CS2302 Lab3 Report **Govinda KC**
Prof. Diego Aguirre Class 10:30 am
TA: Manoj Saha Date: 11/06/18

Problem Description

This lab corresponds to develop the skills on AVL tree, RedBlack Tree and in word embeddings. Word embeddings are the recent advance in Natural Languate Processing (NLP) that consists of representing words by vectors in such a way that if two words have similar meanings, their embeddings are also similar. In this work, I have given a big text file 'glove.6B.50d.txt' which is obtained from https://nlp.stanford.edu/projects/glove which contains words and embedding vectors of various lengths for various vocabulary sizes. Each line starts with the word being followed by the 50 floating point numbers that represents the words vector description. I have to find the similarity between the words in terms of cosine distance. Also, I have to calculate the number of nodes, desired depth of keys and also need to create the files that contain the words stored in tree in ascending order and keys of certain depth asked by user.

Proposed Solution

Since the original text files also contains the words that don't start with alphabetic characters, I have removed them in order to make things more comfortable as instructed. As per the applications of the AVL tree and RedBlack tree in this lab, I have adopted the zyBook's code as suggested and wrote the code for the implementation of the cosine similarity in order to study the relationship between the pair of words through out the whole text document. Number of the nodes can be counted and also depth of the tree desired by the user will be implemented to measure. User will be prompted to choose either to use the AVL tree or RedBlack tree and also for the depth.

Test of Algorithms

In order to test if the algorithm works perfectly, I ran the codes for each solution multiple times. User is asked to choose either AVL or RedBlack tree and on the basis of that it will print out the results for the number of nodes and height as well. A file containing all the words stored in the tree in ascending order is generated.It will also ask the user to input the number to print out the desired depth. Another file will also be generated containing all the keys of that depth. Most important thing of this lab is to measure the cosine similarity between the words, which is nicely implemented using the relation of the cosine similarity. It is basically the cosine angle with range from -1 to 1. Words that are embedded in terms of embedding vectors are taken under calculation. But I have only measured the similarities between the words that are being asked. I can clearly see the relation measured in between -1 to 1. The value close to 1 tells that this pair of the words are highly related to each other and the value close to -1 tells that they are completely opposite to each other in terms of their relation. If the value is 0, that means they are not related. Output is the following as shown below.

Big-O running Time

RedBlack Tree	AVL Tree
19.43 seconds	27.76 seconds

Although both of them are self balanced trees with the Big-O running time

$$O(log(n)).$$

I found that time taken by the AVL is little longer. This is because RedBlack trees are found to work better on insert, addition, or remove.

Conclusion

From this lab, the relation between the pair of words are studied for large number of words that are embedded in terms of embedding vector. The cosine similarity is measured in terms of cosine distance between the words. From the results we see that harvard and stanford are closely related as their cosine distance is 0.84 . In the other hand harvard and ant are not related at all since the value is -0.026, they are rather opposite to each other. I have compared the theoretical and experimental results and they are consistent to each other.

Appendix:

```
# Govinda KC
# CS 2302 lab3
**************************************************

################################################################################
# Govinda KC
#
# CS2302 Lab 3
#
# Instructur: Diego Aguirre
#
# TA : Manoj Saha
#
################################################################################
# The purpose of this lab is to find the relationship between the pair of v
# The value of cosine angle is from -1 to 1. The value close to 1 tells tha
# each other while close to -1 means that they are completely opposite to e

# Importing the required Modules.
import math
import AVLTree
import RedBlackTree
from AVLTree import AVLTree
from RedBlackTree import RedBlackTree
```

```
from AVLTree import Node
import time
start_time = time.time()
# Function to read a given file.
def read_file():
    f = open('glove.6B.50d.txt', encoding="utf-8")
    line = f.readline()
# Creating the array, embedding and node.
    while line:
    # Splitting the line
        _line = line.split(" ")
        # Checking if the first letter is an alphabet.
        word = _line[0]
        if word[0].isalpha():
            # initializing the array as an null.
            embedding_array = []
            for j in range(1,len(_line)):
                embedding_array.append(float(_line[j]))
            node = Node(word, embedding_array)
            # Using try and Except to insert the word in node
            try:
                tree.insert(node)
            except:
                tree.insert(word,embedding_array)
            # Go to next line
        line = f.readline()
    f.close()

# Give the options to user RedBlack or AVL Tree?
while True:
    _input = input("Type 0 for Red-black tree and 1 for AVL tree: ")
    if _input is not '0' and _input is not '1':
        print("Wrong input" )
        continue
    else:
        break
# This is for RedBlackTree when user chose 0
if _input is "0":
    tree = RedBlackTree()
    # Function will be called to read the file.
    read_file()
    # This line prints the nodes count and height in total
    print("RedBlack Tree has "+ str(len(tree)) + ' nodes')
    print('and')
    print("It's height is " + str(tree._height()))

    # Writing all the words into a single file
    output_file = open("RedBlack_tree.txt", "w+", encoding = 'utf-8')
```

```
    tree._write()
    output_file.close()

    # Gives the depth of nodes of user's choice.
    while True:
        _inputuser = input("Please enter the depth of nodes you would like
        # Checks if the input is valid for the tree.
        if int(_inputuser) >= int(tree._height()) or int(_inputuser) < 0:
            print("Depth is not valid, please choose another depth size" )
            continue
        else:
            break

    # Creating file for depth
    k=int(_inputuser)
    depth_file = open("RB_depth.txt", "w+", encoding="utf-8")
    #k=int(_inputuser)
    tree._depth(k)
    depth_file.close()
    print()
    print("---%s seconds---" % (time.time() - start_time))
    print('**********************************************************

# This is for AVL Tree  when user chose 1
if _input is "1":
    tree = AVLTree()
    read_file()
    # This line prints the nodes count and height in total

    print("AVL Tree has "+ str(tree._size())+' nodes')
    print('and')
    print("It's height is " + str(tree._height()))

    # Writing all the words into a single file
    output_file = open("AVL_tree.txt", "w+", encoding = 'utf-8')
    tree._write()
    output_file.close()

    # For depth
    while True:
        _inputuser = input("Please enter the depth of nodes you would like
        print()
        # Checks if the input is valid for the tree.
        if int(_inputuser) >= int(tree._height()) or int(_inputuser) < 0:
            print("Depth is not valid, please choose another depth size: ")
            continue
        else:
            break
```

```
    # Creating file for depth
    k=int(_inputuser)
    depth_file = open("AVL_depth.txt", "w+", encoding = 'utf-8')
    tree._depth(k)
    depth_file.close()
    print()
    print("---%s seconds---" % (time.time() - start_time))
# Read the given file to find the relations of the words interms of cosine
f = open('appendix.txt')
line = f.readline()

# Cosine similarity calculations
while line:
    # spliting the line and creating array
    _line = line.split(" ")
    # searching and assigning the nodes
    w0 = tree.search(_line[0])
    w1 = tree.search(_line[1])
    if w0 is None or w1 is None:
        print('no comparison is found')
        # This section computes the angle of similarity between the two wo
    else:
    # Now Measure the cosine similarity angle between two words.
    # Initialization of terms
        dot_prod = 0
        magnitude_0 = 0
        magnitude_1 = 0
        e0 = w0.get_embedding()
        e1 = w1.get_embedding()
        for i in range (len(e0)):
            # dot product between two embedding vectors
            dot_prod+= e0[i]*e1[i]
            magnitude_0 += e0[i]*e0[i]
            magnitude_1 += e1[i]*e1[i]
        magnitude_0 = math.sqrt(magnitude_0)
        magnitude_1 = math.sqrt(magnitude_1)
        magnitude_0 = magnitude_0 * magnitude_1
        # Relation for cosine similarity measurement
        cosine_similarity = dot_prod/magnitude_0
        print(_line[0],"",_line[1],"", cosine_similarity)
    line = f.readline()
```

****

A signed academic honesty certification: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.