Heuristic Language Analysis: Techniques and Applications

Jeremy Lennert

March 11, 2001

Abstract

In their 1993 paper, "Statistical Techniques for Language Recognition: An Introduction and Guide for Cryptanalysts," Ganesan and Sherman present several statistical formulae useful for language recognition based on using a finite stationary Markov as a model to represent language. In my project, I verify the effectiveness of two of these formulae by examining the results when they are applied to various text samples of known entropy, and apply them to aid in automated ciphertext-only cryptanalysis of specific ciphers.

My automated cryptanalysis program functions by generating possible decryption keys using language heuristics and applying one of the aforementioned formulae to analyze the resulting decryption. The program continues to generate possible decryption keys until the formula no longer shows an overall increase in the likelihood that the resulting text is natural English.

1. Introduction

In February 1993, Ganesan and Sherman [1] presented statistical formulae for testing four statistical problems; this paper deals with two of these formulae. The first is a likelihood ratio test for recognizing a single known language and the second is a uniformly most powerful statistical test for distinguishing a known language from uniform noise. In this paper, their discriminatory power on text of known entropy is investigated, and then their application in the automated cryptanalysis of simple ciphers is explored.

Cryptographic Overview

Ciphers, commonly called "codes," are methods for disguising a message so that only the intended recipient can read the message. A certain secret, called a *key*, is known to the sender and the intended recipient that allows them to conceal and read the message using a cipher. *Cryptography* is the science of creating ciphers, and *cryptanalysis* is the science of "breaking" ciphers. *Cryptology* is both sciences.

A message in its normal, readable state is referred to as a *plaintext*. When a cipher is applied to a plaintext, it is said to be *encrypted*. The result is a *ciphertext*. By applying the inverse cipher to the ciphertext, it is *decrypted* to the original plaintext, and becomes readable again. The cryptanalysis described in this paper uses *ciphertext-only* attacks, which means it is known what the message looks like after it has been encrypted, but not what the original plaintext was.

2. Statistical Tests

In ciphertext-only cryptanalysis, it is sometimes difficult to know when the decryption is successful and the original plaintext is obtained, since the original plaintext is unknown. It is therefore necessary to determine whether a candidate plaintext is reasonable – for example, if the first line reads "Kienpoi aoieuli fskseoik 12xlkaoi7%," the decryption was probably not successful. However, if the first line reads "Attack at dawn!" then it is likely the correct decryption has been found, because the later looks like natural English.

The two statistical tests given by Ganesan and Sherman that are being used are useful for recognizing a known language. The difference between the two is that the first test generates the absolute probability that the text is the expected language, while the second assumes that the text is either the expected language or uniform noise (a random series of letters or, in the case of a computer, bits). Put another way, the first tests "English" versus "not English," and the second tests "English" versus "random." The two test statistics are given by the following equations:

Test #1	**Test #2**
$\Lambda_1 = \sum_{1 \le i,j \le m} n_{ij} \ln \left(\frac{p_{ij}}{\hat{p}_{ij}} \right)$	$\Lambda_2 = \left(\sum_{1 \le i,j \le m} n_{ij} \ln p_{ij} \right) + N \ln m$
m = size of alphabet (26 for English, 256 for ASCII) n_{ij} = number of occurrences of bigram ij p_{ij} = expected probability of occurrence of bigram ij $\hat{p}_{ij}$ = actual rate of occurrence of bigram ij	m = size of alphabet (26 for English, 256 for ASCII) n_{ij} = number of occurrences of bigram ij p_{ij} = expected probability of occurrence of bigram ij N = total number of overlapping bigrams (N = length of message - 1)

NOTE: The use of logarithms makes it impossible to compute the summation over i, j where $p_{ij} = 0$. Because skipping these intervals has the potential to invalidate the results, a very small value (10^{-4}) is substituted for p_{ij} where $p_{ij} = 0$.

In the case of a strong cipher, such as the new Advanced Encryption Standard (AES), it is impossible to have a partially correct decryption; it is an all-or-nothing proposition, and the cipher is specially designed so that incorrect decryptions look like random noise. For this reason, the assumption made by the second test is valid – it will be either a language or random noise. However, in the case of weak ciphers (those that are easier to cryptanalyze) this may not be the case, because accuracy may not be as absolute; it is possible to get only part of a key correct, resulting in some letters or words correctly decrypted, but others still unreadable. One such cipher is called a substitution cipher, in which each letter (or group of letters) is mapped to another arbitrary letter (or group of letters). For this reason, it is necessary to compare the results of both tests on various texts of known quality to determine their effectiveness at detecting correct decryptions of weak ciphers.

Both tests rely on knowing the expected frequency of bigrams (pairs of adjacent characters) in the target language. A fairly large sample of text (approximately 130,000 characters) was collected from the bodies of articles of on-line newspapers and was analyzed to obtain these bigram frequencies. Afterward, the text sample was duplicated five times, and in each of the five copies a random selection of characters were arbitrarily replaced with 0-2 random characters (approximately one of every fifteen letters in the first copy, one of every twelve in the second, one of every nine in the third, one of every seven in the fourth, and one of every five in the fifth).

Both tests were then used to evaluate first the original sample, and then each of the five copies. This tests the effect of partial decryptions, random noise, and missing data on their results. The following data was obtained:

Original Sample	–	$\Lambda_1 = 0$		$\Lambda_2 = -14934.8$	
Copy #1 (1/15 altered)	–	$\Lambda_1 = 10050.9$		$\Lambda_2 = -56608.4$	(+379.0%)
Copy #2 (1/12 altered)	–	$\Lambda_1 = 11333.2$	(+12.8%)	$\Lambda_2 = -65113.5$	(+15.0%)
Copy #3 (1/9 altered)	–	$\Lambda_1 = 14817.6$	(+30.7%)	$\Lambda_2 = -80781.8$	(+24.1%)
Copy #4 (1/7 altered)	–	$\Lambda_1 = 18009.2$	(+21.5%)	$\Lambda_2 = -95978.3$	(+18.8%)
Copy #5 (1/5 altered)	–	$\Lambda_1 = 22494.7$	(+24.9%)	$\Lambda_2 = -122426$	(+27.6%)

As expected, both tests scored the increasingly corrupt texts increasingly poorly. For the first test, lower scores indicate a higher probability that the sample corresponds to the known language; for the second test, higher scores indicate the same probability.

Because both tests produced equivalent results, the second test was used for cryptanalysis, because it requires less time to calculate (since the expected frequencies are known in advance, their natural logarithms can also be calculated in advance; the actual rate of occurrence is not known until run-time).

3. Cryptanalysis

The introduction of computers to the field of cryptology (during and after World War II) revolutionized the field, primarily because it became feasible to "brute force" a cipher by decrypting with every possible key until the correct decryption was found (provided there are few enough possible keys). This has rendered virtually all ciphers that are usable without computers obsolete. With the aid of computers, cryptanalysis of many traditional ciphers can be done in minutes or seconds, rather than days.

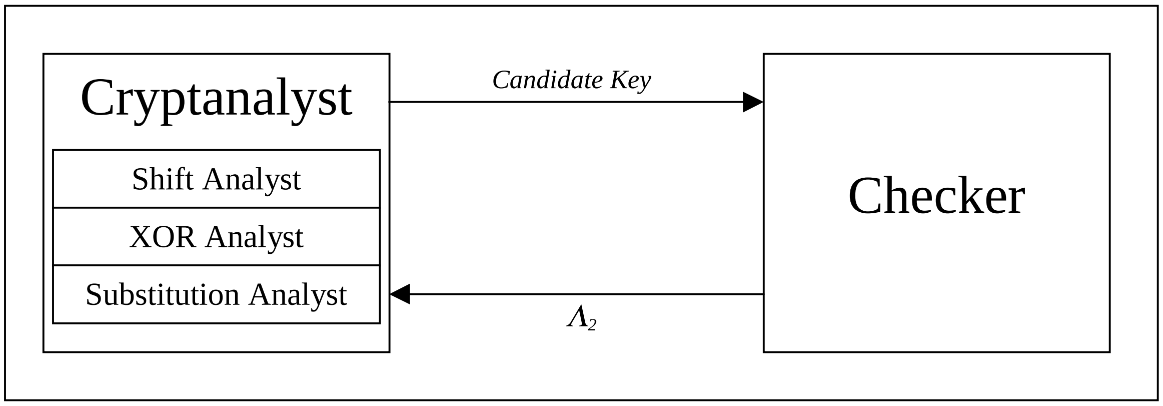
In this project, a computer program was developed using one of Ganesan and Sherman's statistical formulae (see above) to automatically cryptanalyze certain ciphers. Three simple ciphers are dealt with here: shift, eight-bit exclusive or (XOR), and general substitution.

Program Structure

There are two major pieces to this automated decryption program: a cryptanalyzing piece, which generates candidate keys, and a checker piece, which uses one of Ganesan and Sherman's formulae to evaluate those guesses.

The cryptanalyzing piece is further divided into algorithms for cryptanalyzing different ciphers. When one of these algorithms is activated, it is given a time limit to find the best key it can. It will continue generating candidate keys until the time limit is up or it has exhausted its search space. Afterwards, another algorithm may be called if no key found using the first cipher looks promising to the checker.

Keys are stored internally as a special data type, and each key knows how to use itself to decrypt a message.



Once the program is done, the checker reports the key with the best score so far, and this is used to decrypt the original ciphertext. The resulting plaintext and the key are reported to the user, along with the Λ_2 score.

Shift Cipher

The shift is among the oldest known ciphers. It works by "adding" a number to every letter in the plaintext to encrypt it; that is, by moving each letter forward through the alphabet by a certain number of letters. If the end of the alphabet is reached, we start again at "A."

The most famous example of this cipher is the Caesar Cipher, which uses three as its key. To encrypt with the Caesar Cipher, each letter is rolled forward three letters through the alphabet.

```
plaintext:  THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
ciphertext: WKH TXLFN EURZQ IRA MXPSV RYHU WKH ODCB GRJ
```

Encryption by Caesar Cipher

One of the major problems with the shift cipher is that because there are only 26 letters in the English alphabet, there are only 26 keys for the shift cipher (and one of them puts each letter back to itself, so the ciphertext is the same as the plaintext). This is trivial to brute force, even without a computer.

Computers, rather than using a 26-letter alphabet, actually use a 256-letter alphabet called ASCII. This improves security a bit, but it is still trivial to brute force, particularly because in computers, a space bar is treated like a letter when encrypting. Because the space character is so common (nearly 20% of the letters in natural English), whatever key causes the most common letter to decrypt to a space is virtually guaranteed to be the correct key.

It is still worthwhile, however, to use the statistical test to verify that this decryption is accurate, because in an unusual message with many long words, space might not be the most common character. This computer program decrypts the shift cipher by listing all the letters in the ciphertext in descending order of frequency, and then testing keys that map each of those letters, in sequence, to the letter expected to occur most frequently. Usually the first guess is correct, so this algorithm is generally run with a very short time limit to avoid wasting time.

Eight-Bit XOR Cipher

The eight-bit XOR is very similar to the shift cipher, and sometimes used on computers. XOR, often indicated by a "⊕" symbol, is a logical operator such that "a ⊕ b" means "a or b but not both." For this cipher, a *bitwise xor* is used, which means each bit in a number is xor-ed to produce a new number. If the binary number 1010 were to be XOR-ed with the binary number 1100, the result would be 0110 (the first bit of each number is XOR-ed, then the second, and so on). An eight-bit XOR is used because computer letters are eight bits long. In this cipher, a certain string of eight bits is chosen at random, and XOR-ed with each letter of the plaintext to produce the ciphertext.

Any computer program designed to use (or cryptanalyze for) a shift cipher can be changed to do so for an eight-bit XOR very easily, since the only difference is the mathematical operation (⊕ instead of +), and this program cryptanalyzes for eight-bit XOR ciphers in exactly the same way as shift ciphers.

General Monoalphabetic Substitution Cipher

In a general substitution cipher, each letter (or group of letters) is mapped to an another arbitrary letter (or group of letters). To decrypt, the mapping is reversed. In a monoalphabetic substitution, the mappings only involve single letters. This is probably the most well-known class of ciphers. One possible key for a substitution cipher is at the right.

```
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
PBT KXUDA MLJWV ZJR IXYSC JQTL PBT EHFG OJN
```

Encryption with General Substitution Cipher

This particular key has a few non-optimal characteristics; for example, the letter "W" is mapped to itself. Still, the message would be very difficult to decrypt without the key.

For standard English, there are a total of 26! ≅ 4.0 x 10^{26} possible keys, and for computer ASCII, there are 256! ≅ 8.6 x 10^{506} possible keys, far too many for a brute force search. However, substitution ciphers are susceptible to heuristic attack because letters in the ciphertext occur with the same frequency as their counterparts in the plaintext.

This automated decryption program attacks the substitution cipher by starting with a randomly-chosen key, applying Ganesan and Sherman's statistical test, and then considering every possible exchange of two letters in the key and the change it would have on the

PLAINTEXT	CIPHERTEXT
A	H
B	M
C	D
D	O
E	T
F	Z
G	N
H	B
I	U
J	I
K	A
L	E
M	Y
N	V
O	J
P	S
Q	K
R	L
S	C
T	P
U	X
V	Q
W	W
X	R
Y	G
Z	F

General Substitution Key

result of the statistical test. Whichever change would most increase the resulting score is adopted as part of the key, and the process is repeated. This continues until no character swap has a beneficial effect on the score.

Although this attack is highly effective, it still does not reliably decrypt uncommon characters, such as capital letters, numbers, and rare punctuation marks. The resulting decryption is generally still legible to humans.

However, this attack is still very costly in computing time. Running on a computer with a 650 MHz processor, finding a good decryption by this method still takes approximately twelve hours.

Previous Attempts at Decryption

Hunter and McKenzie [2 – 1983] used a relaxation algorithm to try to break a simple substitution cipher. This algorithm operates by scanning every *trigram* (series of three consecutive letters) in the ciphertext, and evaluating the probability that the middle ciphertext letter decrypts to a certain plaintext letter based on the existing probabilities of the adjacent letters. The algorithm repeats until all probabilities are evaluated at 100%.

It was determined that this technique's degree of success is determined in large part by the accuracy of the trigram data for describing the correct decryption. With precisely matching trigram frequency data, the relaxation algorithm was able to find a perfect decryption within the first few iterations; with less accurate data, no perfect decryptions were found, even after many iterations.

Carroll and Martin [3 – 1986] developed an expert system to cryptanalyze substitution ciphers. This algorithm evaluated probabilities of particular decryptions of individual letters using letter frequency, and additional refinement based on "illegal" *bigrams* (series of two consecutive letters), single-letter words, word beginnings, and word endings, and also using bigrams and trigrams considered "common." The program functioned fairly well without intervention, but was designed to utilize human input during the decryption process.

Michael Lucks [4 – 1998] created a computer program that utilized a constraint satisfaction algorithm to decrypt substitution ciphers. The algorithm guessed decryptions for entire words based on repeated letters using an on-line dictionary. The program was able to find a perfect decryption with no intervention 60% of the time, and in many other instances was able to achieve a correct decryption with very little human intervention. However, the required on-line dictionary is very large, so this approach requires fairly powerful hardware.

4. Conclusions

Two statistical tests presented by Ganesan and Sherman were successfully tested and applied. Tests suggest that they are approximately functionally equivalent in detecting entropy in a text of known language.

The automated cryptanalysis of three ciphers (shift, eight-bit XOR, and substitution) was largely successful utilizing one of these statistical tests. The shift and XOR ciphers were essentially trivial to decrypt, while the general monoalphabetic substitution cipher took approximately one half-day to decrypt, and the decryption was readable but imperfect.

References

[1] Ganesan, Ravi; and Alan T. Sherman, "Statistical Techniques for Language Recognition: An Introduction and Guide for Cryptanalysts," (February 1993)

[2] Hunter, D.G. N.; and A.R. McKenzie, "Experiments with Relaxation Algorithms for Breaking Simple Substitution Ciphers," *Computer Journal* vol. 26 no. 1 (1983)

[3] Carroll, John M.; and Steve Martin, "The Automated Cryptanalysis of Substitution Ciphers," *Croptologia* vol. 10 no. 4 (Octoboer 1986)

[4] Lucks, Michael, "A Constraint Satisfaction Algorithm for the Automated Decryption of Simple Substitution Ciphers," (1998)