CS32 Project 4 Report

* Arsh Malik
* Finished all implementations.
* Used only allowed STL containers in all the classes.
* No bugs that I know of.

Test cases for various classes:

**RadixTree**

rt.insert("slow", 1);

rt.insert("slowe", 2)

rt.insert("slowest", 3);

These inserts check the order in which radix tree is built. Each new word must join as chain under the existing word in the tree.

rt.insert("slower", 1);

rt.insert("slow", 2);

rt.insert("slo", 3);

These statements check if the existing words break into subtrees as the incoming words are entirely their prefixes.

rt.insert("water", 1);

rt.insert("waste", 2);

rt.insert("watch", 3);

rt.insert("wat", 4);

These inserts check for branching into subtrees when the prefix is a subset for both the existing word and the new word being added.

RadixTree<int> rt;

rt.insert("test", 1);

rt.insert("toaster", 2);

rt.insert("toasting", 3);

rt.insert("slow", 4);

rt.insert("slowly", 5);

int\* x = rt.search("slowly");

cerr << \*x << endl;

I used these tests to check the search function in the radix tree.

**AttributeTranslator**

AttributeTranslator ab;

ab.Load("/Users/arshmalik/Documents/Projects/CS32-Projects/Projects/Unhinged/Unhinged/translator.txt");

vector<AttValPair> test;

AttValPair source("trait","transparent");

test = ab.FindCompatibleAttValPairs(source);

for (int i = 0; i!= test.size(); i++)

{

cerr << test[i].attribute << endl;

cerr << test[i].value << endl;

}

I used the above code to test my attribute translator class. ab.Load loads the file. I create an AttValPair source and use ab.FindCompatibleAttValPairs for source that returns a vector of AttValPairs. I then loop through the contents of the vector to ensure I got the expected result.

**MemberDatabase**

MemberDatabase mdb;

mdb.LoadDatabase("/Users/arshmalik/Documents/Projects/CS32-Projects/Projects/Unhinged/Unhinged/members.txt");

AttValPair prof("job","accountant");

vector<string> emails = mdb.FindMatchingMembers(prof);

cerr << "--------------------------" << endl;

for (int i = 0; i!=emails.size(); i++)

{

cerr << emails[i] << endl;

}

I create a member database object called mdb. Using the LoadDatabase method, I load all the contents of the file provided. Then, I create an AttValPair and find its matching members using the FindMatchingMembers. To ensure both the methods worked, I use a for loop on the returned vector and cerr the emails I got.

**PersonProfile**

Since I use PersonProfile in the MemberDatabase class, this class is indirectly checked in the above test. Hence I wasn’t required to write a separate test case for this class.

**MatchMaker**

**The following test is used to test MatchMaker**

bool findMatches(const MemberDatabase& mdb, const AttributeTranslator& at)

{

// Prompt for email

std::string email;

const PersonProfile\* pp;

for (;;) {

std::cout << "Enter the member's email for whom you want to find matches: ";

std::getline(std::cin, email);

if (email.empty())

return false;

pp = mdb.GetMemberByEmail(email);

if (pp != nullptr)

break;

std::cout << "That email is not in the member database." << std::endl;

}

// Show member's attribute-value pairs

std::cout << "The member has the following attributes:" << std::endl;

for (int k = 0; k != pp->GetNumAttValPairs(); k++) {

AttValPair av;

pp->GetAttVal(k, av);

std::cout << av.attribute << " --> " << av.value << std::endl;

}

// Prompt user for threshold

int threshold;

std::cout << "How many shared attributes must matches have? ";

std::cin >> threshold;

std::cin.ignore(10000, '\n');

// Print matches and the number of matching translated attributes

MatchMaker mm(mdb, at);

std::vector<EmailCount> emails = mm.IdentifyRankedMatches(email, threshold);

if (emails.empty())

std::cout << "No member was a good enough match." << std::endl;

else {

std::cout << "The following members were good matches:" << std::endl;;

for (const auto& emailCount : emails) {

const PersonProfile\* pp = mdb.GetMemberByEmail(emailCount.email);

std::cout << pp->GetName() << " at " << emailCount.email << " with "

<< emailCount.count << " matches!" << std::endl;

}

}

std::cout << std::endl;

return true;

}

**Explanation:**

In the first part of the code, I retrieve the PersonProfile from the member database using the provided email.

Then, I print all its attribute value pairs.

Next, I prompt the user for the threshold. I use my MatchMaker’s IdentifyRankedMatches method to retrieve a vector of EmailCount objects and display them using a for loop

**Note:** For threshold <= 0, I display Persons with matches >=1.

**Utilities**

I have also implemented a comparison function that is used in MatchMaker’s IdentifyRankedMatches method to sort a vector of EmailCounts using the sort method provided in algorithms library.

Thank you