COS126: Final Project Report.

Fall 2022

Instructions: Make a copy of this Google doc and respond to all required questions. Then, download this Google doc as a PDF and change its name to `report.pdf`. Finally, upload `report.pdf` to TigerFile.

1. Basic Information

Name #1: Bryson Jandwa
NetID #1: bj7218

Name #2: Jenny Fan
NetID #2: jf8083

Project Advisor: Maya Gupta
Link to YouTube video presentation: https://www.youtube.com/watch?v=qTd2_mfXtbU

2. Project/Course Feedback

***Before moving on**, fill up the following form (https://forms.gle/MSQC4cswCJ3PBszY8) to provide feedback on the project and the course.*

We filled up the feedback form: (x)

3. Project Description

What is your project title?
Edit Distance Algorithm

Please summarize your project in 1 paragraph. Only describe things you implemented.
This project is about the edit distance algorithm, which calculates the distance between two strings based on the edits needed to transform one string into another. This project seeks to implement this algorithm using the Levenshtein distance, along with the creation of our own distance metric based on distances between keys on the keyboard. This is then applied to a document search, a user can choose a document and give a reference word, and the program can output all of the near-matches and their count, as well as produce two visual displays of all the distances between the reference word and the words in the document.

Please list all the relevant files (.java files, datasets, .jar libraries) from your project. Include a brief explanation of their contents.

1. EditDistance.java - contains the methods to calculate the Levenshtein edit distance given two strings and the keyboard edit distance using the metric we came up with
2. DocumentSearch.java - a user-interactive client where the user can choose a document, give a reference word, and choose out of three methods: find all of the near matches in the document, draw a histogram, or draw a visual map.
3. Tests.java - tests all of the methods in the above java files
4. keyboard.txt - contains a weight table we made for substituting between letters based on distances between letters on the QWERTY keyboard
5. dream.txt, raven.txt, etc. - 5 files containing the number of words in the document on the first line and the document after, one of which is very short and used for testing

Describe instructions on how to compile and run your project (e.g. compilation commands).
Compile: javac-introcs DocumentSearch.java
Run: java-introcs DocumentSearch

4. Features and Project Requirements

For each of the following, mention the .java files, classes, and methods that implement the things you are describing. If you used any external libraries, mention those as well.

Feature #1: Implementation of the Edit Distance algorithm

In 2-3 sentences, please describe your feature. Only describe things you implemented.
This feature implements the edit distance algorithm in Java (https://en.wikipedia.org/wiki/Edit_distance#Common_algorithm). Given two strings, it outputs the Levenshtein distance between the two strings. It also implements an edit distance we designed based on distances between keys on the QWERTY keyboard by taking in a weight table to change the cost for substituting between close and far letters.

In 1-2 sentences, please describe how you tested your feature.
We used known examples and small cases we confirmed by hand to test that the methods produce the correct results.

What was the agreed category of your feature using the buckets paradigm?
( ) Standard
( ) Sprinkle
(x) Sparkle

Feature #2: Visual Output

In 2-3 sentences, please describe your feature. Only describe things you implemented.
This feature is a visualization of the distance from every word in a chosen document to a reference word given by the client. The user can choose either to use the Levenshtein distance or the keyboard distance. The first visual is a circle of plotted points connected by line segments where each point, in polar coordinates, has angle corresponding to the fraction of the document the word is at and radius equal to the distance from the word to the reference word. The second visual is a histogram of the number of words for each distance. We decided to do both because the first is visually appealing but not as useful for analyzing, while the second is visually less appealing but more useful for analyzing.

In 1-2 sentences, please describe how you tested your feature.
We used a single method that takes the results from the edit distance implementation above and displays it using StdDraw.

What was the agreed category of your feature using the buckets paradigm?
( ) Standard
(x) Sprinkle
( ) Sparkle

Feature #3: Near Match Document Search

In 2-3 sentences, please describe your feature. Only describe things you implemented.
This feature is a near-match document search, where in a chosen document and a user-given reference word, the output is the set of all words in the document that have a distance less than a certain threshold to the reference word based on the keyboard edit distance, as well as the count of these matches. The distance metric is chosen by the user.

In 1-2 sentences, please describe how you tested your feature.
Using a small sample of text and words that we checked by hand.

What was the agreed category of your feature using the buckets paradigm?
(x) Standard
( ) Sprinkle
( ) Sparkle

Other Project Information:

What would you try to add to your project if you had more time to implement extra features?

- Creating a spectrum of colors to match the visual element (both bars and lines).
- Other distance metrics that use weight tables for insertion, deletion, and substitution, possibly also add transposition

What did you learn by working on this project?

- More about the edit distance algorithm in general, like the different metrics for measuring distance
- Visual output with StdDraw - particularly with creating the histogram; also manipulating symbol tables and nested symbol tables
- Taking into consideration the allocated time when deciding the size of the project.

What advice would you give to students next year, to help them have a better experience working on their projects?
Don't procrastinate. Attempt a reasonable size project.

5. Extra credit

You should use this section to mention any of the extra credit items you did. Please provide justification for each one (for example, for the version control item you should have a link to your project on a website like GitHub)

(a) It's a partner assignment.