Assignment 2. Distinguish Natural and Random Peptides

Due date (Feb. 18, 2019)

Write a program to tell whether a peptide sequence is natural or not. We say a peptide is "natural" if it is a substring of a protein produced by a real biology.

Command Line:

We will run your program with "program inputFile". Your program will output to the standard output.

Input:

A single text file with a list of peptides in it, each line contains one peptide's sequence. The peptide's sequence contains only the 20 common amino acids. Each peptide takes a single line. The length of each peptide is uniformly randomly distributed between 20 and 40 (inclusive). For example:

```
LLLSLYYPNDRKLLDYKE
VSRVSSDADPAGGWCRKWYSAHRGPDQDAALG
```

Output:

Each line of the output corresponds to a peptide in the input file, in the same order. Each line contains three columns: score1, score2, and the original peptide sequence. The three columns are separated with one or more whitespace characters.

The scores should indicate the confidence your program think the peptide is natural. A higher score indicates a higher confidence. The method calculating score1 is provided to you later in this document. You are responsible to develop the method to calculate score2. For example, the following illustrates the format of the output for the above input (the actual numbers given here are only for illustration purpose)

```
1.73  2.3 LLLSLYYPNDRKLLDYKE
0.59 -1.2 VSRVSSDADPAGGWCRKWYSAHRGPDQDAALG
```

Test Cases:

Each test file will contain a mixture of equal number of natural and random peptide sequences. The natural peptides will be sampled from a real protein sequence database. We will use the uniprot/swissprot protein sequence database to sample natural peptides. The resource is downloadable at http://www.uniprot.org/downloads. The "Reviewed (Swiss-Prot)" database is the one. The fasta file format is recommended for your development purpose (see screenshot).

Note that the annotation line (starting with a '>' sign) of each protein in the fasta file is irrelevant for our purpose and should be discarded. Also, there may be letters in the sequences that do not code one of the 20 common amino acids. Our test cases will not include those peptides.

The random peptides will be obtained by randomly shuffling the amino acids within the natural peptides. Notice that the random and natural peptides in a test file may be sampled independently to each other.

Performance Evaluation:

The prediction accuracy of a score function is calculated as follows. Suppose a test file contains N natural and N random peptides. First, your results are sorted according to the descending order of the score. Then the first N results with the highest scores are taken.

$$accuracy = \frac{number\ of\ natural\ peptides\ in\ top\ N}{N}$$

Marking will be based on this accuracy and then normalized to fit a nice grade distribution for the whole class.

Scoring Functions:

- The first score (score1) should be the following k-mer frequency score. For each k-mer, let p be its frequency in the real peptides and q be its frequency in random peptides, respectively. Then assign a score $\log_2 \frac{p}{q}$ to it and record it in a parameter file or your program. A peptide's score is the sum of its k-mer scores. Use k=3. You need to estimate these frequencies by yourself and find a way to embed the information in your program.

- The second score (score2) is your own. If you want to use a machine learning package, you must check with the TA to ensure the particular package is available under the marking environment;

What to Submit:

(1) The source code.

(2) A pdf file contains (a) a brief description of your scoring function, (b) a description of a testing data you used to test your program, and (c) the accuracies of the two score functions according to your test.

Additional Information:

1. You may need to use additional parameter files (to store the k-mer frequency, for example) for your program. But your total submission size should not exceed 2M bytes. Being larger than 2M bytes may lead to penalty or rejection of the submission.
2. To save size, you are allowed to zip your parameter file and expand it in memory in your program while loading.
3. Yes, you are allowed to copy score1 to score2 - if you choose not to develop a better score function. You are also allowed to use score1 as a feature in score2.