Cryptography I
Exercise sheet 2

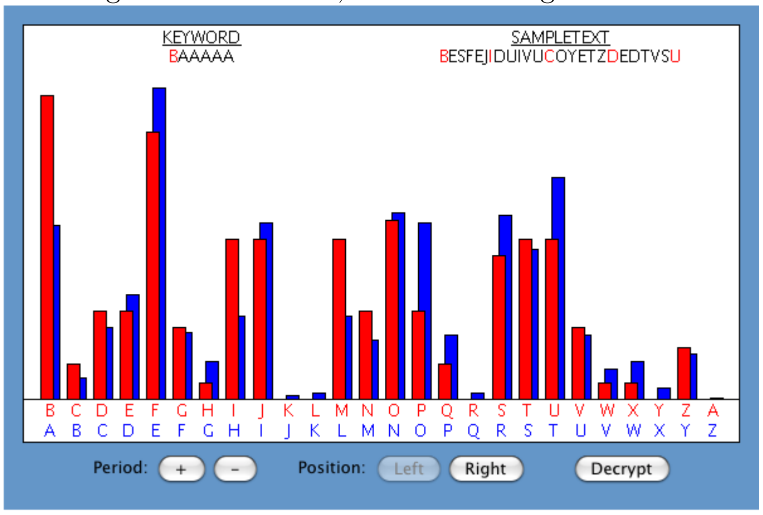
Ilja Kuzovkin

September 29, 2010

1 Exercise 2a

1.1 Applet

I was able to break the text using this applet[1]. First step was to determine length of the key. Because text was long enough, when I got to the length of 6, is became obvious, in comparsion with lengths I tried before, that 6 is the right one.


KEYWORD
BAAAAA
SAMPLETEXT
BESFEJIDUIVUCOYETZDEDTVSU
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 A
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
Period: + -
Position: Left Right
Decrypt

So. the first letter was "B", finding other letter was even easier. In the end I got "BAQRRR" for the key and

be considered coincidence, but a strange one. (The rest of my book is scarcely at all like Randś-thank goodness.) I wrote A Time of Changes in the summer of 1970, and it was, I suppose, my response to all that had happened in the last few years of the 1960ś, that time of changes for so many of us. I had been as rigid and controlled as anyone else in the old pre-Beatle, pre-psychedelic, pre-revolutionary world of the Eisenhower years, and I had been rocked by transformations in the crazy decade that followed, transformations that had altered my attitude toward life, my way of dress, my work, and just about everything else. In 1970 I hovered emotionally and spiritually somewhere between New York and California, between the old life and the new, and I oscillated uncertainly, not yet having opted fully for California; and A Time of Changes is the record of that inner upheaval, altered by the metaphors of science fiction but thoroughly recognizable for what lay behind them. (Some of my more straigh

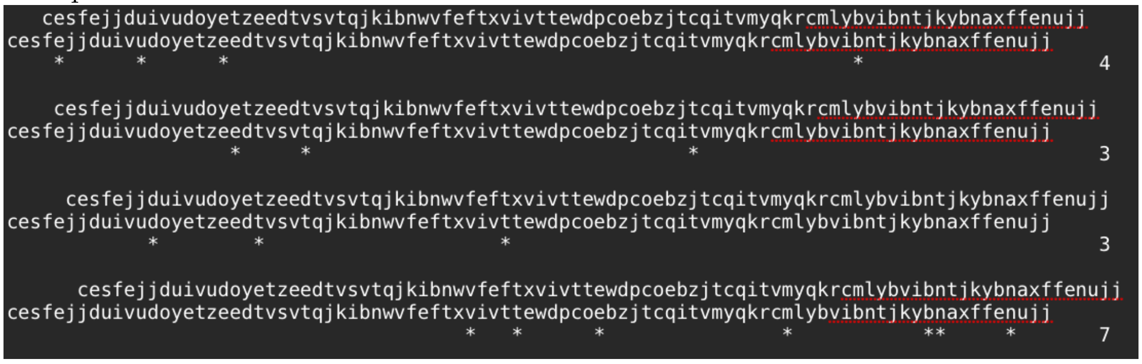
for the plain text.

1.2 Background

Now, to show understanding of the background work, I'll describe how it could be done by hand.

1. Take an usual letter frequency table for the english language
2. Count the letter coincidences between original cypher text and the shifted one
   (a) Put tho copies of cypher text one below another
   (b) Shift one of them to n positions
   (c) Count amount of letter, which are on the same places
   (d) $n + 1$ and do the steps a, b, c
   (e) Stop when for certain shift the amount of coincidences will be markably greater

   example

```
   cesfejjduivudoyetzeedtvsvtqjkibnwvfeftxvivttewdpcoebzjtcqitvmyqkrcmlybvibntjkybnaxffenujj
cesfejjduivudoyetzeedtvsvtqjkibnwvfeftxvivttewdpcoebzjtcqitvmyqkrcmlybvibntjkybnaxffenujj
    *      *      *                                                     *                    4

    cesfejjduivudoyetzeedtvsvtqjkibnwvfeftxvivttewdpcoebzjtcqitvmyqkrcmlybvibntjkybnaxffenujj
cesfejjduivudoyetzeedtvsvtqjkibnwvfeftxvivttewdpcoebzjtcqitvmyqkrcmlybvibntjkybnaxffenujj
                   *     *                                *                                  3

     cesfejjduivudoyetzeedtvsvtqjkibnwvfeftxvivttewdpcoebzjtcqitvmyqkrcmlybvibntjkybnaxffenujj
cesfejjduivudoyetzeedtvsvtqjkibnwvfeftxvivttewdpcoebzjtcqitvmyqkrcmlybvibntjkybnaxffenujj
            *        *                    *                                                  3

      cesfejjduivudoyetzeedtvsvtqjkibnwvfeftxvivttewdpcoebzjtcqitvmyqkrcmlybvibntjkybnaxffenujj
cesfejjduivudoyetzeedtvsvtqjkibnwvfeftxvivttewdpcoebzjtcqitvmyqkrcmlybvibntjkybnaxffenujj
                                       *   *      *               *           **     *       7
```

   Even on such small piece of text we already can see, that $shift = 6$ gives us more coincidences. So, let's assume, that key length is 6.

3. Create letter frequency tables for every nth letter in the cypher text, $1 \leq n \leq 6$ - so we will get 6 tables, each of them shows the frequency of letter appearence on the nth place. We call them $K1 \dots K6$
   (a) Get list of all letters, which occure on position n
   (b) From this list create the letter frequency table
   (c) Repeat steps a, b for $i \in \{1, 2, 3, 4, 5, 6\}$
4. Create letter frequency tables for every letter in the alphabet
   (a) For letter "a" it will be the frequency table from step 1.
   (b) To get table for letter "b" we just shift values from "a"-table one position to the right
   (c) To get table for letter "c" we shift values from "a"-table two positions to the right
   (d) Repeat steps for each letter - as a result we will get 26 tables, we call them $A1 \dots A26$
5. Now we are ready to find the key
   (a) Represent all tables we have as 26-element vectors
   (b) compute dot products $< Ki, Aj >= N_{ij}$ where $i \in 1 \dots 6$ and $j \in 1 \dots 26$. We will get 26 numbers for each of 6 Ki tables
   (c) Find the $max(N_{i1} \dots N_{i26}) = M_i$ for $i \in \{1 \dots 6\}$
   (d) Now, if for example $N_{1,10} = M_1$ - that will mean, that most possible first letter of the key is letter 10, which is "j"

   (e) If for example $N_{2,17} = M_2$ - then most possible second letter of the key is letter 17, which is "q"

   (f) And so on until we find all 6 letters of the key

6. We have the key now - we can decrypt the cypher text.

2 Exercise 2b

Our first step will be computing probability distribution of ciphertexts $c_1 \dots c_6$. This can be done by summarizing product of all pairs $P(key_i) * P(plaintext_j)$ which give us ciphertext. For example for c_1 it will be

$$c_1 = k_1 * p_6 + k_2 * p_1 + k_3 * p_1 + k_4 * p_6 + k_5 * p_2 + k_6 * p_3 = 0.2$$

using small python script, which you can find in appendix we will find all other c_i's
$c_1 = 0.2$
$c_1 = 0.155$
$c_1 = 0.16$
$c_1 = 0.18$
$c_1 = 0.17$
$c_1 = 0.135$
Now we can find the entropy of P,K and C using the following formula

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

They appear to be
$H(P) = 2.50370169606$ bits
$H(K) = 2.52821294584$ bits
$H(C) = 2.57420514662$ bits
Now we are ready to calculate $H(K|C)$ using the folmula

$$H(K|C) = H(K) + H(P) - H(C)$$

Proof: $H(K,P,C) = H(C|K,P) + H(K,P)$, since each pair of key and plain text determine ciphertext uniquely $H(C|K,P) = 0$, which gives us $H(K,P,C) = H(K,P)$. Because K and P are independent $H(K,P) = H(K) + H(P)$, so now we have $H(K,P,C) = H(K) + H(P)$.
In the same way we can reduce $H(K,P,C) = H(P|K,C) + H(K,C) = 0 + H(K,C) = H(K,C)$.
Now we can transform $H(K|C)$ into desired formula $H(K|C) = H(K,C) - H(C) = H(K,P,C) - H(C) = H(K) + H(P) - H(C)$
Using this formula we get

$$H(K|C) = 2.52821294584 + 2.50370169606 - 2.57420514662 = 2.45770949528$$

I have not found such an easy solution for $H(P|C)$, so here we will have to compute all $P(p,c), p \in plaintextdistributions, c \in ciphertextdistributions$ using the following folmula

$$P(p|c) = \frac{P(p) \sum\limits_{K:p=d_k(c)} P(k)}{\sum\limits_{K:c \in C(K)} P(k) P(p = d_k(c))}$$

where: $K : p = d_k(c)$ can be read as "keys, which give us p when decrypting c"
$K : c \in C(K)$ - "keys, which can be used to get ciphertext c"

$p = d_k(c)$ - "plain text, which we get decrypting c using key k"

Calculations are made in the same python script.
After compyting those 36 values we can use them in formula

$$H(P|C) = -\sum_c \sum_p P(c)P(p|c)\log_2 P(p|c)$$

and calculate (also in the script)

$$H(P|C) = 1.81658688075$$

which seems to be something, what may be correct :)

A Python sctipt

```
#!/usr/bin/python

import math

pd = [0.2, 0.1, 0.15, 0.1, 0.2, 0.25]
kd = [0.15, 0.15, 0.25, 0.2, 0.1, 0.15]
mapping = [[4,3,5,2,6,1], [1,6,3,4,2,5], [1,6,4,2,5,3], [4,3,6,5,2,1], [6,1,2,3,4,5],
           [3,6,1,5,2,4]]
cd = [0,0,0,0,0,0]

for c in range(0,6):
for k in range(0,6):
cd[c] += kd[k] * pd[mapping[k].index(c + 1)]

print ""
print "Plain text distribution " + str(pd)
print "Key distribution " + str(kd)
print "Cipher text distribution " + str(cd)

ent_p = 0;
ent_k = 0;
ent_c = 0;
for i in range(0,6):
ent_p += pd[i] * math.log(pd[i],2)
ent_k += kd[i] * math.log(kd[i],2)
ent_c += cd[i] * math.log(cd[i],2)

ent_p *= -1
ent_k *= -1
ent_c *= -1

print ""
print "Plain text entropy " + str(ent_p) + " bits"
print "Key text entropy " + str(ent_k) + " bits"
print "Cipher text entropy " + str(ent_c) + " bits"
print ""


#
# H(P|C) starts here
#

# mapping2 [plain][cypher] = [key,key,...]
# mapping3 [plain][cypher] = sum P(k1) + P(kn) + ..
# mapping4 [cypher] = P(k1) * P(p1) + ...
mapping2 = [[list() for col in range(6)] for row in range(6)];
mapping3 = [[0.0 for col in range(6)] for row in range(6)];
mapping4 = [0.0 for col in range(6)];
for k in range(0, 6):
for p in range(1, 7):
for c in range(1, 7):
if (mapping[k].index(c) == p-1):
mapping2[p-1][c-1].append(k+1)
mapping3[p-1][c-1] += kd[k]
mapping4[c-1] += kd[k]*pd[p-1]

# P(x|y)
pr_p_c = [[0.0 for col in range(6)] for row in range(6)];
for p in range(0, 6):
for c in range(0, 6):
pr_p_c[p][c] = (pd[p] * mapping3[p][c]) / (mapping4[c])

h_p_c = 0.0;
for c in range(0,6):
for p in range(0,6):
if (pr_p_c[p][c] != 0):
h_p_c += cd[c] * pr_p_c[p][c] * math.log(pr_p_c[p][c],2)

h_p_c *= -1

print "H(P|C) = " + str(h_p_c)
```

References

[1] http://math.ucsd.edu/ crypto/java/EARLYCIPHERS/Vigenere.html