a simplified version in this lab.

Appendix

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class