ESE303 Final Project

Group Members: Hagar Elhanbly, William Harkless, M. Abdullah Khalid

```
#imports
import math
import random
import numpy as np
import string
from collections import defaultdict
import math
```

```
#read the lines of the standard text War and Peace
example1 = "warandpeace.txt"
file1 = open(example1, "r")
lines = file1.readlines()
```

```
#Transition Probability Matrix
#(counts frequencies of each alphabet pair in the standard text)
freqs = defaultdict(int)
#iterate through all the alphabet pairs and add one to avoid
#future multiplication by zero
for i in range(26):
  for j in range(26):
    freqs[chr(i+97)+chr(j+97)] = 1

for i in range(26):
  freqs[chr(32)+chr(i+97)] = 1

for i in range(26):
  freqs[chr(i+97)+chr(32)] = 1
```

```
# Compute the frequencies of all alphabet pairs in the standard text
for line in lines:
  for i in range(len(line) - 1):
    if not ((line[i].isalpha() or line[i] == ' ') and
            (line[i+1].isalpha() or line[i+1] == ' ')):
      continue
    pair = line[i].lower() + line[i+1].lower()
    freqs[pair] += 1
```

(a) The Score function

```
def score(cypher, freqs, encoded_message):
  score = 1
  #Find indices
  for i in range(len(encoded_message) - 1):
    if encoded_message[i] == ' ':
      char1 = 26
    else:
      char1 = string.ascii_lowercase.index(encoded_message[i].lower())

    if encoded_message[i+1] == ' ':
      char2 = 26
    else:
      char2 = string.ascii_lowercase.index(encoded_message[i+1].lower())
    # Concatenate chars
    pair = cypher[char1].lower() + cypher[char2].lower()
    score *= freqs[pair]

  score = math.log(score + 1)

  return score
```

(b) Markov Chain Simulation

```
#helper method to compute f* from f
def swap_random(seq):
     seq2 = seq.copy()
     idx = range(len(seq2))
     i1, i2 = random.sample(idx, 2)
     seq2[i1], seq2[i2] = seq2[i2], seq2[i1]
     return seq2
```

```
"""
function main:
    Score initial cypher
    Randomly swap two characters in the cypher mapping call this new cypher f star
    Score f star
    if (score(f star) >= score (f)) : f = f star
    else : f = f star with probability e^score(f star) / e^score(f)

"""
def main(cypher, freqs, encoded_message):
  for i in range(10000):
    score_f = score(cypher, freqs, encoded_message)
    cypher_star = swap_random(cypher)
    score_f_star = score(cypher_star, freqs, encoded_message)
    print("Score_f:" + str(score_f))
    print("Score_f_star:" + str(score_f_star))
    if (score_f_star >= score_f):
```

```
      cypher = cypher_star
    else:
      if random.random() < np.exp(score_f_star) / np.exp(score_f):
        cypher = cypher_star

  return cypher
```

```
#read in encoded_message from txt file
encoded_file = "encoded_message.txt"
encoded = open(encoded_file, "r")
encoded_lines = encoded.readlines()

encoded_message = ""
for line in encoded_lines:
  encoded_message += line
```

```
#random cypher as initial guess
np.random.RandomState(seed = 1)
alphabet_string = string.ascii_lowercase
cypher_initial = list(alphabet_string)
cypher_initial.append(' ')
np.random.shuffle(cypher_initial)
```

```
#running the Markov chain
final_cypher = main(cypher_initial, freqs, encoded_message)
```

```
print(final_cypher)
```

```
['y', 'r', 'o', 'z', 'h', 'v', 'u', 'k', 'p', 'c', 'b', 'n', 'w', ' ', 'd', 'q',
```

```
def decode(encoded_message, cypher_guess):
  decoded_message = ""
  for i in range(len(encoded_message)):
    if encoded_message[i] == ' ':
      decoded_message += cypher_guess[26]
    else:
      decoded_message += cypher_guess[string.ascii_lowercase.index(
          encoded_message[i].lower())]
  return decoded_message
```

```
decoded_message = decode(encoded_message, final_cypher)
print(decoded_message)
```

```
i spent my saturday nights in new york because those gleaming dazzling parties o
```

We have successfully decoded the message:

i spent my saturday nights in new york because those gleaming dazzling parties of his were with me so vividly that i could still hear the music and the laughter faint and incessant from his garden and the cars going up and down his drive one night i did hear a material car there and saw its lights stop at his front steps but i didnt investigate probably it was some final guest who had been away at the ends of the earth and didnt know that the party was over on the last night with my trunk packed and my car sold to the grocer i went over and looked at that huge incoherent failure of a house once more on the white steps an obscene word scrawled by some boy with a piece of brick stood out clearly in the moonlight and i erased it drawing my shoe raspingly along the stone then i wandered down to the beach and sprawled out on the sand most of the big shore places were closed now and there were hardly any lights except the shadowy moving glow of a ferryboat across the sound and as the moon rose higher the inessential houses began to melt away until gradually i became aware of the old island here that flowered once for dutch sailors eyesa fresh green breast of the new world its vanished trees the trees that had made way for gatsbys house had once pandered in whispers to the last and greatest of all human dreams for a transitory enchanted moment man must have held his breath in the presence of this continent compelled into an aesthetic contemplation he neither understood nor desired face to face for the last time in history with something commensurate to his capacity for wonder and as i sat there brooding on the old unknown world i thought of gatsbys wonder when he first picked out the green light at the end of daisys dock he had come a long way to this blue lawn and his dream must have seemed so close that he could hardly fail to grasp it he did not know that it was already behind him somewhere back in that vast obscurity beyond the city where the dark fields of the republic rolled on under the night gatsby believed in the green light the orgastic future that year by year recedes before us it eluded us then but thats no matter tomorrow we will run faster stretch out our arms further and one fine morning so we beat on boats against the current borne back ceaselessly into the pas

ESE303 Project c- f

Thursday, December 2, 2021 8:24 PM

c) Since there are 27 characters in the domain of f, there are $\binom{27}{2}$ ways to move out of f.

Given that we move entirely randomly, the probability of transitioning from state f to f' is $\boxed{\frac{1}{\binom{27}{2}}}$.

d) In part (b), we swap states based on the scores of the two mappings. In Markov Chain J, however, we swap entirely randomly. i.e, without any evidence that f' is better than f.

e) Want to show:

$$\pi(f)\, K(f, f') = \pi(f')\, K(f', f)$$

$$\pi(f)\,K(f,f') = \pi(f')\,K(f',f)$$

Case 1: $f = f'$

trivial, as $\pi(f)\,K(f,f) = \text{LHS} = \text{RHS}$. ✓

Case 2: $f \neq f'$, $A(f,f') \geq 1$

Then, LHS: $\pi(f)\,K(f,f') = \pi(f)\,J(f,f')$

RHS: $\pi(f')\,K(f',f) = \pi(f')\,J(f',f)\,A(f',f)$

$(A(f',f) < 1)$

$$= \pi(f')\,J(f',f)\,\frac{\pi(f)\,J(f,f')}{\pi(f')\,J(f',f)}$$

$$= \pi(f)\,J(f,f')$$

$\therefore$ LHS = RHS ✓

Case 3: $f \neq f'$, $A(f,f') < 1$

LHS: $\pi(f)\, K(f, f') = \pi(f) \cdot \dfrac{\pi(f')\, J(f', f)}{\pi(f)\, J(f, f')}\, J(f, f')$

$= \pi(f')\, J(f', f)$

RHS: $\pi(f')\, K(f', f) = \pi(f')\, J(f', f)$

$(A(f', f) \geq 1)$

$\therefore$ LHS = RHS ✓

This completes the proof.

Let Ω be the set all states in the Markov chain.

Summing both sides over all f' in Ω,

$$\sum_{f' \in \Omega} \pi(f)\, K(f, f') = \sum_{f' \in \Omega} \pi(f')\, K(f', f)$$

$$\pi(f)\sum_{f'\in\Omega} K(f,f') = \sum_{f'\in\Omega} \pi(f')K(f',f)$$

$$\therefore \quad \pi(f) = \sum_{f'\in\Omega} \pi(f')\,K(f,f')$$

which has the form of balance equations. Since $\pi(f)$ satisfies the balance equations, therefore $\pi(f)$ is a stationary distribution of the Markov chain K.

f) In part (b), the probability of returning to a state/mapping (after moving to a state with higher score) is very very low. If we only change state from f to f^* if $S(f^*) > S(f)$, then there is no way to come back to state f and thus the Markov chain is not ergodic.
Note however we do allow to move to a lower score state in the implementation to avoid getting stuck in a sub-optimal absorbing state but

implementation to avoid getting stuck in a suboptimal absorbing state, but the probability of this is very low.

In (e) we showed that the Markov chain K has a stationary distribution. Markov Chain K has a finite state space with only one class and is also aperiodic, thus it is ergodic. Therefore there is a higher chance of returning to a worse-off state in Markov chain K than in (b).

Therefore, we conclude that the algorithm in (b) is a good approach.