Cryptography Homework 1

Samuel Montalbano

April 8, 2025

Instructions

Provide clear answers to the questions below. Include any relevant code snippets and explanations.

Question 1

Question: (A plaintext message consisting of English words was encrypted with a shift cipher to give this cipher text:

RKDDOQHUDQ

Cryptanalyze the text by looking at every possible decryption. You'll see that there's something unusual about this example. What can you conclude about the original plaintext and about the key?

Describe the process you would go through to create a similar question.)

Answer

Using my brute force decipher, I came up with a MIC for all the possibilities as:

```
Decrypted Text: rkddoqhudq, Mutual IC: 0.0363
Decrypted Text: qjccnpgtcp, Mutual IC: 0.0303
Decrypted Text: pibbmofsbo, Mutual IC: 0.0393
Decrypted Text: ohaalneran, Mutual IC: 0.0743
Decrypted Text: ngzzkmdqzm, Mutual IC: 0.0190
Decrypted Text: mfyyjlcpyl, Mutual IC: 0.0235
Decrypted Text: lexxikboxk, Mutual IC: 0.0346
Decrypted Text: kdwwhjanwj, Mutual IC: 0.0334
Decrypted Text: jcvvgizmvi, Mutual IC: 0.0245
Decrypted Text: ibuufhyluh, Mutual IC: 0.0373
Decrypted Text: hattegxktg, Mutual IC: 0.0592
Decrypted Text: gzssdfwjsf, Mutual IC: 0.0322
Decrypted Text: fyrrcevire, Mutual IC: 0.0584
Decrypted Text: exqqbduhqd, Mutual IC: 0.0321
Decrypted Text: dwppactgpc, Mutual IC: 0.0372
Decrypted Text: cvoozbsfob, Mutual IC: 0.0379
Decrypted Text: bunnyarena, Mutual IC: 0.0615
Decrypted Text: atmmxzqdmz, Mutual IC: 0.0292
Decrypted Text: zsllwypcly, Mutual IC: 0.0294
Decrypted Text: yrkkvxobkx, Mutual IC: 0.0206
Decrypted Text: xqjjuwnajw, Mutual IC: 0.0231
Decrypted Text: wpiitvmziv, Mutual IC: 0.0388
Decrypted Text: vohhsulyhu, Mutual IC: 0.0447
Decrypted Text: unggrtkxgt, Mutual IC: 0.0406
Decrypted Text: tmffqsjwfs, Mutual IC: 0.0333
Decrypted Text: sleepriver, Mutual IC: 0.0703
```

If I read these options and look for the most possible decrypted text, I would find that bunnyarena or sleep river are the only ones that look like english. However, my MIC calculation actually rates ohaalneran as the highest possibility. This is because the letters in ohaalneran are more common in the english language than the letters in bunnyarena or sleepriver. I believe this problem arises from the fact that the encrypted text is too short. If I were to create a similar question, I would make a longer encrypted text so that the MIC is more accurate.

Code

```python

def shift_decryt_brute_force(encrypted_text):
encrypted_text = encrypted_text.lower()
possibilities = []
for shift in range(26): #brute force shift each character by 1-26
    decrypted_phrase = ''
```

```
        for char in encrypted_text:
            if char.isalpha():
                decrypted_letter = chr(((ord(char) - ord('a')) - shift) % 26) + ord('a'))#ascii shift
                decrypted_phrase += decrypted_letter
        possibilities.append(decrypted_phrase)
    return possibilities

def calculate_mutual_index_of_coincidence(possibilities):
    english_freq = [
        0.082, 0.015, 0.028, 0.043, 0.127, 0.022, 0.020, 0.061, 0.070, 0.002,
        0.008, 0.040, 0.024, 0.067, 0.075, 0.019, 0.001, 0.060, 0.063, 0.091,
        0.028, 0.010, 0.023, 0.001, 0.020, 0.001
    ]

    def calculate_frequency_distribution(text):
        frequency = [0] * 26
        total_chars = 0
        for char in text:
            if char.isalpha():
                index = ord(char) - ord('a')
                frequency[index] += 1
                total_chars += 1
        if total_chars > 0:
            frequency = [f / total_chars for f in frequency]
        return frequency

    def mutual_index_of_coincidence(freq1, freq2):
        return sum(f1 * f2 for f1, f2 in zip(freq1, freq2))

    mutual_ic = []
    for possibility in possibilities:
        freq_distribution = calculate_frequency_distribution(possibility)
        mic = mutual_index_of_coincidence(freq_distribution, english_freq)
        mutual_ic.append(mic)
        print(f"Text: {possibility}, MIC: {mic:.4f}")
    return mutual_ic
```

```python

def main():
    encrypted_text = "RKDDOQHUDQ"
    possibilities = shift_decryt_brute_force(encrypted_text)
    mutual_ic = calculate_mutual_index_of_coincidence(possibilities)

    # Print all possibilities with their mutual index of coincidence
    for possibility, ic in zip(possibilities, mutual_ic):
        print(f"Decrypted Text: {possibility}, Mutual IC: {ic:.4f}")

    # Find and print the possibility with the highest mutual index of coincidence
    max_ic_index = mutual_ic.index(max(mutual_ic))
    print("\nMost Likely Decryption:")
    print(f"Decrypted Text: {possibilities[max_ic_index]}, Mutual IC: {mutual_ic[max_ic_index]:.4f}")


if __name__ == "__main__":
    main()
```

Question 2

Question: (For the ciphertext given below, the least common letters are __________ and __________ with a frequency of __________. There are __________ letters tied for the most-frequent, with a frequency of __________. One of the most frequent letters is __________. (Your frequencies should be numbers from 0 to 1, please round to three decimal places).

While this problem can of course (painfully) be done by hand, I strongly recommend that you write a program to count the occurrences of letters and compute the frequency. You will need and use this functionality later, so I also recommend that you think about how to organize your work into functions/methods with usefully-defined and well-documented interfaces.

Here's the ciphertext. Apologies that Canvas forced me to enter newlines in the text; you'll have to strip those out, either by hand or within your code.

OIPPLRVBPQJXGWRUETENAXQRPCCCFTVONGZPDYXTGYQNWFLTLKFFCCUYGV
OACDLEZFHDULBVOJPILFUIHRJUEXMFTZEZPMITEBEPRNICFSHNQVCHNPOEETL

HRHYITIRLUGWMCNJJUXHPRDAMLPLWTRUYYAMNKRXFMCAMYTVLFVBTSFBD
CSXHGGIEMYBLXYCNXU)

Answer

S,K - .014 5 .057 P

Question 3

Question: (In a previous question, you did a frequency count of the ciphertext

OIPPLRVBPQJXGWRUETENAXQRPCCCFTVONGZPDYXTGYQNWFLTLKFFCCUYGW
OACDLEZFHDULBVOJPILFUIHRJUEXMFTZEZPMITEBEPRNICFSHNQVCHNPOEETL
HRHYITIRLUGWMCNJJUXHPRDAMLPLWTRUYYAMNKRXFMCAMYTVLFVBTSFBD
CSXHGGIEMYBLXYCNXU

Using a qualitative analysis of the frequency distribution, what conclusion can you make about the encryption system used to encrypt the text? Explain.)

Answer

We can tell that the encryption system is likely a Vigenere because the MIC is .0409 which is far off from natural english at .065

Question 4

Question: (You travel back in time to the Roman Empire during the Gallic wars. The Gauls do not yet understand the technology of the shift cipher, so they agree to pay you numerous gold coins to disrupt communications within Caeser's army. You capture a message from Cicero to Caeser, rolled up and hidden within a hollowed-out javelin. The message contains the following text:

FNJANDWMNABRNPNCQNUNPRXWRBRWPAJENMJWPNAKNFJANCQNNWNVHRBI

Cryptanalyze the message, and replace it with an alternate message to deliver to Caeser, that Caeser will believe to be authentic and will confound his decision-making.

Note: I *strongly* recommend that you solve this problem by writing a program that will *automatically* decrypt text by choosing the key that gives the best mutual index of coincidence in comparing to English-language frequencies. You can then use this code as a component in an upcoming project.)

Answer

(Decryption: Decrypted Text: shift: 9 , phrase: weareundersiegethelegionisingravedangerbewaretheenemyisclosinginonyou, Mutual IC: 0.0687

replaced message: KNURNENRWKADCDBQNFRUUQNUYBJENCQNNVYRANOAXVCQNNWNVHLUXBRWPRW

)

Code

```
    def shift_decryt_brute_force(encrypted_text):
    encrypted_text = encrypted_text.lower()
    possibilities = []
    for shift in range(26): #brute force shift each character by 1-26
        decrypted_phrase = ''
        for char in encrypted_text:
            if char.isalpha():
                decrypted_letter = chr(((ord(char) - ord('a') - shift) % 26) + ord('a'))#ascii shift
                decrypted_phrase += decrypted_letter
        possibilities.append(f"shift: {shift} , phrase: {decrypted_phrase}")
    return possibilities

def calculate_mutual_index_of_coincidence(possibilities):
    english_freq = [
        0.082, 0.015, 0.028, 0.043, 0.127, 0.022, 0.020, 0.061, 0.070, 0.002,
        0.008, 0.040, 0.024, 0.067, 0.075, 0.019, 0.001, 0.060, 0.063, 0.091,
        0.028, 0.010, 0.023, 0.001, 0.020, 0.001
    ]

    def calculate_frequency_distribution(text):
        frequency = [0] * 26
```

```python
        total_chars = 0
        for char in text:
            if char.isalpha():
                index = ord(char) - ord('a')
                frequency[index] += 1
                total_chars += 1
        if total_chars > 0:
            frequency = [f / total_chars for f in frequency]
        return frequency

    def mutual_index_of_coincidence(freq1, freq2):
        return sum(f1 * f2 for f1, f2 in zip(freq1, freq2))

    mutual_ic = []
    for possibility in possibilities:
        freq_distribution = calculate_frequency_distribution(possibility)
        mic = mutual_index_of_coincidence(freq_distribution, english_freq)
        mutual_ic.append(mic)
        print(f"Text: {possibility}, MIC: {mic:.4f}")
    return mutual_ic

def shift_encrypt_brute_force(plaintext, shift):
    encrypted_text = ''
    for char in plaintext:
        if char.isalpha():
            encrypted_letter = chr(((ord(char) - ord('a') + shift) % 26) + ord('a'))#ascii shift
            encrypted_text += encrypted_letter
    return encrypted_text.upper()





def main():
    encrypted_text = "FNJANDWMNABRNPNCQNUNPRXWRBRWPAJENMJWPNAKNFJANCQNNWNVHRBLUXBRWPRWXWHXD"
    possibilities = shift_decryt_brute_force(encrypted_text)
    mutual_ic = calculate_mutual_index_of_coincidence(possibilities)

    # Print all possibilities with their mutual index of coincidence
    for possibility, ic in zip(possibilities, mutual_ic):
        print(f"Decrypted Text: {possibility}, Mutual IC: {ic:.4f}")

    # Find and print the possibility with the highest mutual index of coincidence
    max_ic_index = mutual_ic.index(max(mutual_ic))
    print("\nMost Likely Decryption:")
    print(f"Decrypted Text: {possibilities[max_ic_index]}, Mutual IC: {mutual_ic[max_ic_index]:.4f}")
    print(shift_encrypt_brute_force("believeinbrutushewillhelpsavetheempirefromtheenemyclosingin", 9))

main()
```

Question 5

Question: (By nefarious means you capture the ciphertext message below:

WHPSLPRBYXLOIVJXZBRYBIV

Villanously, you also gain access to the associated encryption machine, found here:

https://www.cs.du.edu/ ftl/vigplaintextattack.html

Decipher the message. Describe the process you went through to figure this out (including any points where you might have gotten stuck, and what realizations helped you figure it out).)

Answer

I want to start by saying process began with my inspecting of the JS file to see if I could just reverse the code, however , it was encrypted as well. Darn! Then I put all a's and got an interesting pattern PHXPHXPHXPHXPHXPHXPHXPHXPHXPHXPHXPHXPHXPHXPHXPHXPHXPHXPHXPHXPHXPHXPHXPHXPH
So using the key PHX for the vigenere cypher I get, hadescubiertomisecreto.

Question 6

Question: (Choose a passage of text, and encrypt it with a (nontrivial) Vigenere cipher. In your answer, give me the plaintext, the value of the key, and the ciphertext. Choose a passage of interest to you, but make sure it is appropriate to share with the class.

These will become part of a future assignment - someone else in the class will do a cryptanalysis of your ciphertext. Please make the text sufficiently long so that the frequency analyses we learn will be effective for decryption. That is, give us at least about 50 characters of text per character of your key.

Post your ciphertext (without the key or the plaintext) to the discussion Vigenere Cipher Messages

)

Answer

plaintext: givemeyourtiredyourpooryourhuddledmassesyearningtobreathefreethewretchedrefuseofyo

key: lie

Ciphertext: rqzpuijwycbmcmhjwycxszzczcvschotiouedaidgilzrtvkewfcmeepiqzipblpevpbgsmhcmjfaiznczcvemixqrralzzidmroblpaiepiswqptidaxputpaxewwebsxmmwqjeucwiqaj

Code

```
    def vigenere_encryption(plaintext, key):
    encrypted_text = ''
    key_length = len(key)
    for i, char in enumerate(plaintext):
        if char.isalpha():
            shift = ord(key[i % key_length].lower()) - ord('a')
            encrypted_letter = chr(((ord(char.lower()) - ord('a') + shift) % 26) + ord('a'))
            encrypted_text += encrypted_letter
        else:
            encrypted_text += char
    return encrypted_text

def vigenere_decryption(ciphertext, key):
    decrypted_text = ''
    key_length = len(key)
    for i, char in enumerate(ciphertext):
        if char.isalpha():
            shift = ord(key[i % key_length].lower()) - ord(’a’)
            decrypted_letter = chr(((ord(char.lower()) - ord(’a’) - shift) % 26) + ord(’a’))
            decrypted_text += decrypted_letter
        else:
            decrypted_text += char
    return decrypted_text

def main():
    ciphertext="RQZPUIJWYCBMCMHJWYCXSZZCZCVSCHOTIOUEDAIDGILZRTVKEWFCMEEPIQZIPBLPEVPBGSMHC"
    key="lie"
    plaintext = vigenere_decryption(ciphertext, key)
    print("plaintext:", plaintext)


main()
```

Question 7

Question: (Each of the following ciphertext messages was encrypted either with the shift cipher or with the vigenere cipher. By calculating the index of coincidence for each, make an educated guess as to which system each was encrypted with. Along with your answer, include the numerical value you found for the index of coincidence. Apologies that Canvas forced me to enter newlines in the text; you'll have to strip those out. The code that you write to do these computations will be re-used in later assignments, so I recommend that you keep your work organized, designing function/method interfaces thoughtfully.

Message 1:

OIPPLRVBPQJXGWRUETENAXQRPCCCFTVONGZPDYXTGYQNWFLTLKFFCCUY GWIUHYOACDLEZFHDULBVOJPILFUIHRJUEXMFTZEZPMITEBEPRNICFSHNQVCHNPO EETLVXDXHRHYITIRLUGWMCNJJUXHPRDAMLPLWTRUYYAMNK CAMYTVLFVBTSFBDITKPXM

CSXHGGIEMYBLXYCNXU

Message 2:
GHPPATMATIIXGLTLDXWMAXFTGBGUETVDPXYTVXXTVAHMAXKTLZHWBGMX
SBDLTBWGHMKBVDLGHPXTIHGLLDBEETZTBGLMLDBEETEHGXRHNFXTGRHNE
PGRHNKKHVDTGWBEEINMWHPGFRLPHKWTGWPXEEMKRMHDBEEXTVAHMAX
OBEBSXWIXHIEXBLMATMBM

Message 1 has an index of coincidence of __________ (please round to three decimal places) and is encrypted with __________ (enter "shift" or "vigenere").

Message 2 has an index of coincidence of __________ (please round to three decimal places) and is encrypted with __________ (enter "shift" or "vigenere").

)

Answer

Message 1 has an index of coincidence of .039 (please round to three decimal places) and is encrypted with vigenere (enter "shift" or "vigenere").

Message 2 has an index of coincidence of .058 (please round to three decimal places) and is encrypted with shift (enter "shift" or "vigenere").

Question 8

Question: (Write a program that outputs a multiplication table (mod n). You can ask the user to input the value of n. Mess with the formatting just enough so the output is legible. Experiment with the program to conjecture(s) about when an integer m has a multiplicative inverse mod n. (Note that two numbers are multiplicative inverses of each other if their product is 1). List any patterns you noticed, or any conjectures you reach at the top of your answer. Then copy/paste your code at the bottom of your answer.)

Code

```
    def multiplication_table_mod(n):
    print(f'\nMultiplication Mod({n})')
    print('     ' + ' '.join(f'{i:3}' for i in range(n)))
    print('     ' + '-' * (4 * n))
    for row in range(n):
        print(f'{row:3} |', end=' ')
```

```python
        for col in range(n):
            print(f'{(row * col) % n:3}', end=' ')
        print()


def main():
    multiplication_table_mod(10)

if __name__ == "__main__":
    main()
```

Question 9

Question: (How many numbers in the set 0, 1, 2, ..., 18 have multiplicative inverses (mod 19)?)

Answer

(Write your explanation or solution here)

Code

```python
# Write your Python code here
def another_example():
    return "Another example"
```

Question 10

Question: (How many numbers in the set 0,1,2,...,51 have multiplicative inverses (mod 52)?)

Answer

(Write your explanation or solution here)

Code

```python
# Write your Python code here
def another_example():
    return "Another example"
```

Question 11

Question: (List the classmates you worked together with on this assignment. Use this as an opportunity to acknowledge them for specific creative ideas or realizations they shared. Also explain in one or two sentences how your work together fit within the parameters for collaboration. Finally, how long did you spend on this assignment?)

Answer

(Write your explanation or solution here)

Code

```python
# Write your Python code here
def another_example():
    return "Another example"
```