Hidden Markov Models for Vigenère Cryptanalysis

Mark Stamp[*] **Fabio Di Troia**[†]
Department of Computer Science
San Jose State University
San Jose, California
[*]`mark.stamp@sjsu.edu`
[†]`fabioditroia@msn.com`

Miles Stamp
Los Gatos High School
Los Gatos, California
`milez000782@gmail.com`

Jasper Huang
Lynbrook High School
San Jose, California
`jhuang821@student.fuhsd.org`

Abstract

Previous work has shown that hidden Markov models (HMM) can be effective for the cryptanalysis of simple substitution and homophonic substitution ciphers. Although computationally expensive, an HMM-based attack that employs multiple random restarts can offer a significant improvement over classic cryptanalysis techniques, in the sense of requiring less ciphertext to recover the key. In this paper, we show that HMMs are also applicable to the cryptanalysis of the well-known Vigenère cipher. We compare and contrast our HMM-based approach to recent research that uses Vigenère cryptanalysis to supposedly illustrate the strength of a type of neural network known as a generative adversarial network (GAN). In the context of Vigenère cryptanalysis, we show that an HMM can succeed with much less ciphertext than a GAN, and we argue that the model generated by an HMM is considerably more informative than that produced by a GAN.

1 Introduction

Hidden Markov models (HMMs) are a class of machine learning techniques that have proved useful in a wide variety of applications, ranging from speech analysis (Rabiner, 1989) to malware detection (Wong and Stamp, 2006). In the realm of classic ciphers, HMMs have been shown to perform well in the cryptanalysis of monoalphabetic substitution ciphers (Berg-Kirkpatrick and Klein, 2013; Lee, 2002; Vobbilisetty et al., 2017).

In this paper, we build on the work in (Vobbilisetty et al., 2017) to show that an HMM is a powerful and practical tool for the cryptanalysis of a Vigenère cipher. Furthermore, we show that an HMM trained on Vigenère ciphertext is informative, in the sense that the model enables us to clearly see which features contribute to the success of the technique. We compare our results to recent work in (Gomez et al., 2018), where a neural network is used to break Vigenère ciphertext messages.

The remainder of this paper is organized as follows. In Section 2, we discuss relevant background topics, with the emphasis on hidden Markov models. Experimental results obtain by applying HMMs to Vigenère ciphertexts are given in Section 3. Finally, in Section 4 we give our conclusions and briefly consider future work.

2 Background

2.1 Vigenère Cipher

A simple substitution cipher uses a fixed one-to-one mapping of the alphabet. It is a standard textbook exercise to break a simple substitution using frequency analysis. A homophonic substitution can be viewed as a generalization of a simple substitution, where a fixed many-to-one mapping is used. That is, in a homophonic substitution, more than one ciphertext symbol can map to a single plaintext letter. In contrast, for a polyalphabetic substitution, the "alphabet" (i.e., the mapping between plaintext and ciphertext) changes. As compared to a simple substitution, a homophonic substitution tends to flatten the ciphertext statistics, thereby making frequency analysis more difficult.

A Vigenère cipher is a simple polyalphabetic scheme, where a keyword is specified, and each letter of the keyword represents a shift of the alphabet. For example, suppose that the keyword is `CAT` and we want to encrypt `attackatdawn`. Then we have

keyword:	`CATCATCATCAT`
plaintext:	`attackatdawn`
ciphertext:	`ctmccdctwcwg`

Letter	a	b	c	d	e	f	g	h	i	j	k	l	m
Frequency	.082	.015	.028	.043	.127	.022	.020	.061	.070	.002	.008	.040	.024
Letter	n	o	p	q	r	s	t	u	v	w	x	y	z
Frequency	.067	.075	.019	.001	.060	.063	.091	.028	.010	.023	.001	.020	.001

Table 1: English monograph statistics

That is, each C in the keyword specifies a shift by 3, each A represents a shift by 0, and each T is a shift by 19, and the keyword is repeated as many times as needed. Of course, if the keyword is known, it is trivial to decrypt a Vigenère ciphertext.

2.2 Friedman Test

When attempting to break a Vigenère ciphertext, the first step is to determine the length of the keyword. The Friedman test (Friedman, 1987), which is based on the index of coincidence (IC), is a well-known method for determining the length of the keyword, provided that sufficient ciphertext is available. An alternative method of finding the keyword length is the Kasiski test (Kasiski, 1863); here we focus on the Friedman test. In any case, once the keyword is known, the Vigenère cipher consists of a sequence of shift ciphers, and the shifts can be determined by a variety of means.

The IC measures the "repeat rate," i.e., the probability that two randomly selected letters from a given string are identical. This test relies on the non-uniformity of letter frequencies in the underlying plaintext.

Suppose that we have a string of text of length N with $n_a > 0$ occurrences of A and $n_b > 0$ occurrences of B, and so on. If we randomly select two letters (without replacement) from this string, the probability that the letters match is given by

$$\frac{n_a(n_a-1)}{N(N-1)} + \frac{n_b(n_b-1)}{N(N-1)} + \cdots + \frac{n_z(n_z-1)}{N(N-1)}$$

In general, the repeat rate, or IC, which we denote as κ, is given by

$$\kappa = \frac{1}{N(N-1)} \sum_{i=0}^{c-1} n_i(n_i-1) \qquad (1)$$

where c is the size of the alphabet, n_i is the frequency of the i^{th} symbol, and N is the length of the string. For English text (without spaces, punctuation, or case), we have $c = 26$, and the the expected frequency of each n_i is known from the language monograph statistics. The monograph statistics for standard English appear in Table 1.

Let κ_e denote the IC for English text. If we compute the IC for a large selection of English text, then based on Table 1, we would expect to find

$$\kappa_e = 0.082^2 + 0.015^2 + 0.028^2 + \cdots + 0.020^2 + 0.001^2 \approx 0.0656.$$

On the other hand, if we have random text drawn from the 26 letter English alphabet, we would expect to find that the IC is

$$\kappa_r = (1/26)^2 + (1/26)^2 + \cdots + (1/26)^2 \approx 0.0385.$$

For a simple substitution cipher, we relabel the letters, which has no effect on the IC. That is, when a monoalphabetic substitution is applied to English plaintext, the IC of the ciphertext is the same as that of the plaintext. Friedman noted that for a polyalphabetic substitution, the larger the number of alphabets, the closer the IC is to κ_r. Hence, for a polyalphabetic substitution, we can use the observed IC to estimate the number of alphabets and, in particular, the length of the keyword in a Vigenère cipher.

Let L be the length of the Vigenère keyword, and assume that the ciphertext is of length N. Then we have L Caesar's ciphers. To simplify the notation, we assume that each of these L ciphers has exactly N/L letters. Under this assumption, the probability of selecting two letters from the same Caesar's cipher is given by

$$\frac{N(N/L-1)}{N(N-1)} = \frac{N/L-1}{N-1}.$$

Similarly, the probability of selecting two letters from different alphabets is

$$\frac{N-N/L}{N-1}.$$

In the former case, the letters are derived from the same simple substitution (in fact, Caesar's cipher), so the chance that they match is κ_e, while in the latter case, the letters are from different Caesar's ciphers, so the chance that they match is about κ_r.

Let κ_c be the computed IC for a given Vigenère ciphertext. Then κ_c is the probability of selecting two letters at random that match and, evidently, this probability is given by

$$\kappa_c = \kappa_e \frac{N/L-1}{N-1} + \kappa_r \frac{N-N/L}{N-1}. \qquad (2)$$

Solving equation (2) for L, we obtain

$$L = \frac{N(\kappa_e - \kappa_r)}{N(\kappa_c - \kappa_r) - (\kappa_e - \kappa_r)}.$$

Since N is large relative to κ_e, κ_r, and κ_c, we can approximate the keyword length by

$$L = \frac{\kappa_e - \kappa_r}{\kappa_c - \kappa_r} \qquad (3)$$

For the case of English text, the expected IC is $\kappa_e \approx 0.0656$, while for the random case (and under the assumption that we have 26 symbols), the IC is $\kappa_r \approx 0.0385$. Recall that κ_c is the IC for the ciphertext, which is computed as in (1). Thus, we can approximate the Vigenère keyword length using (3). In practice, when attempting to break a Vigenère ciphertext message, we would need to test various keyword lengths near the value given by (3).

In Section 3, we compare an HMM-based technique to the results obtained using the standard approach to Vigenère cryptanalysis, as discussed in this section. For our test cases, we find that the HMM outperforms the Friedman test, in the sense of giving us a more precise result for the keyword length. In addition, the HMM simultaneously recovers the shifts, so that the entire Vigenère key is determined.

2.3 Hidden Markov Models

True to its name, a hidden Markov model (HMM) includes a Markov process that is "hidden," in the sense that it is not directly observable. Along with this hidden Markov process, an HMM includes a sequence of observations that are probabilistically related to the (hidden) states. An HMM can be viewed as a machine learning technique that relies on a discrete hill climb algorithm for training.

A generic HMM is illustrated in Figure 1, where A is an $N \times N$ matrix that defines the state transitions in the underlying (hidden) Markov process, and the matrix B contains discrete probability distributions that relate each hidden state X_i to the corresponding observation O_i. That is, row i of the B matrix contains a discrete probability distribution that gives the probabilities of the various observation symbols when the hidden Markov process is in state i. As we show below, the component matrices of an HMM can reveal information about the underlying data that is not otherwise readily apparent to a human analyst. This could be considered an advantage of an HMM over other more opaque forms of machine learning, such as neural networks.

The following notation (Stamp, 2004) is commonly used for HMMs:

T	=	length of the observation sequence
N	=	number of states in the model
M	=	number of observation symbols
Q	=	$\{q_0, q_1, \ldots, q_{N-1}\}$
	=	distinct states of the Markov process
V	=	$\{0, 1, \ldots, M-1\}$
	=	set of possible observations
A	=	state transition probabilities
B	=	observation probability matrix
π	=	initial state distribution
O	=	$(O_0, O_1, \ldots, O_{T-1})$
	=	observation sequence.

Note that the observations are associated with the integers $0, 1, \ldots, M-1$, since this simplifies the notation with no loss of generality. Consequently, we have $O_i \in V$ for $i = 0, 1, \ldots, T-1$.

If we are given a sequence of observations of length T, denoted $(O_1, O_2, \ldots, O_T)$, we can train an HMM, that is, we can determine matrices A and B in Figure 1 that maximize the probability of this training sequence. The HMM training process can be viewed as a discrete hill climb on the high dimensional parameter space of the matrices A and B, and an initial state distribution matrix that is denoted as π. Once we have trained an HMM, we can use the resulting model, denoted $\lambda = (A, B, \pi)$, to compute a score for a given observation sequence—the higher the score, the more closely the scored sequence matches the training sequence.

The HMM matrix A is $N \times N$, while B is $N \times M$ and π is $1 \times N$. Here, N is the number of hidden states and M is the number of distinct observation symbols. All three of these matrices are row stochastic, that is, each row satisfies the conditions of a discrete probability distribution. To train an HMM, we specify N, the number of hidden states,

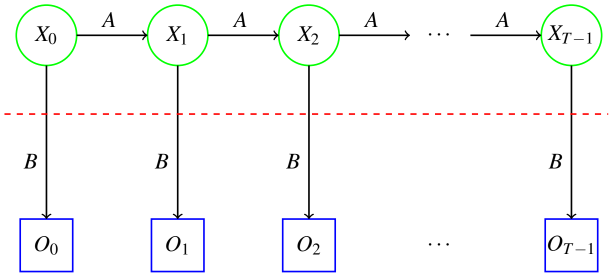

Figure 1: Hidden Markov model

while M, the number of distinct observation symbols, is determined from the data.

Typically, the matrices that define the HMM, i.e., $\lambda = (A,B,\pi)$, are initialized so that they are approximately uniform. That is, each element of A and π is initialized to approximately $1/N$, while each element of B is initialized to approximately $1/M$. In addition, each row is subject to the row stochastic condition. Also, we cannot use an exact uniform initialization as this represents a peak in the hill climb from which the model is unable to climb.

On the other hand, if we know something specific about the problem, we can sometimes use this knowledge when initializing the matrices, which can serve to speed convergence and reduce the data requirements. For example, in (Vobbilisetty et al., 2017) it is shown that an HMM can be used to recover the key in a simple substitution ciphertext, where the underlying language is English. In this case, the A matrix corresponds to English language digraph statistics, and hence we can initialize the A matrix based on such statistics, and there is no need to re-estimate A when training the model.

An HMM is a machine learning technique in the sense that very little is required of the human analyst. Specifically, we need to specify the number of hidden states N, but all other initial parameters are derived from the data, or can be generated at random. During training, we rely entirely on the "machine" (specifically, the HMM training algorithm) to generate the model. Surprisingly often, the HMM training algorithm succeeds in automatically extracting relevant and useful information from the data.

For additional information on HMMs, the standard reference is (Rabiner, 1989). The notation and description here closely follows that in the tutorial (Stamp, 2004).

In a classic illustration of the strengths of the HMM technique, (Cave and Neuwirth, 1980) show that HMMs can be successfully applied to English text analysis. In (Stamp, 2004), the specific English text example in Table 2 is given. In this case, the observations consist of the 26 letters and word space, for a total of $M = 27$ symbols, and the analyst chose to use $N = 2$ hidden states. The B matrix is initialized so that each element is approximately $1/27$, subject to the row stochastic condition—the precise initial values used in this example are given in first 2 columns in Table 2. After training the HMM using 50,000 observations, the resulting transpose of the B matrix is given in the final 2 columns of Table 2.

From the example in Table 2, we see that when the Markov process is in (hidden) state 1, the probability that the observed symbol is `a` is 0.13845, the probability that the observed symbol is `b` is 0.00000, the the probability that the observed symbol is `c` is 0.00062, the the probability that the observed symbol is `d` is 0.00000, the probability that the observed symbol is `e` is 0.21404, and so on. On the other hand, if the model is in (hidden) state 2, then the probability that the observed symbol is `a` is 0.00075, the probability that the observed symbol is `b` is 0.02311, the probability that the observed symbol is `c` is 0.05614, the probability that the observed symbol is `d` is 0.06937, the probability that the observed symbol is `e` is 0.00000, and so on. In this case, we can clearly see that the 2 hidden states correspond to consonants and vowels. Since no a priori assumption was made about the letters, this simple example nicely illustrates the "machine learning" aspect of an HMM.

	Initial		Final	
a	0.03735	0.03909	0.13845	0.00075
b	0.03408	0.03537	0.00000	0.02311
c	0.03455	0.03537	0.00062	0.05614
d	0.03828	0.03909	0.00000	0.06937
e	0.03782	0.03583	0.21404	0.00000
f	0.03922	0.03630	0.00000	0.03559
g	0.03688	0.04048	0.00081	0.02724
h	0.03408	0.03537	0.00066	0.07278
i	0.03875	0.03816	0.12275	0.00000
j	0.04062	0.03909	0.00000	0.00365
k	0.03735	0.03490	0.00182	0.00703
l	0.03968	0.03723	0.00049	0.07231
m	0.03548	0.03537	0.00000	0.03889
n	0.03735	0.03909	0.00000	0.11461
o	0.04062	0.03397	0.13156	0.00000
p	0.03595	0.03397	0.00040	0.03674
q	0.03641	0.03816	0.00000	0.00153
r	0.03408	0.03676	0.00000	0.10225
s	0.04062	0.04048	0.00000	0.11042
t	0.03548	0.03443	0.01102	0.14392
u	0.03922	0.03537	0.04508	0.00000
v	0.04062	0.03955	0.00000	0.01621
w	0.03455	0.03816	0.00000	0.02303
x	0.03595	0.03723	0.00000	0.00447
y	0.03408	0.03769	0.00019	0.02587
z	0.03408	0.03955	0.00000	0.00110
space	0.03688	0.03397	0.33211	0.01298
sum	1.00000	1.00000	1.00000	1.00000

Table 2: Initial and final B^T for English plaintext

For the example in Table 2, the converged A matrix as given in (Stamp, 2004) is

$$A = \begin{pmatrix} 0.25596 & 0.74404 \\ 0.71571 & 0.28429 \end{pmatrix}$$

This A matrix tells us that when the Markov process is in (hidden) state 1, the probability that it transitions to state 1 is 0.25596, while the probability that it transitions to state 2 is 0.74404. Similarly, if the Markov process is in state 2, it next transitions to state 1 with probability 0.71571, and it stays in state 2 with probability 0.28429. In this case, the A matrix is not particularly interesting, as this matrix simply gives the probability of transitioning from a consonant to a vowel, a vowel to a consonant, and so on.

As mentioned above, an HMM also includes an initial state distribution denoted as π, which for the example above converges (Stamp, 2004) to

$$\pi = \begin{pmatrix} 0.00000 & 1.00000 \end{pmatrix}$$

This tells us that the model started in the second hidden state which, according to the converged B matrix, corresponds to the vowel state. Again, this is not particularly enlightening. For this English text example, we see that the B matrix contains the interesting information.

Again, an HMM is defined by the 3 matrices, A, B and π, and it is standard practice to denote an HMM as $\lambda = (A, B, \pi)$. We also want to emphasize that each of these matrices is row stochastic, with each row representing a discrete probability distribution.

Now, suppose that we train an HMM with 2 hidden states on simple substitution ciphertext, where the plaintext is English. The resulting model will partition the ciphertext letters into those corresponding to consonants and vowels. On the other hand, if we set the number of hidden states N to equal the number of symbols (i.e., either $N = 26$ or $N = 27$, depending on whether we include word spaces), the simple substitution key can be easily determined from a converged B matrix of an HMM (Vobbilisetty et al., 2017). Furthermore, in this latter case, the A matrix contains digraph probabilities of the English plaintext.

An analogous HMM-based attack applies to homophonic substitution ciphers. However, in the homophonic substitution case, the key recovery from the B matrix is slightly more complex as the number of symbols mapping to each plaintext letter is typically unknown (Vobbilisetty et al., 2017).

For these HMM-based cryptanalytic models to converge, we generally require large amounts of ciphertext, making such attacks impractical for most classic cryptanalysis problems. However, since HMM training is a hill climb technique, random restarts can be used in an attempt to generate an improved solution. It is shown in (Berg-Kirkpatrick and Klein, 2013), and from a slightly different perspective in (Vobbilisetty et al., 2017), that by using large numbers of random restarts, the performance of HMM-based attacks can surpass other techniques, in the sense of requiring less ciphertext. For example, it is shown in (Vobbilisetty et al., 2017) that HMMs can outperform Jakobsen's algorithm (Jakobsen, 1995), which is a well-known general-purpose simple substitution solving technique that is based on digraph statistics.

In this paper, we consider HMM-based cryptanalysis of the classic Vigenère cipher. For the Vigenère cryptanalysis problem considered here, we will train an HMM, then we show that by examining the resulting matrices A, B, and π of a converged model, we can easily determine the Vigenère key.

2.4 Related Work

In (Berg-Kirkpatrick and Klein, 2013) an expectation maximization (EM) technique is applied to homophonic substitutions, with the goal of analyzing the unsolved Zodiac 340 cipher. The EM technique in (Berg-Kirkpatrick and Klein, 2013) is analogous to the HMM process discussed in the previous section. A novelty of this work is the use of an extremely large number of random restarts to improve on the hill climb results.

The paper (Lee, 2002) appears to be the first to explicitly apply HMMs (or similar) to substitution ciphers. However, the work in (Cave and Neuwirth, 1980), which focused on English text analysis, anticipates later cipher-based studies.

In (Vobbilisetty et al., 2017), HMMs are applied to simple and homophonic substitutions, and a careful comparison is made to other automated cryptanalysis techniques. This work shows that HMMs can achieve superior results in many cases, although the computational expense can also be quite high.

The work presented here is motivated in part by the recent paper (Gomez et al., 2018), where it is shown that a generative adversarial network (GAN), which is a type of neural network, can be used to successfully break a Vigenère cipher. However, this GAN-based Vigenère attack assumes unlimited ciphertext, which is unrealistic in any classic cryptanalysis context. In addition, in (Gomez et al., 2018) it is claimed that a strength of the GAN technique is its ability to handle a large vocabulary (up to 200 symbols), which seems to be of somewhat dubious value in the context of Vigenère cryptanalysis. Finally, as is generally true of neural networks, the resulting GAN is opaque, leaving the authors to make statements such as the following (Gomez et al., 2018):

> For both ciphers, the first mappings to be correctly determined were those of the most frequently occurring vocabulary elements, suggesting that the network does indeed perform some form of frequency analysis to distinguish outlier frequencies in the two banks of text.

The implication here is that the authors are forced to conjecture as to the relative importance of the various features in the GAN, since such basic information is not at all clear from an examination of the model itself.

In the next section, we give experimental results for an HMM-based attack on a Vigenère cipher. We also provide some discussion of our results, and we compare our technique to the GAN-based approach mentioned above.

3 Experimental Results

First, we train an HMM with $N = 3$ hidden states on a Vigenère ciphertext that was generated using the keyword CAT. Note that in this experiment we have selected the number of hidden states N to be equal to the keyword length. Also, we have used an observation sequence (i.e., English text) of length 1,000 extracted from the Brown Corpus (Francis and Kucera, 1969). In all of our experiments, we have removed all special characters and word space, and all letters have been converted to lower case, resulting in $M = 26$ distinct observation symbols.

For this experiment, the converged A matrix is give by

$$A = \begin{pmatrix} 0.00000 & 0.00000 & 1.00000 \\ 1.00000 & 0.00000 & 0.00000 \\ 0.00000 & 1.00000 & 0.00000 \end{pmatrix}$$

In contrast to the English text and simple substitution examples discussed in Section 2, here the A matrix is very informative—for one thing, this A matrix tells us that the transition between the $N = 3$ hidden states is actually deterministic. From the nature of the Vigenère cipher, it is clear that these states correspond to individual column shifts, and hence this is a result that we would expect for a keyword of length 3.

The corresponding B matrix appears in Table 3, which reveals that the first hidden state corresponds to a shift of 0 (i.e., keyword letter `A`), as the probabilities approximately match the expected letter frequencies of English. We also see that the second hidden state corresponds to a shift by 2 (i.e., keyword letter `C`) since the letter frequencies in this column are offset by 2 from those of English, while the final column corresponds to a shift by 19 (i.e., keyword letter `T`).

From the converged B matrix and the state transitions in the converged A matrix, we deduce that the keyword must be either `ATC`, `TCA`, or `CAT`. In this specific example, we also find that the initial state distribution matrix π converges to

$$\pi = \begin{pmatrix} 0.00000 & 1.00000 & 0.00000 \end{pmatrix}$$

a	0.08761	0.01290	0.04950
b	0.01560	0.00000	0.06811
c	0.03540	0.07411	0.00480
d	0.04290	0.01470	0.00450
e	0.13171	0.03030	0.04320
f	0.02100	0.04740	0.02520
g	0.02190	0.12181	0.06661
h	0.04170	0.02160	0.07291
i	0.06841	0.01470	0.02610
j	0.00180	0.04470	0.00060
k	0.00600	0.08101	0.06541
l	0.04080	0.00360	0.06541
m	0.02340	0.00300	0.09871
n	0.06001	0.04800	0.02520
o	0.08131	0.02280	0.01050
p	0.02430	0.06721	0.01680
q	0.00090	0.07711	0.00300
r	0.06601	0.02160	0.02130
s	0.06151	0.00090	0.00030
t	0.09481	0.06451	0.07951
u	0.02910	0.06541	0.01500
v	0.01020	0.09781	0.03090
w	0.01500	0.03180	0.03870
x	0.00180	0.00960	0.13171
y	0.01590	0.02040	0.02310
z	0.00090	0.00300	0.01290

Table 3: Final B^T for Vigenère ciphertext with keyword `CAT`

This implies that we start in the second hidden state, which corresponds to `C`, and hence we have determined that the keyword is `CAT`.

Suppose that instead of using $N = 3$ hidden states, we train an HMM with $N = 2$ hidden states using the same Vigenère encrypted data as in the previous example. In this case, we find that the A matrix converges to

$$A = \begin{pmatrix} 0.75236 & 0.24764 \\ 0.34235 & 0.65765 \end{pmatrix}$$

which tells us that we do not have deterministic transitions between the states, and hence the keyword length is greater than 2.

If, on the other hand, we attempt to train a model with $N = 4$ hidden states, we obtain

$$A = \begin{pmatrix} 0.00000 & 1.00000 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & 0.99884 & 0.00116 \\ 1.00000 & 0.00000 & 0.00000 & 0.00000 \\ 0.00000 & 0.15595 & 0.00000 & 0.84405 \end{pmatrix}$$

Since some state transitions are deterministic, we suspect that the keyword length is less than 4 in this case. Similarly, an HMM with $N = 5$ hidden states yields

$$A = \begin{pmatrix} 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0.00 & 0.47 & 0.00 & 0.00 & 0.53 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{pmatrix}$$

which, again, implies that the keyword length is likely less than 5. Finally, we point out that multiples of the keyword length behave similarly—for this example, with $N = 6$ hidden states we obtain

$$A = \begin{pmatrix} 0.00 & 0.00 & 0.54 & 0.00 & 0.46 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \\ 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\ 0.49 & 0.51 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \end{pmatrix}$$

From these results, we conclude that the A matrix in a converged HMM will enable us to precisely determine the keyword length used to encrypt a Vigenère ciphertext. Furthermore, if sufficient ciphertext is available so that English letter distributions are (roughly) apparent, the B matrix, together with the initial state matrix π, will completely determine the keyword. That is, we simply train HMMs with different values of N until we obtain a deterministic A matrix, and then we use the corresponding B and π matrices to determine the Vigenère key. Due to the fact that an HMM is a hill climb, to obtain a converged model, we might need to train each HMM multiple times with different randomly-selected starting values.

Next, we consider the amount of ciphertext needed to determine the Vigenère key using this HMM-based attack. Of course, the amount of ciphertext will depend on the length of the keyword.

We tested a few small keyword lengths until we found an initialization that yielded a solution. Then we reduced the amount of ciphertext until the HMM was unable to solve the problem. This gives us an upper bound on the amount of ciphertext needed, at least in these selected cases. In these experiments, we define a "solution" as a trained HMM where the average of the maximum value in each row of the A matrix is at least 0.99. Our results are given in Table 4, based on 100 random restarts of the HMM for each test case.

Keyword	Keyword length	Minimum ciphertext	Friedman test
IT	2	175	1.4235
DOG	3	250	3.7209
MORE	4	450	3.8208
NEVER	5	1200	3.6467
SECURE	6	1400	4.5545
ZOMBIES	7	1300	9.9334

Table 4: HMM attack (100 random restarts)

From the results in Table 4, it seems likely that with a large number or random restarts, we can significantly reduce the required length of the ciphertext. In any case, even the ciphertext lengths in Table 4 are far from the "unlimited" ciphertext that is assumed for the GANs training discussed in (Gomez et al., 2018). It is also interesting that our HMM result is accurate, even in cases where the Friedman test gives an incorrect result.

4 Conclusion

In this paper, we showed that a hidden Markov model (HMM) is a powerful and effective tool for the cryptanalysis of Vigenère ciphertext messages. We also showed that a trained HMM is informative in this context, in particular when compared to the neural network (GAN) based Vigenère attack discussed in (Gomez et al., 2018). Undoubtedly, GANs are powerful and useful tools for many types of problems. However, it appears that the Vigenère cipher may not be an ideal test case to illustrate the strengths of this particular type of neural network.

For future work, it would be interesting to test an HMM-based Vigenère attack with large numbers (i.e., millions) of random restarts to determine the minimum amount of ciphertext needed. It would also be interesting to test similar HMM based attacks on other more complex polyalphabetic substitutions. Various combinations of classic substitution and, perhaps, elementary transposition ciphers are also possibly amenable to HMM-based analysis. Due to their general applicability to classic substitution ciphers, HMMs might be useful as a first line of analysis in cases where a (classic) encryption technique is not completely known.

References

Taylor Berg-Kirkpatrick and Dan Klein. 2013. Decipherment with a million random restarts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 874–878.

Robert L. Cave and Lee P. Neuwirth. 1980. Hidden Markov models for English. In J. D. Ferguson, editor, *Hidden Markov Models for Speech*, IDA-CRD, Princeton, NJ, October 1980, pages 16–56. `https://www.cs.sjsu.edu/~stamp/RUA/CaveNeuwirth/index.html`.

W. Nelson Francis and Henry Kucera. 1969. The Brown Corpus: A standard corpus of present-day edited American English. `http://www.nltk.org/nltk_data/`.

William F. Friedman. 1987. *The Index of Coincidence and Its Applications in Cryptography*. Aegean Park Press.

Aidan N. Gomez, Sicong Huang, Ivan Zhang, Bryan M. Li, Muhammad Osama, and Lukasz Kaiser. 2018. Unsupervised cipher cracking using discrete GANs. `https://arxiv.org/abs/1801.04883`.

Thomas Jakobsen. 1995. A fast method for the cryptanalysis of substitution ciphers. *Cryptologia*, 19:265–274.

Friedrich W. Kasiski. 1863. *Die Geheimschriften und die Dechiffrirkunst (Cryptography and the Art of Decryption)*. Mittler und Sohn, Berlin. `http://pages.mtu.edu/~shene/NSF-4/Tutorial/VIG/Vig-Kasiski.html`.

Dar-Shyang Lee. 2002. Substitution deciphering based on HMMs with applications to compressed document processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1661–1666, December.

Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Mark Stamp. 2004. A revealing introduction to hidden Markov models. `https://www.cs.sjsu.edu/~stamp/RUA/HMM.pdf`.

Rohit Vobbilisetty, Fabio Di Troia, Richard M. Low, Corrado Aaron Visaggio, and Mark Stamp. 2017. Classic cryptanalysis using hidden Markov models. *Cryptologia*, 41(1):1–28.

Wing Wong and Mark Stamp. 2006. Hunting for metamorphic engines. *Journal in Computer Virology*, 2(3):211–229.