SIL765 - Assignment 1

Name : ***Jitesh Shamdasani***

Entry Number : ***2024JCS2043***

Date of Submission : ***12/01/2025***

Overview

The `DecipherText` class implements a hill-climbing algorithm to decrypt ciphertext based on trigram frequencies. This algorithm aims to maximize the similarity of decrypted text to English text by optimizing a key through iterations. The fitness of the deciphered text is determined using trigram statistics derived from English text.

The algorithm intelligently swaps characters in the decryption key to improve the fitness score iteratively. If the fitness score reaches a local maximum without finding the true solution, the process restarts multiple times to find better keys and plaintext.

Procedure

Input

1. **Ciphertext**: The encoded text that needs to be deciphered.
2. **TRIGRAMS Dictionary**: A dictionary containing trigram frequencies used to evaluate the fitness of deciphered text.

Steps

1. **Initialization**:
    - Trigram probabilities are converted to logarithmic values for efficiency.
    - A random key (`current_key`) and the initial fitness score are generated.
2. **Hill-Climbing Algorithm**:
    - Randomly swap two characters in the key (`randomize` function).
    - Calculate the fitness score of the text deciphered with the new key.
    - If the fitness score improves, the new key is retained; otherwise, it might be retained with a probability influenced by a `random_factor` (to escape local maxima).
    - Repeat this process until 10,000 iterations without improvement occur.
3. **Decipher Final Text**:
    - Decrypt the ciphertext with the best key found.
    - Maintain special characters and spaces from the original ciphertext.
    - Map the deciphered characters to the original ciphertext to create the final key.
4. **Output**:

- The deciphered plaintext.
- The deciphered key.

Scoring

- Fitness is computed using trigram probabilities (`compute_scores`).
- Higher scores indicate text closer to standard English.

Randomization and Decay

- A `random_factor` is used to occasionally accept worse keys to escape local maxima.
- This factor decays over time to prioritize better keys as the iterations progress.

Function Explanations

`score_sequence(seq, score_dict, block_size, default_value)`

Purpose:

Calculates the fitness score of a sequence (text) based on trigram probabilities.

Explanation:

- **Input**:
  - `seq`: The text sequence to evaluate.
  - `score_dict`: The dictionary containing trigram probabilities.
  - `block_size`: The size of the trigrams (fixed at 3 for trigrams).
  - `default_value`: The score to assign for unknown trigrams.
- **Process**:
  1. Splits the sequence into overlapping blocks of `block_size` (trigrams).
  2. For each block, fetches its score from `score_dict`. If the trigram is not found, uses `default_value`.
  3. Accumulates the scores of all blocks to produce the total score for the sequence.
- **Output**:
  - The total fitness score of the sequence.

`compute_scores(new_key, letter_set, word_list, scores, size, def_val)`

Purpose:

Computes the fitness score of the text produced by applying a candidate decryption key to the ciphertext.

Explanation:

- **Input**:
  - `new_key`: A trial decryption key.
  - `letter_set`: Unique characters from the ciphertext.

    - `word_list`: List of words in the ciphertext (without anchors).
    - `scores`: The trigram score dictionary.
    - `size`: Size of trigrams.
    - `def_val`: Default score for unknown trigrams.
- **Process**:
    1. Maps characters in the ciphertext to their decrypted equivalents using `new_key`.
    2. Transforms each word in `word_list` using the mapping.
    3. Calculates the fitness score of the transformed words using `score_sequence`.
    4. Returns the cumulative score for the entire text.
- **Output**:
    - The total fitness score for the text using the `new_key`.

`decryptt(char_list, key_try, word_list)`

Purpose:

Generates decrypted text by applying a candidate decryption key to the ciphertext.

Explanation:

- **Input**:
    - `char_list`: Unique characters from the ciphertext.
    - `key_try`: A trial decryption key.
    - `word_list`: List of words from the ciphertext (without anchors).
- **Process**:
    1. Maps characters in the ciphertext to their decrypted equivalents using `key_try`.
    2. Iterates over `word_list` and replaces each character based on the mapping.
    3. Combines the transformed words into a single decrypted string.
- **Output**:
    - The decrypted text.

`randomize(key_str)`

Purpose:

Randomly swaps two characters in a key string to create a new candidate key.

Explanation:

- **Input**:
    - `key_str`: The current decryption key.
- **Process**:
    1. Converts the key into a list of characters.
    2. Selects two random indices in the key.
    3. Swaps the characters at the selected indices.
    4. Converts the modified list back into a string.

- **Output**:
    - A new randomized key.

Outputs

1. **Deciphered Plaintext**: The human-readable text derived from the ciphertext.
2. **Deciphered Key**: The key mapping used to decrypt the ciphertext.

Example:

Cipher Text-1

```
Ciphertext: 1981y, $pp1n1yuux oq@ 2@3s5u1n $p 1981y, 1v y n$s9o2x 19 v$soq yv1y.
1o 1v oq@
 v@6@9oq uy27@vo n$s9o2x 5x y2@y, oq@ v@n$98 0$vo 3$3su$sv n$s9o2x, y98 oq@ 0$vo
 3$3su$sv 8@0$n2ynx 19 oq@ #$2u8. 5$s98@8 5x oq@ 1981y9 $n@y9 $9 oq@ v$soq, oq@
 y2y51y9 v@y $9 oq@ v$soq#@vo, y98 oq@ 5yx $p 5@97yu $9 oq@ v$soq@yvo, 1o vqy2@v
 uy98 5$28@2v #1oq 3yw1voy9 o$ oq@ #@vo; nq19y, 9@3yu, y98 5qsoy9 o$ oq@ 9$2oq;
y98
 5y97uy8@vq y98 0xy90y2 o$ oq@ @yvo. 19 oq@ 1981y9 $n@y9, 1981y 1v 19 oq@ 61n191ox
$p
 v21 uy9wy y98 oq@ 0yu816@v; 1ov y98y0y9 y98 91n$5y2 1vuy98v vqy2@ y 0y21o10@
5$28@2
 #1oq oqy1uy98, 0xy90y2 y98 198$9@v1y. 7$$8, 9$# os29 p$2 oq@ v@n$98 3y2o $p oq@
 4s@vo1$9, 7$$8 usnw!
```

Plain Text-1

```
Deciphered Plaintext:  india, officially the republic of india, is a country in
south asia. it is the seventh largest
 country by area, the second most populous country, and the most populous
democracy in the world.
 bounded by the indian ocean on the south, the arabian sea on the southwest, and
the bay of bengal
 on the southeast, it shares land borders with pakistan to the west; china, nepal,
and bhutan to the
 north; and bangladesh and myanmar to the east. in the indian ocean, india is in
the vicinity of sri
 lanka and the maldives; its andaman and nicobar islands share a maritime border
with thailand,
 myanmar and indonesia. good, now turn for the second part of the question, good
luck
```

Deciphered Key-1

```
Deciphered Key:  y5n8@p7q1xwu09$342vos6#xxx
```

Cipher Text-2

```
Ciphertext: 64s48u46 8y6 q480ryp nrv 6ryy43 2yu$2tn46, n4 54yu u$ o46. un8u yrpnu
n4 6r6 y$u
 vq441 54qq, n80ryp s4043rvn 6348wv, n80ryp y$ 34vu. n4 58v 2yv234 5n4un43 n4 58v
8vq441 $3
 6348wryp. t$yvtr$2v, 2yt$yvtr$2v, 8qq 58v 8 oq23. n4 34w4wo4346 t3#ryp, 5rvnryp,
n$1ryp,
 o4ppryp, 404y q82pnryp. n4 sq$8u46 un3$2pn un4 2yr043v4, v44ryp vu83v, 1q8y4uv,
v44ryp
 483un, 8qq o2u nrwv4qs. 5n4y n4 q$$z46 6$5y, u3#ryp u$ v44 nrv o$6#, un434 58v
y$unryp. ru
 58v x2vu un8u n4 58v un434, o2u n4 t$2q6 y$u s44q 8y#unryp s$3 x2vu nrv 134v4yt4
```

Plain Text-2

```
Deciphered Plaintext:   defeated and leaving his dinner untouched, he went to bed.
that night he did not sleep well,
 having feverish dreams, having no rest. he was unsure whether he was asleep or
dreaming.
 conscious, unconscious, all was a blur. he remembered crying, wishing, hoping,
begging, even
 laughing. he floated through the universe, seeing stars, planets, seeing earth,
all but himself. when
 he looked down, trying to see his body, there was nothing. it was just that he
was there, but he could
 not feel anything for just his presence
```

Deciphered Key-2

```
Deciphered Key:   8ot64spnrxzqwy$1x3vu205x#x
```