Algorithmically Breaking Substitution Ciphers with Markov Chains

Jacob Kawako

August 2024

1 Introduction

The substitution cipher is one of the most basic and intuitive ciphers there is, where each distinct character of the plaintext is swapped for a different character. Most methods of breaking substitution ciphers involve analyzing frequency of individual characters or possible occurences of common letter groupings. However, thanks to Markov chains, there exists an algorithm that can automate the solving of substitution ciphers. This paper will introduce basic facts of Markov chains and language models and apply them, along with the Metropolis-Hastings algorithm to the automated solving of substitution ciphers.

2 Substitution Ciphers

It will help us down the line to define substitution ciphers rigorously.

Definition 1 (Substitution Ciphers)**.** *Let A and A' be "alphabets" (sets of characters) of the same size. Given a plaintext m composed of elements of A and a bijective mapping $f : A \to A'$, Elements of m are swapped with their image under f to form the ciphertext.*

If $A = A'$ and both represent the English language, then $f(m)$ simply permutates the distinct letters.

Techniques of breaking substitution ciphers tend to rely on the simple fact that the plaintext should make sense and resemble a language that exists. These attacks' reliance on linguistic patterns suggest the use of a language model.

2.1 Language Models

Language models are probabilistic representations of language, which, given a string of characters or words can, by some metric, determine the likelihood that the string resembles real language and not formless nonsense.

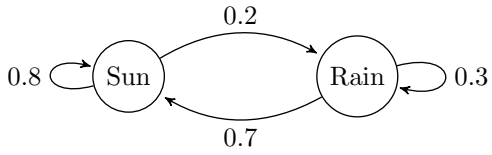

Figure 1: Markov chains are commonly represented using diagrams.

More advanced language models (LLMs, or Large Language Models) are capable of forming their own language with adequate clarity. As it turns out, the language model required to break substitution ciphers is very simple. We will return to language models later.

3 Markov Chains

A Markov chain is a process along a set of possible states $S_1, S_2, \cdots, S_n$ along with probabilities representing movement from one state to another.

We might have our states represent the rain, and our probability chart may be as follows:

	Sun	Rain
Sun	0.8	0.2
Rain	0.7	0.3

According to this chart, if on Monday it is sunny, the chance of it raining on Tuesday is 0.2, and if on Friday it rains, it will be sunny the next day with probability 0.7. The Markov chain in this case would be defined as the progression of the days. See Fig.1 for a visual representation of this Markov chain.

However, this only tells us about the weather tomorrow; we may want to know about the weather next week. From here, it becomes beneficial to represent Markov Chains using matrices.

$$\begin{bmatrix} 0.8 & 0.2 \\ 0.7 & 0.3 \end{bmatrix}$$

We refer to this as the *transition matrix* of our Markov chain. Let p_{ij}^k be the probability that we will be in state j after k iterations given that we are currently in state i. In our example, we let state s denote sun and state r denote rain. p_{ij}^1 is given by the chart for all $i, j \in s, r$, but we may want to compute p_{sr}^2. There are two possiblities; we either have Sun $\to$ Rain $\to$ Rain or Sun $\to$ Sun $\to$ Rain. The first occurs with probability $0.2 * 0.3 = 0.06$ and the second occurs with probability $0.8 * 0.2 = 0.16$. Summing the two gives us a probability of 0.22. More generally, $p_{ij}^2 = \sum_k p_{ik} p_{kj}$. This definition is very important, due to its resemblence to matrix multiplication. By induction, we have the following:

Theorem 1. *Letting A be the transition matrix of a Markov chain and p_{ij}^k be the probability that starting from state i, we will be in state j in k iterations, then p_{ij}^k is equal to entry i, j of A^k.*

Evaluting A^2 for our weather matrix gives us the following. Note that the rows still sum to 1.

$$\begin{bmatrix} 0.78 & 0.22 \\ 0.77 & 0.23 \end{bmatrix}$$

We don't see anything interesting let, so let's try computing A^{20}. This gives us

$$\begin{bmatrix} 0.77778 & 0.22222 \\ 0.77778 & 0.22222 \end{bmatrix}$$

It can be inferred from this that $\lim_{k\to\infty} A^k = \begin{bmatrix} \frac{7}{9} & \frac{2}{9} \\ \frac{7}{9} & \frac{2}{9} \end{bmatrix}$ Moreover, we have the following theorem.

Theorem 2. *Let A be the matrix of a Markov chain such that there exists some k such that the entries of A^k are nonzero. Then, as $k \to \infty$, A^k approaches a matrix B with all rows equal to the vector v. Furthermore, $vA = v$.*

We will not prove the first fact, but we will prove $vA = v$. We will, however provide some intuition for the existence claim. Essentially, what the A^k claim states is that for all p_{ij}^k there exists some k such that $p_{ij}^k > 0$. This means that it should not be possible for us to leave a state and never come back. Additionally, this condition upholds that there will never be states practically inaccessible to us.

Proof. Let $\lim_{k\to\infty} A^k = B$. We have $B = \lim_{k\to\infty} A^k = \lim_{k\to\infty} A^k * A = A * (\lim_{k\to\infty} A^k) = AB$ Since all of the rows of B are equal to v, it follows that $v = Av$. □

Those familiar with linear algebra may note that v is the eigenvector of A with an eigenvalue of 1. This means it is possible to compute v using linear algebra techniques, but this tends to get complex and unreasonable for large A. It is much more common to simply guess v and check that $vA = v$. This may sound unintuitive, but it is the basis behind the Metropolis algorithm.

4 The Metropolis-Hastings Algorithm

Now we tackle the the problem of algorithmically breaking substitution ciphers. Language models are useful here, as they provide some metric of determining whether or not a key is satisfactory. A function helpful for developing a useful language model for substitution ciphers is one that, given a character and a character preceding it, returns the likelihood of such a combination being reasonable. A language model could use all the consecutive letter pairs in a string

to figure out the likelihood of a string being reasonable English.

This discussion of a letter being determined by the letter before it may suggest the use of Markov chains, but this is not the form Markov chains take in the Metropolis algorithm. During our definition of Markov chains, we discussed the significance of "states". For example, the states in our weather example would be the sun and the rain. We can define our states to be possible keys, of which there are $26! \approx 4 * 10^{26}$. From here, all we need to define are our probabilities, which we will use our language model for.

Let $g(a, b)$, where a and b are characters, be the probability that, given a, b follows. One possible way to define this function is by programmatically analyzing a text for data on character usage.[1] Furthermore, if we let C, our ciphertext, be an ordered multiset of characters $c_1, c_2, \cdots, c_n$, we can define a function $l(C) = \prod_{i=1}^{n-1} (g(c_i, c_{i+1}))$ as a way of representing the likelihood of C being a valid string of letters. Finally, define the permutation $f : \{\text{letters}\} \to \{\text{letters}\}$ as our key. The Metropolis Algorithm is as follows.

- Pick f at random.
- Swap the images of two random values of the domain of f, and let this new key be f'.
- Let $p = \frac{l(f'(C))}{l(f(C))}$.
- Change f to f' with probability $\min(1, p)$.
- Repeat from step 2, and after a set amount of iteration, typically in the thousands, halt the algorithm.

At first, step 3 may seem counterintuitive; if f' turns out to be worse than f, why switch? However, it is important to note that from any state S_n, where the states are our possible keys, only $\binom{n}{2}$ new states are attainable, as this represents all the possible pairs of values whose images under f can be swapped. Without step 3, we could get stuck. Every possible change could result in a worse key, and yet the current key may not be the best possible key. This is why introducing a probabilistic element is necessary.

This algorithm was investigated in depth in [CR10], in which it was tested against an encoded sentence from the Dickens novel "Oliver Twist". The decoding took 2200 iterations, and the sentence made little sense until the very end. This highlights something remarkable about this algorithm: the fact that it can decode a message off of very little information. Substitution ciphertexts tend to be easier to decode if they are lengthy, as they provide more information

[1] Mathematicians love using Tolsoy's "War and Peace" for this.

on the relative frequency of characters and bigrams.

The fundamental idea behind the Metropolis algorithm lies in the fact that the transtion matrix A of our Markov chain representing this algorithm has a stationary distribution, and the row vector representing it is such that its i-th element is equal to the likelihood of the key f_i.

It is not obvious that a stationary distribution of A exists. Recall theorem 2. For all entries of A, does there exist a k such that that entry of A^k are nonzero? There does. We can restate this fact as: "There exists some k such that it is possible to move from one state S_i to any other state S_j in k steps, or $p_{ij}^k > 0$ for all i, j and some k. This is true for the Metropolis algorithm. Given a random key f and another key f', it is possible to recieve f' from swapping values among the image of f.

4.1 More on Metropolis-Hastings

The Metropolis algorithm has proven very useful in breaking substitution ciphers, but this is not its only application. More, generally, the Metropolis algorithm involves choosing an intial state x_0, computing a candidate x_t', then letting $x_{t+1} \coloneqq x_t'$ with probability $\min(\frac{g(x_t')}{g(x_t)}, 1)$, where g is some function that determines the "quality" of x. Otherwise, $x_{t+1} \coloneqq x_t$.

Common applications of the Metropolis algorithm include approximating values of integrals and approximating probability distributions.

References

[CR10] Chen and Rosenthal. Decrypting classical cipher text using markov chain monte carlo. Technical report, University of Toronto, 2010.

[Dia09] Diaconis. The markov chain monte carlo revolution. *Bulletin of the American Mathematical Society*, 2009.

[Low] Lowen. Efficient search with markov chains. Blog Post.

[RS18] Rubinstein-Salzedo. *Cryptography.* Springer, 2018.