Ajuna Kyaruzi & Alex Mitchell
CSC 301
Programming Assignment 3

Summary

StepWise is an AI Math Tutor that requires a parser for student input into what method the student claims to be using to solve their problem. Given a set of methods that StepWise supports, our solution takes those methods and computes a document of all the possible one-letter typos that are related to that input, pre-processes it and adds it to our dictionary structure. Using this as a base, student input is matched against our dictionary and given a probability of which methods the students might have been referring to.

Assumed Inputs

While we only support 6 functions and methods, the input file of keywords should have the following format where each keyword is separated by four spaces and the different intentions that are associated with that word. The list below is the complete list for the 6 beginning methods we have. The starting file is provided with the code. Please provide the name of the input file as command line input when prompted.
Sample input file is provided as “EXAMPLEINPUT” in the accompanying code and documents.

Algorithm

The goal of our solution is to frontload as much computation as possible so that the runtime is as quick as possible. By generating a long list of possible phonetic and typo allowable variations of given keywords, we can load these into an efficient Trie-based dictionary. Although this requires us to have a good understanding of what kind of errors students make, machine learning on denied allowable variants could improve allowable keyword generation.
A sample run is provided as “EXAMPLE” in the accompanying code and documents. This run includes providing all input, examples of student input and our solution’s output.

Resources

Not currently using Open Source Resources. A list of proposed resources are supplied below.

Restrictions

Currently our solution doesn’t support more than one typo at a time and it doesn’t yet support phonetic typos. While we have identified possible open source libraries that can help with this but didn’t have time to implement them. Furthermore we should look into supporting conjugations for our verbs. Another thing is our code in partially in Swift which has Apple proprietary code and needs Mac OS.

- https://commons.apache.org/proper/commons-codec/apidocs/org/apache/commons/codec/language/DoubleMetaphone.html
- https://github.com/KevinStern/software-and-algorithms/blob/master/src/main/java/blogspot/software_and_algorithms/stern_library/string/DamerauLevenshteinAlgorithm.java