DECRYPTING CIPHERED TEXT USING MARKOV CHAIN MONTE CARLO METHODS

IME625A: STOCHASTIC PROCESSES AND THEIR APPLICATIONS

PROJECT REPORT

GROUP 6

Aastha Ramteke (180009)

Kaushiki Agarwal (180350)

Neha Aggarwal (180466)

May 03, 2021

Abstract

Cipher Breaking is a rigorous task which is usually done manually. In this paper, we will propose a code based on Markov Chain Monte Carlo method to solve ***SUBSTITUTION CIPHER****. We exploited the redundancy and patterns in English language to decrease our sample space significantly. We used a Search Method to search for a candidate decryption key and a Scoring Function to come up with the best decryption key amongst the possible candidate. This method can further be extended to other semantics of English language to improve the efficiency.*

Keywords: *Encryption, Decryption, Substitution Cipher, MCMC model, Markov Chain, Monte Carlo, N-gram model*

INTRODUCTION

Substitution cipher is a cipher where a plain text is converted to cipher text using characters from the same set. There are various ways through which substitution cipher can be solved: One simple way could be to just select a key and then the key can be used to decrypt our code, but since each letter can be replaced by 26 letters, it would lead to many combinations thus increasing the space complexity. So, to reduce the search space and increase the efficiency many other methods were published.

Hidden Markov Chain was earlier used to solve Substitution cipher. In HMM we assume a state Y which is dependable on state X which follows Markov Model, but the problem with this approach was that we need enough cipher text. To solve the above problem, we investigated better models and found Markov Chain Monte Carlo model where series of cipher are produced by Monte Carlo where the cipher to be produced depends only the latest information available to us.

THEORETICAL BACKGROUND

Monte Carlo Methods

Monte Carlo Method [1] is a mathematical technique, used to estimate the possible outcomes of an uncertain event. It is also known as Monte Carlo Simulation or Multiple probability simulation. It is different from normal predictive methods in a way that it predicts a set of outcomes which is based on an estimated range of values versus a set of fixed input values. So, in general a Monte Carlo simulation generates a model of possible outcomes by leveraging a probability distribution for a variable which possesses inherent uncertainty. It calculates the results again and again iteratively using different sets of random inputs every time. When a Monte Carlo Simulation is complete, it gives a range of possible outcomes with the probability of each result occurring. In a typical Monte Carlo simulation, the process is repeated thousands of times to generate better results.

Markov Chain

A Markov Chain is a stochastic process which possesses the property that the probability of transitioning to any particular state is dependent solely on the current state and time elapsed. That is, the probability of future actions is only dependent on the current state irrespective of the path through which the process has arrived to the current state.

In the language of conditional probability and random variables, a Markov chain is a sequence X_0 , X_1, X_2 ,… of random variables satisfying the rule of conditional independence:

$P(X_{n+1} = j|X_n = i, X_{n-1} = i_1, X_1 = i_2, \dots , X_0 = i_n) = P(X_{n+1} = j|X_n = i)$

for all $j, i, i_1, i_2, \dots , i_n \in \Omega$

Markov Chain Monte Carlo Method

Combining the two methods, Markov Chain and Monte Carlo, a Markov Chain Monte Carlo Method is a Monte Carlo sampling in which the next sample is dependent on the existing sample, by Markov property. Unlike the normal Monte Carlo sampling methods that are able to draw independent samples from the distribution, in MCMC the choice of the next sample depends on the current sample.

MCMC algorithms [2] are sensitive to their starting point, and often require a warm-up phase or burn-in phase to move in towards a fruitful part of the search space, after which prior samples can be discarded and useful samples can be collected.

Substitution Cipher

A substitution cipher [3] is a method of encryption where the units of plaintext are replaced with some ciphertext in a specified way with the help of a key. The “units” which are replaced might be single letters or a combination of letters. By performing the inverse of the process we can decipher the text.

Substitution cipher can be of many types. The simplest way a substitution cipher can be performed is by substitution of a single alphabet by some other alphabet, which is unique throughout the cipher text.

Example of Substitution cipher

Alphabet	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Encryption key	KDGFNSLVBWAHEXJMQCPZRTYIUO

Plain text - where are you going

Cipher text - yvncn kcn ujr ljbxl

INPUT

```
states,best_state,history = MCMC_decrypt(10000,cipher_text,scoring_params,plain_text)
```

In the main function we call the MCMC decrypt where we provide

- 10000 - Number of iterations.
- cipher_text - The encrypted form of the text.
- scoring_params - These are the Scoring parameters which are pairs of words with their frequency in the reference text that is provided, which in this case is the book "War and Peace".
- plain_text - It is the decrypted text that is given to us.
- best_state – The most optimum decryption key obtained
- states – All proposed decryption keys
- history - Ratio of how much our decrypted text is equal to the original text

METHODOLOGY

We have designed an experiment based stochastic system to find the key for a ciphered text generated using a substitution cipher. Our solver system consists of two major components - one is a stochastic search method for traversing the key space and the other one is a scoring function for calculating the correctness of the available keys.

N-gram Model

N-gram model [4] refers to (n-1) th order Markov chain of language. It is a continuous sequence of n-items taken from a sample text. The items can be characters, words, phrases, etc. When n=2 it is known as bigram model. Here we are using bi-gram model of characters.

Search Method

It is a stochastic search method over the entire key space. Key space includes 26! permutations of English alphabets which are potential keys for an encrypted text. The search method is based on first generating a random key and then swapping its letters for the generation of potential keys. We calculate the score associated with each key using the scoring function so that we can get the best match possible. Following are the steps followed in the search method-

1. We start by picking up a random current key using "string.ascii_uppercase" function and maintain the best key variable to store the potential best key with maximum score in each iteration.
2. Given the number of iterations as the input hyper-parameter of the function, we generate a new key by swapping two random letters in the current alphabets of the key in each iteration.

3. Using the scoring function we calculate the score of the current key and proposed key and store it in score_current_cipher and score_proposed_cipher respectively.
4. When the score of the proposed state is greater than the current state, we update our best key to the proposed key.
5. Else we flip a coin with probability of head = score_proposed_cipher/ score_current_cipher. We take a uniform random variable between 0 and 1 and generate its value randomly, if it comes to be greater than probability of head, we update our current key to the proposed key.
6. For each iteration, we repeat the same process from the 2nd step until we get a maxima in the score.

Scoring Function

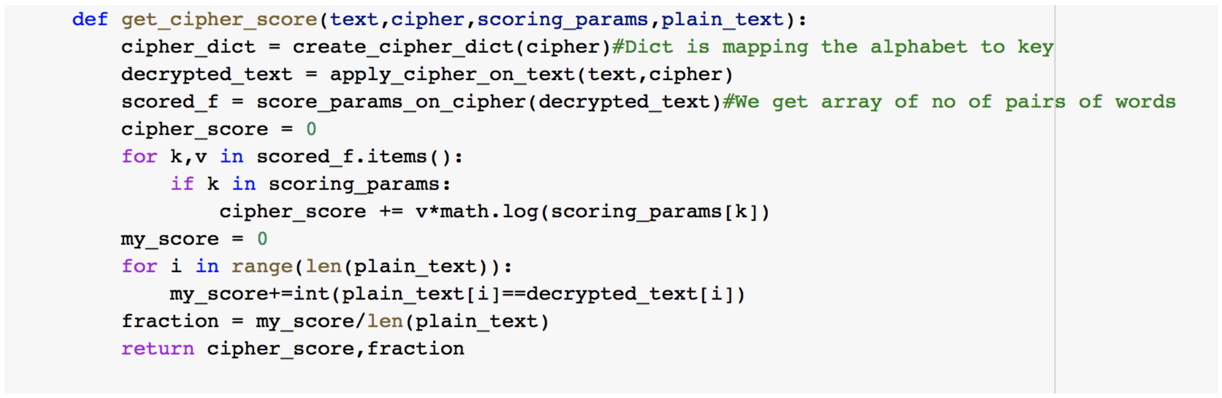
It is a way to "score" our newly generated keys according to their closeness to the expected correct key. In the function, we will try to find out how many times one alphabet of generated decrypted text comes after another in a long reference English text (here we are using "War and Peace" for this purpose). For each pair of characters β_1 and β_2, we record the number of times that specific pair appears consecutively in the reference text in the variable $R(\beta_1, \beta_2)$.
In the same way, for a decryption key x, we let $F_x(\beta_1, \beta_2)$ to store the number of instances that pair appears when the ciphertext is decrypted using the decryption key x.
We then define the score variable for a particular decryption key x using:

$$Score(x) = \prod R(\beta_1, \beta_2)^{F_x(\beta_1, \beta_2)}$$

For every consecutive pair of letters in the decrypted text, the scoring function can be thought of as *log(Score (x))*, where *Score (x)* is as defined above.
As the pair frequencies in the decrypted text nearly equals that of reference text, the score function increases, thus increasing the chances that the decrypted key is the correct one.

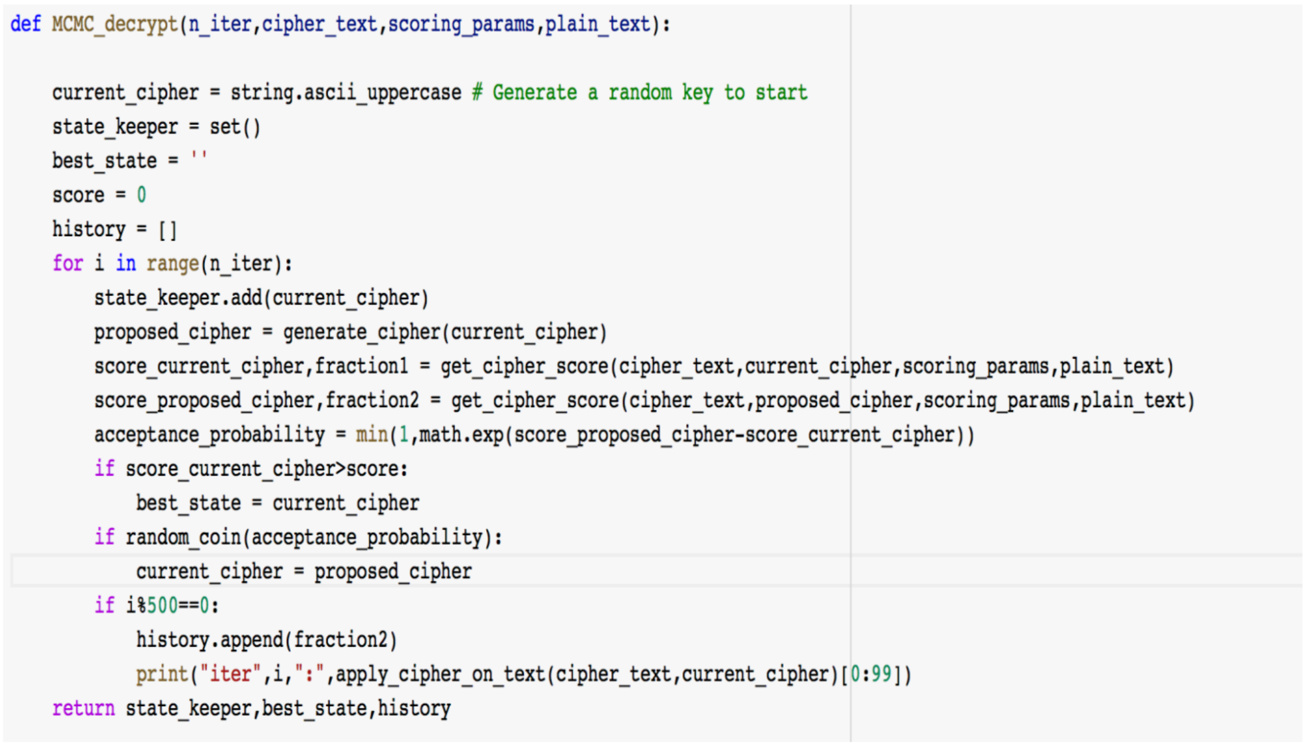
CODE SNIPPETS

Scoring function



```
def get_cipher_score(text,cipher,scoring_params,plain_text):
    cipher_dict = create_cipher_dict(cipher)#Dict is mapping the alphabet to key
    decrypted_text = apply_cipher_on_text(text,cipher)
    scored_f = score_params_on_cipher(decrypted_text)#We get array of no of pairs of words
    cipher_score = 0
    for k,v in scored_f.items():
        if k in scoring_params:
            cipher_score += v*math.log(scoring_params[k])
    my_score = 0
    for i in range(len(plain_text)):
        my_score+=int(plain_text[i]==decrypted_text[i])
    fraction = my_score/len(plain_text)
    return cipher_score,fraction
```

MCMC Decrypt



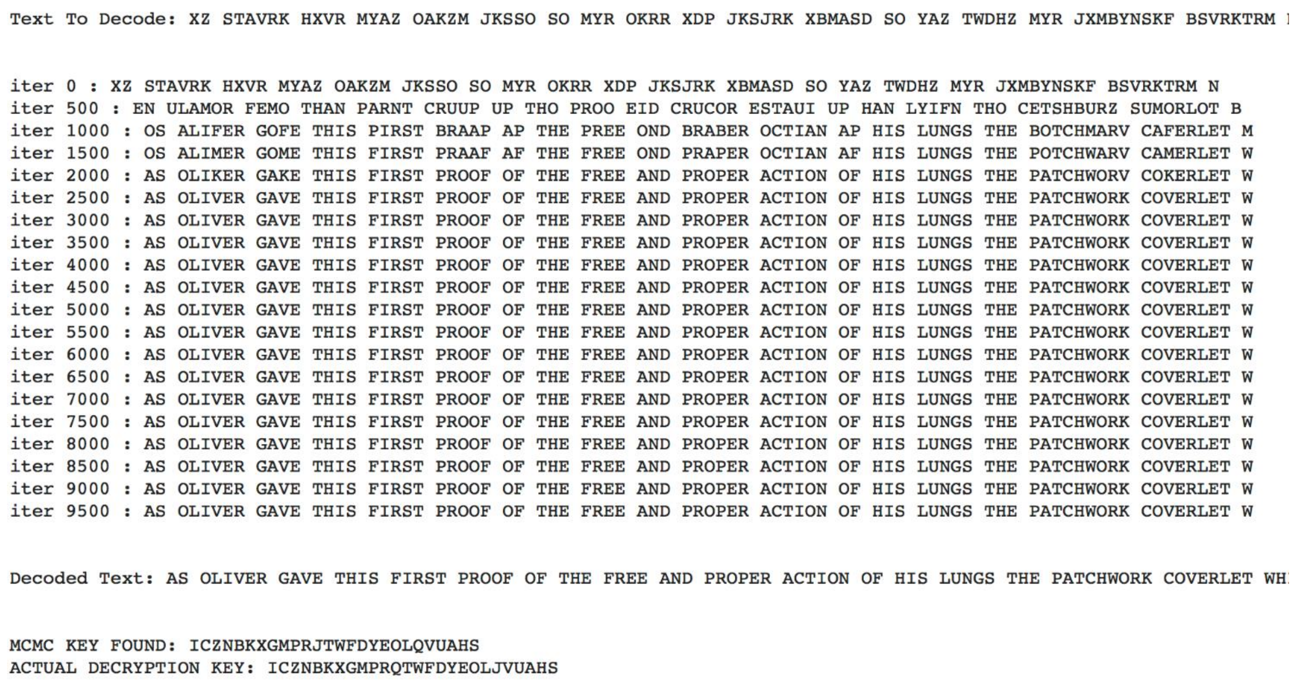
```python
def MCMC_decrypt(n_iter,cipher_text,scoring_params,plain_text):

    current_cipher = string.ascii_uppercase # Generate a random key to start
    state_keeper = set()
    best_state = ''
    score = 0
    history = []
    for i in range(n_iter):
        state_keeper.add(current_cipher)
        proposed_cipher = generate_cipher(current_cipher)
        score_current_cipher,fraction1 = get_cipher_score(cipher_text,current_cipher,scoring_params,plain_text)
        score_proposed_cipher,fraction2 = get_cipher_score(cipher_text,proposed_cipher,scoring_params,plain_text)
        acceptance_probability = min(1,math.exp(score_proposed_cipher-score_current_cipher))
        if score_current_cipher>score:
            best_state = current_cipher
        if random_coin(acceptance_probability):
            current_cipher = proposed_cipher
        if i%500==0:
            history.append(fraction2)
            print("iter",i,":",apply_cipher_on_text(cipher_text,current_cipher)[0:99])
    return state_keeper,best_state,history
```

RESULTS

The below image shows the results obtained by decrypting the cipher text –

XZ STAVRK HXVR MYAZ OAKZM JKSSO SO MYR OKRR XDP JKSJRK XBMASD SO YAZ TWDHZ MYR JXMBYNSKF BSVRKTRM NYABY NXZ BXKRTRZZTQ OTWDH SVRK MYR AKSD ERPZMRXP KWZMTRP MYR JXTR OXBR SO X QSWDH NSIXD NXZ KXAZRP ORRETQ OKSI MYR JATTSN XDP X OXADM VSABR AIJRKORBMTQ XKMABWTXMRP MYR NSKPZ TRM IR ZRR MYR BYATP XDP PAR MYR ZWKHRSD YXP ERRD ZAMMADH NAMY YAZ OXBR MWKDRP MSNXKPZ MYR OAKR HAVADH MYR JXTIZ SO YAZ YXDPZ X NXKI XDP X KWE XTMRKDXMRTQ XZ MYR QSWDH NSIXD ZJSFR YR KSZR XDP XPVXDBADH MS MYR ERP Z YRXP ZXAP NAMY ISKR FADPDRZZ MYXD IAHYM YXVR ERRD RGJRBMRP SO YAI



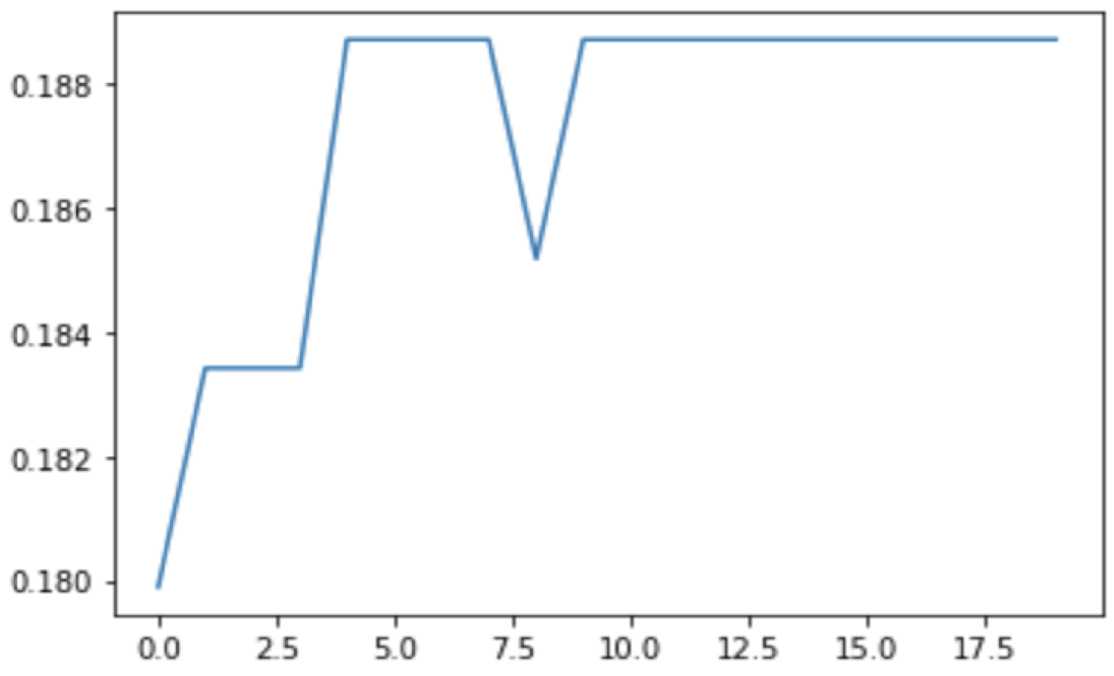
```
Text To Decode: XZ STAVRK HXVR MYAZ OAKZM JKSSO SO MYR OKRR XDP JKSJRK XBMASD SO YAZ TWDHZ MYR JXMBYNSKF BSVRKTRM


iter 0 : XZ STAVRK HXVR MYAZ OAKZM JKSSO SO MYR OKRR XDP JKSJRK XBMASD SO YAZ TWDHZ MYR JXMBYNSKF BSVRKTRM N
iter 500 : EN ULAMOR FEMO THAN PARNT CRUUP UP THO PROO EID CRUCOR ESTAUI UP HAN LYIFN THO CETSHBURZ SUMORLOT B
iter 1000 : OS ALIFER GOFE THIS PIRST BRAAP AP THE PREE OND BRABER OCTIAN AP HIS LUNGS THE BOTCHMARV CAFERLET M
iter 1500 : OS ALIMER GOME THIS FIRST PRAAF AF THE FREE OND PRAPER OCTIAN AF HIS LUNGS THE POTCHWARV CAMERLET W
iter 2000 : AS OLIKER GAKE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORV COKERLET W
iter 2500 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET W
iter 3000 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET W
iter 3500 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET W
iter 4000 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET W
iter 4500 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET W
iter 5000 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET W
iter 5500 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET W
iter 6000 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET W
iter 6500 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET W
iter 7000 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET W
iter 7500 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET W
iter 8000 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET W
iter 8500 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET W
iter 9000 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET W
iter 9500 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET W


Decoded Text: AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET WH


MCMC KEY FOUND: ICZNBKXGMPRJTWFDYEOLQVUAHS
ACTUAL DECRYPTION KEY: ICZNBKXGMPRQTWFDYEOLJVUAHS
```

The below graph is between Ratio of how much our decrypted text is equal to the original text (on y- axis) and number of iterations in thousands (on x- axis).



PREVIOUS WORK

Earlier many works have been done on Substitution ciphers, but none of them were quite effective, let us look into some of them:

- One of the earliest research was done by Carrol and Martin [5]who used hand-coded heuristics (it is technique where we provide the computer the function to solve the problem), but the results weren't that satisfactory.
- One more method that was quite prevalent was to solve the Substitution cipher using Hidden Markov Chain [6], but the problem with this method is that it requires a large dataset, also it is computationally very intensive.
- Many researchers used Unigram and bigram statistics with different algorithm like Genetic algorithm , another used Relaxation scheme [7].
- A method similar to our method used MCMC as their approach and used the Quipster algorithm to decrypt the Substitution Cipher [8].

FURTHER DEVELOPMENTS

- Here we are considering the bi-gram model for characters of a word. This can also be extended to n-gram model of words.
- Also the semantics of language can be combined to improve the output results.
- In order to increase the efficiency further we can even apply techniques to take into account cases where the bi-grams of low frequency are missing from the training corpus.

CONCLUSION

As can be seen from the code snippet of 1 our MCMC key found is ICZNBKXGMPRJTWFDYEOLQVUAHS and the actual key should be ICZNBKXGMPRQTWFDYEOLJVUAHS and we can see that the keys are quite similar.
From the graph we can be see that as the number of iterations increase the fraction also increases but after a particular iteration it becomes a constant and converges to the best key.

REFERENCES

[1] IBM Cloud Education (24th August 2020) https://www.ibm.com/cloud/learn/monte-carlo-simulation

[2] Jason Brownlee (November 6, 2019) https://machinelearningmastery.com/markov-chain-monte-carlo-for-probability/

[3] Wikipedia (14th April 2021) Substitution Cipher

[4] Wikipedia (28th April 2021) https://en.wikipedia.org/wiki/N-gram

[5] J. Carrol and S. Martin. The automated cryptanalysis of substitution ciphers. *Cryptologia*, 10(4):193–209, 1986

[6] Vobbilisetty, Rohit, "Cryptanalysis of Classic Ciphers Using Hidden Markov Models" (2015). Master's Projects. 407. DOI: https://doi.org/10.31979/etd.qxu5-d8pk https://scholarworks.sjsu.edu/etd_projects/407

[7] R. Spillman, M. Janssen, B. Nelson, and M. Kepner. Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers. *Cryptologia*, 17(1):31–44, January 1993

[8] Hasinoff, Sam. (2003). Solving Substitution Ciphers.