Fall 2017 CIS 3362 Homework #3 SOLUTION: Classical Ciphers – Code Breaking

Overview: In this homework assignment, you were required to break both substitution and Vigenere ciphers. Both of these ciphers have a key space too large to reasonably brute-force; the best alternative is to utilize letter frequency analysis. Luckily, the “cryptool” application found on the course webpage has an assortment of tools that are specifically tailored to assisting with frequency analysis of the substitution and Vigenere ciphers – I recommend taking advantage it!

For substitution, the method is straightforward: determine the frequency of each character in the ciphertext and compare these frequencies with those of the English alphabet. By matching the most common characters to the most-used characters in the English alphabet (‘e’, ‘t’, ‘a’, etc.), it only takes some guess-and-check to find patterns that begin to make sense. Additionally, recognizing double letters and repeated n-grams can help you narrow down your options.

The Vigenere cipher was designed to disrupt frequency analysis; however, using either the Kasiski test or index of coincidence analysis will help you overcome this obstacle. Both of these methods are described in detail in the Chapter 3 Lecture Notes.

Methodology (Questions 1 and 2): When faced with the task of breaking a substitution cipher, a good place to start is testing every combination of the first five most frequently used letters in the English alphabet against the first five most frequently used letters in the ciphertext. Often, incorrect key choices will present a plaintext with unusual combinations of letters, helping you narrow down your options. When you feel like you have gotten a few of the letters correct, look for common n-grams such as “the” and “ing”, double letter combinations, and any longer repeating n-grams to facilitate working out the rest of the key. Using these techniques, I was able to determine “goodluck” and “hopefully” from questions 1 and 2, respectively; from these I was able to jumpstart the rest of the code-breaking process.

1) Decode the following message, which was encrypted using the substitution cipher. Make sure to discuss all the steps you took, the key you arrived at, and the decoded message.

Ciphertext:
frajpssycncmkxactkepfmgnphfrhdohgchnhnknoaacipfkhnphjcznhdkpdchxardskjphcijpyd
khnkxactkgrosmzkhdsrgfpgpypfygpycxardpdsynpukjyoeopsxphhkafecfhncejkeepzkhnph
enrosmjpvkchkpeckahrdakpvzrrmsoivgchnoecfzskhhkalakwokfiymczapjpfmhaczapjcflraj
phcrfpfmboehlravcivecgcssakxkphpsrfzgramhnphcepsakpmycfhncejkeepzkchcecflrajphc
rfzrrmsoiv

Plaintext:
normallyihideprizesandwhatnotbutwiththehurricanethatmightbeabitproblematicmaybethe
prizewouldgetblownawayanywayiprobablyhavemyusualpatternsinthismessagethatshould
makeiteasiertobreakgoodluckwithusingletterfrequencydigramandtrigraminformationandju
stforkicksiwillrepeatalongwordthatisalreadyinthismessageitisinformationgoodluck

Key:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
r	j	i	b	s	n	w	t	c	m	e	f	d	h	u	a	x	o	l	z	v	k	q	p	y	g

2) Decode the following message, which was encrypted using the substitution cipher. Make sure to discuss all the steps you took, the key you arrived at, and the decoded message.

Ciphertext:
rkwxruurlsrlwtgnnroegluajqxfktklfrurxzjruexjlltmeekxrbtogjbfakveakfobkxxugfktcjurxjp
pgxerurxceuuequarxmktkxrsguwgqemgqdfglectwtxebnagpenjbbttgjkbbmeqeelhgtrlsuaef
ktkxmebbklfagpenjbbtuarxajqqroklefgexluogweuggobgxeuggqbklfgmebbpkquxgnrufenrl
ruebtmrbbcjuagpenjbbtuaeoelueqmrbblgulguwjoaebxeugpjurluarxwexxksexrloerkwlguar
frlskpqrye

Plaintext:
iamsittinginmyofficeonthursdayanditisquitesunnyweeasilycouldhavehadclasstodaybutisu
pposeitisbetterthiswayasigotmoreworkdonebymyselfhopefullyyouallwereenjoyingthedaya
swellandhopefullythishurricanedoesntcometooclosetoorlandowellpartsofitdefinitelywillbu
thopefullythecenterwillnotnotmuchelsetoputinthismessagesinceiamnothidingaprize

Key:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
h	l	b	k	e	d	o	j	x	u	a	n	w	f	c	p	r	i	g	y	t	v	m	s	z	q

Methodology (Questions 3 and 4): As mentioned earlier, you have two tools to choose from when trying to break the Vigenere cipher: the Kasiski test and index of coincidence analysis. To use the Kasiski test effectively, you need to find an n-gram that repeats at least three times. Unfortunately there were no suitable options between questions 3 and 4. This left only index of coincidence analysis which is a bit more guess-and-check. Using the cryptool, you can determine that the highest index of coincidences (*indices of coincidence?*) occur when the key lengths are 10 and 11 for questions 3 and 4, respectively. Using the Vigenere decryption tool within the cryptool app, you can match letter frequencies within each bin to those of the English alphabet. Through trial and error, a keyword should begin to emerge. To then determine the plaintext, I wrote a program that could perform the Vigenere decryption algorithm. This code (written in Java) is included below.

3) Decode the following message, which was encrypted using the Vigenere cipher. Make sure to discuss all the steps you took, the key you arrived at, and the decoded message.

Ciphertext:
nenusyegjlegnpwzealpffgzcohojvvsjkwoddirsaoomyaoevzwvoztwjvwfsxldyuselxmngoks
vzfyifwcaxouevcnxgqpvrwjtbumuofvdcllusmhzpletrusepwejrtgkshafpovafwmaqocojhbxz
pccjhvizlhmpwfqxazhcdnsrgkhrfvmlhvuzngksokrvefoxdgkkinhclxqcietprlaehfqwrzqxtul
gfdwjzrsrqqdudkhuuflbspvvyzgrqsdaqzsyeelvalsprlwrusmxzvwfuglzuvsdxkunowzzsorwc
blboervqtebuamupvbfusrizzoihgenwspscigkhnwweebjbecjlhtpvvnvyjrfphsejkhrlhtafndph
bsskkixhktbusmzhyhdefvosapvifrrwvqdcdhnoenweziv

Plaintext:
wearemissingaminimumoftwoclasseswhichprobablymeansiwillhavetoskipmyenigmaday
whichistoobadbecausethatstoryisreallyreallyinterestingattheveryleastyouallshouldgoseeim
itationgameorreadthechapterinthecodebookasyouwellknowifyouarereadingthisthiswasenc
ryptedusingthevigenerecipherihopeyouwereabletoutilizemutualindexofcoincidenceormay
beyoufoundarepeatedwordthatisinsyncwiththekeywordlengtheitherwayhopeyougettoreadt
hemessagebeforethehomeworkisdue

Key: randomword

4) Decode the following message, which was encrypted using the Vigenere cipher. Make sure to discuss all the steps you took, the key you arrived at, and the decoded message.

Ciphertext:
tevrfusdmvztagnypsklgkeurymiyxbqtcmnqizetgdwywghyksnaglsmjullexausrmvycvamuc
qmmuzubobtnelsronmslqgjtahvjwkxegottyyymfhekottyxynjtqpxcwzamqfzbjwwmgzyzw
ywtlsmekmbgybjwxmfyuowppqadiybcvuaqrefvxghifbucdzvnzpazhevzlbvylqyabvztavybi
jwuqkeebxxauhvvstvnibijpqvzlapocdseakpegcroaxhcthwcflicjitwkfvbnullhoqlzacghqkaa
qjcggncaaltqldwsibijlaqaniaynwhgwy

Plaintext:
iwillmakethisthehardestmessagetobreakormaybenotbecausethewordmessageisinitmaybeht
atwillmakeiteasierisupposewewillseenowiwillrefrainfromusingtheletterefortherestoftheme
ssagewhoisalvinchipmunkananimalonatvshowthatisrightthisishardmostwordscontainachar
withasciivalsixtyplusfourplusfourplusunohadtogotospanishsorrythisislotsofcharssoicansto
pnow

Key: linguistics

Vigenere Cipher Decryption (in Java)

```java
// This program was written to assist with the decryption of a Vigenere cipher.
// The main method can use an entered ciphertext and key to decrypt the cipher.
// The class can also assist determing the GCD thruogh the use of the
// getNGramDiffVals() method.

public class VegenereDecrypt {
        public static void main(String[] args) {
                // If you have an n-gram suitable for the Kasiski test, enter it here.
                String nGram = "<Write n-gram here>";
                String cipherText = "<Write ciphertext here>";
                String key = "<Write key here>";

                // Uncomment the line below if you have an n-gram to use for the Kasiski test.
                /* getNGramDiffVals(cipherText, nGram); */

                // The loop below will print plaintext using the keys and ciphertext
                // provided above. (Useful for determining characters in the key.)
                String plainText = "";
                for(int i = 0; i < cipherText.length(); i++) {
                        if(key.charAt(i % key.length()) == '-')
                                plainText += '-';
                        else
                                plainText += (char) ((((cipherText.charAt(i) - (key.charAt(i % key.length()))) +
                                                        26) % 26) + 'a');
                }

                System.out.println(plainText);
        }

        // Given an n-gram (that repeats at least 3 times) corresponding to a given ciphertext,
        // you can use the getNGramDiffVals()method to determine the difference in character
        // index between three instances of the n-gram. The GCD between the two difference values
        // produced can be used for the Kasiski test.
        public static void getNGramDiffVals(String cipherText, String nGram) {
                int firstIndex = cipherText.indexOf(nGram);
                int nextIndex = cipherText.indexOf(nGram, firstIndex + nGram.length());
                int lastIndex = cipherText.indexOf(nGram, nextIndex + nGram.length());
                System.out.println("Difference between first two nGrams: " + (nextIndex - firstIndex));
                System.out.println("Difference between second two nGrams: " + (lastIndex - nextIndex));
        }
}
```