EECS 349 Fall 2015
Jason Brown - Homework 2

1a) A feature set for a facebook account could be $x_i = < x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5} >$ such that x_1 represents number of friends, x_2 represents height in inches, x_3 represents user-id, x_4 represents state she is from mapped by state onto an index of [0,50), and x_5 represents hair color in her profile picture mapped onto an index of [0,5) for (brown, blond, red, black, and bald). Determining the hair color would necessitate using an advanced visual computer graphics learner that finds eyes in a profile picture and can determine where the hair would be in the image based on that. x_1 can take natural numbers up to 20,000, x_2 can take natural numbers up to 108, and x_3 can take natural numbers up to 2 billion (1.5B facebook users). The features were picked because i) they would facilitate classification of user, ii) they are representable as real numbers, and iii) it is unlikely/impossible that any two users would have the same feature set.

1b) The metric chosen over this feature set is x_3, user-id, as proven below:
Reflexivity: the distance between two user-ids is 0 only if the user ids are for the same facebook user. Non-negative: the distance between the two natural numbers is always positive, like in geometry distance. Symmetry: the distance between user x and user y is the same as the distance between user y and user x; the user-ids will be represented as natural numbers, and distance here will be defined as the absolute value of the difference between the two user-ids. Triangle Inequality: Since the feature attribute can be represented as a natural number in the range [0,2 billion), the distance between any two user-ids a and b will always be $\leq$ than the sum of the distances between (user a, user b) and (user b, user c).

1c) Range of *height in feet*: (0,9); range of *weight in kgs*: (0,150); *number of hairs on head*: (0, 3 million); the most important determinant of this classification would be height and weight which should be weighted more importantly; number of hairs on head should be scaled down to fit a range of (0,150), the same as the weight attribute. A metric could look at low weights and heights being kids, medium weights and heights being teens, higher weights and heights being middle-aged, and higher weights and heights and less number of hairs being elderly.

1d) Yes, we could determine the distance between two DNA strands by measuring the minimum cost sequence of edit operations needed to change one DNA into another. Each base could be akin to a letter in the *string to string correction problem.* A few modifications are needed: i) substitution cost should be based on which bases are more likely to mutate to each-other (lower substitution cost) and which bases that are less likely to mutate to each-other (higher substitution cost), ii) deletion and insertion costs should be infinity, since bases don't delete themselves or self-create into DNA (I would need to check with my wife on that medical/biology assertion).

2a) Developed Find-Closest-Word and Levenshtein Distance functions as described with InsertionCost = SubstitutionCost = DeletionCost = 1

2b) Documented assumption: The ToBeSpellCheckedFileName contains only words that are meant to be spell-checked; ie. it does not include the trueword, like in wikipediatypo.txt. This assumption is based on my reading of the homework request and Professor Pardo's response on

Piazza to post ”for hw2.2 B”.

3a) Takes about an hour (61 mins) for 635 test words over a dictionary of 21,877 words; this was on a wikipediatypoclean.txt. A file approximately 6x larger would take about 6 hours. This test was not run, but instead extrapolated based on a smaller data set. Our algorithm is brute-force O(n*m) where n = size of test set and m = size of dictionary.

Here are smaller samples below:

```
<run-measure-error-over-dataSet(n = 2, m = 21,877)>
closest to ”abandonned”: ”abandoned” (1.0 LD score)
closest to ”aberation”: ”aberration” (1.0 LD score)
*error-rate: 0.0
*Time it took run function: 15.4750211239

<run-measure-error-over-dataSet(n = 20, m = 21,877)>
closest to ”abandonned”: ”abandoned” (1.0 LD score)
closest to ”aberation”: ”aberration” (1.0 LD score)
closest to ”abilty”: ”ability” (1.0 LD score)
closest to ”abondon”: ”abandon” (1.0 LD score)
closest to ”abbout”: ”about” (1.0 LD score)
closest to ”abotu”: ”abate” (2.0 LD score) - ERROR resulting in 1/20 or .05 error rate
closest to ”abscence”: ”absence” (1.0 LD score)
...
closest to ”acadamy”: ”academy” (1.0 LD score)
*error-rate: 0.05
*Time it took run function: 116.391160965
```

3a continuted) Based on an hour for one run of n = 635 test words, testing $4^3 = 64$ parameter combinations would take 2 days, 17 hours. 10-fold cross validation - determining the parameter based on a subset training run, then applying the best parameters to a test set - would increase this by a factor of 10. That is infeasible, and judgement is necessary.

3b) The experiment will use a smaller sample set (wikipediatypoclean-small.txt, attached) with multiple errors with all costs=1. The experiment will use gradient descent to determine a locally optimal solution of parameter combinations. Randomly choose a parameter in the range of 0,1,2,4 for insertion, deletion, and substitution costs. Run a test run to get an error rate. Iterate each of the parameters individually (± 1); if any parameter combination resulted in an improved error rate, move there. Repeat until the parameter combination yields an error rate that doesn't improve via any iteration of the parameters, or time exceeds one hour. If time remains when the algorithm returns, re-run the experiment choosing a new randomly-located parameter starting set, and see if the resulting parameters improves the error-rate.

3c) See attached excel file .pdf.

4a) Developed qwerty-levenshtein-distance function as described in the assignment. Here are a few example outputs:

```
qwerty distance between keys 1 and ! is: inf; qwerty distance between keys ! and 1 is: inf
qwerty distance between keys Z and z is: 0; qwerty distance between keys z and Z is 0
qwerty distance between keys Z and t is: 6; qwerty distance between keys t and Z is: 6
```

4b) See attached excel file .pdf.