February 7th 2017

Assignment 2

COMP 7041 - Transposition Cipher

Filip Gutica - A00781910
Shahin Nikvand - A00809275

Introduction

In COMP 7401 we learned about basic methods of cryptography such as Caesar Cipher and the Transposition Cipher. We discovered that a lot of the basic ciphers suffer from things like Frequency analysis and can be effortlessly defeated with a little bit of patience.

Today we will be encrypting files using the transposition cipher and breaking the encryption through a simple brute force approach. The plan is to attempt every single key length and see if we can detect english words from the output of the brute force.

Constraints

We will be limiting our development to the following:

1. Python Language
2. Using a built in C based Python library called pyenchant which can use multiple dictionaries from different languages as well as personal word lists.
    a. python3 -m pip install pyenchant
3. File IO is a must
4. All the information will be passed through command line arguments to the application
5. Demonstrate the applications capabilities using War and Peace, Alice in Wonderland, and Lorem Ipsum
    a. Goal is that War and Peace and Alice in Wonderland should decrypt whereas Lorem Ipsum (which is made up of latin) should not be cracked.

Action Plan

Development plans is as follows:

1. Investigate and understand the provided python files for the transposition cipher on milliways
2. Practice the transposition cipher by hand to fully understand the process to both encrypt and decrypt
3. Reimplement the transposition cipher from the provided script on milliways
4. Simplify and improve the dictionary detection to output “attempted words, English words, and percentage of actual words over attempted words”
5. If time allows implement frequency analysis instead of plain bruteforcing

Instructions

Carefully read all items to ensure the application functions as intended

Requirements

- Python 3.4 or higher is required on the machine
- PyEnchant library is required (instructions below on how to install)

PyEnchant

1. Execute “python” or “python3” and ensure that python 3.4 or higher is current installed on the computer
2. Using Python 3 (on fedora typically “python3”) execute the following command:
    a. sudo python3 -m pip install pyenchant
    b. Note that you might be required to have a newer version of pip and you can do so by running “sudo python3 -m pip --upgrade pip”
3. Ensure that pyenchant installed correctly

trencode.py

1. trendcode requires 3 arguments and 3 switches
    a. -t <PlainTextFile>
    b. -k <an integer key length typically below the maximum length of the provided plaintext file>
    c. -o <OutputCipherTextResultsFile>
2. Execute the following command using python3:
    a. python3 trencode.py -t plaintext_WarAndPeace.txt -k 50 -o cipher_WarAndPeace.txt
3. Once the command above is executed it will quit the application upon completion. Please keep in mind depending on the size of your file this may take a few minutes to complete.
4. View the output results to confirm the encryption.

trbruteforce.py

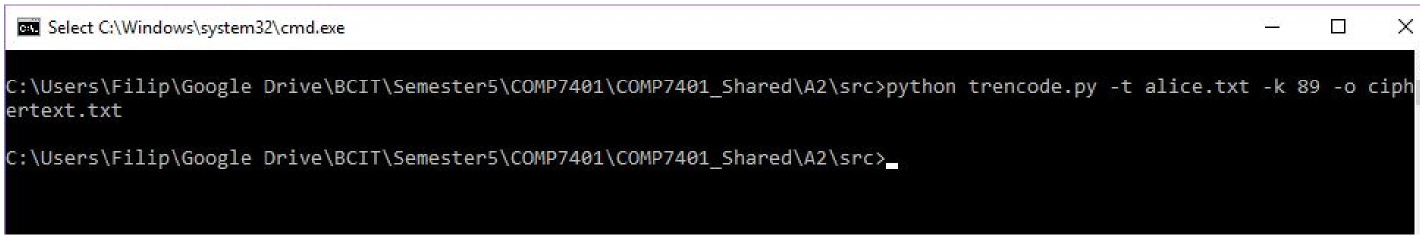
1. trbruteforce requires 2 arguments and 2 switches
    a. -t <CipherFilepath>
    b. -o <outputPath>
2. Execute the following command using python3:
    a. python3 trbruteforce.py -t cipher_WarAndPeace.txt -o result_WarAndPeace.txt
3. Once the command in step 2 is executed you will be prompted to enter a start key length as well as a final key length.
    a. Note: the application provides the maximum possible key length
    b. Start: <an integer between 1 and maximum possible key length>
    c. End: <an integer above the start key length but below the maximum possible keylength>
4. The application will attempt to brute force the ciphertext and will prompt the user once it finds a significant number percentage of english words detected in the ciphertext.
    a. The user can press Y to output the information to the output file indicated in the command line argument
    b. The user can press N to continue bruteforcing and skipping the potential decrypted results

Test Cases

Test Case	Scenario	Expected Behavior	Actual Results	Status
1	Run trencode with large textfile	Will encrypt file using provided key and output to specified output file	File encrypted and output to specified output file	pass
2	Attempt to decrypt large ciphertext from first test case	Program will continue trying until more than 50% english words are found in the potential plaintext	Program will continue trying until more than 50% english words are found in the potential plaintext	pass
3	Results Outputted to file Plaintext if same key used as trencode	Results appear in specified output file Plaintext if same key as trencode	Results appear in specified output file Same key was used as trencode so the result is plaintext	pass

Test case 1

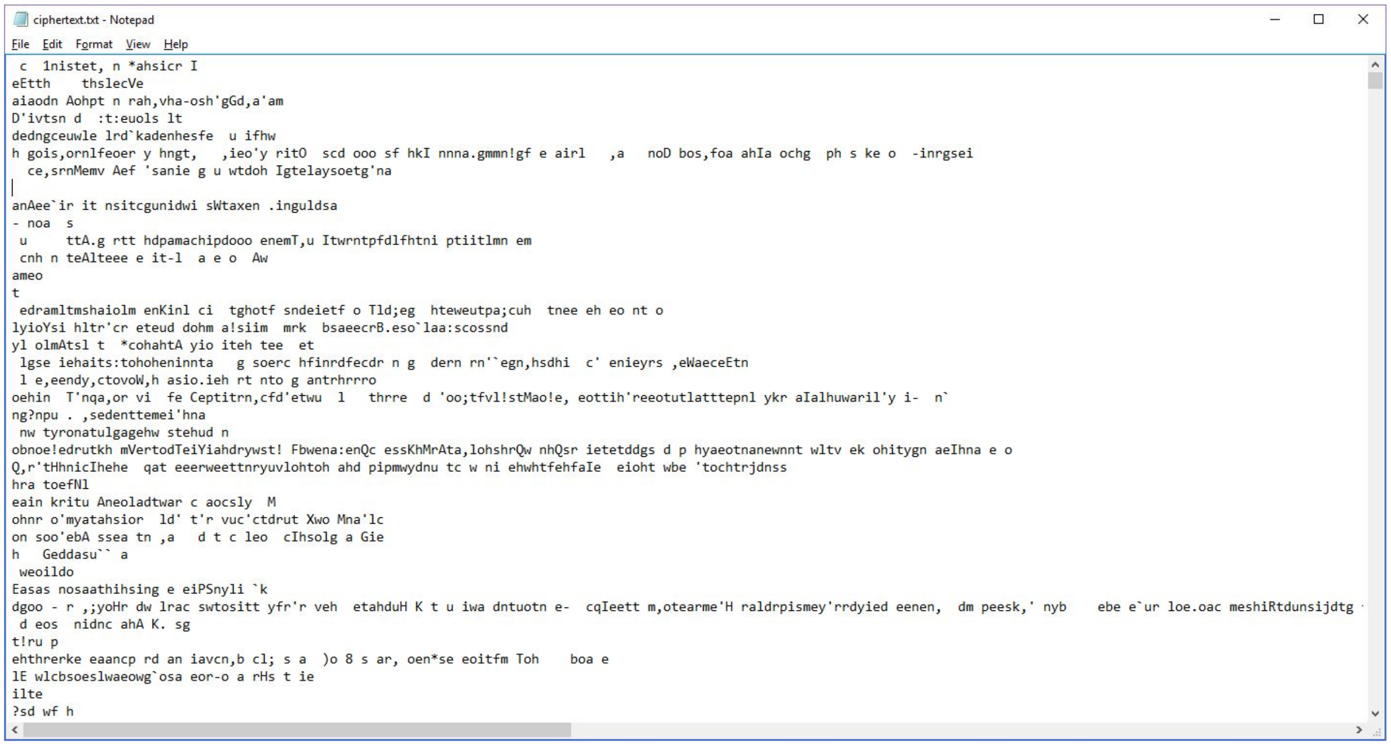
Run trencode.py on alice.txt with key size 89



```
C:\Users\Filip\Google Drive\BCIT\Semester5\COMP7401\COMP7401_Shared\A2\src>python trencode.py -t alice.txt -k 89 -o ciph
ertext.txt

C:\Users\Filip\Google Drive\BCIT\Semester5\COMP7401\COMP7401_Shared\A2\src>
```

Alice.txt after encryption

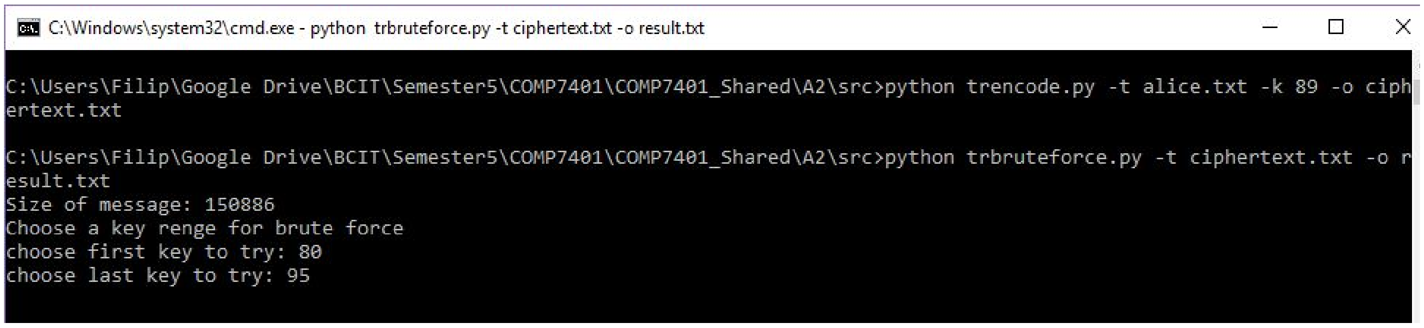


```
 c  1nistet, n *ahsicr I
eEtth    thslecVe
aiaodn Aohpt n rah,vha-osh'gGd,a'am
D'ivtsn d  :t:euols lt
dedngceuwle lrd`kadenhesfe  u ifhw
h gois,ornlfeoer y hngt,   ,ieo'y ritO  scd ooo sf hkI nnna.gmmn!gf e airl   ,a   noD bos,foa ahIa ochg  ph s ke o  -inrgsei
  ce,srnMemv Aef 'sanie g u wtdoh Igtelaysoetg'na

anAee`ir it nsitcgunidwi sWtaxen .inguldsa
- noa  s
 u     ttA.g rtt hdpamachipdooo enemT,u Itwrntpfdlfhtni ptiitlmn em
 cnh n teAlteee e it-l  a e o  Aw
ameo
t
 edramltmshaiolm enKinl ci  tghotf sndeietf o Tld;eg  hteweutpa;cuh  tnee eh eo nt o
lyioYsi hltr'cr eteud dohm a!siim  mrk  bsaeecrB.eso`laa:scossnd
yl olmAtsl t  *cohahtA yio iteh tee  et
 lgse iehaits:tohoheninnta   g soerc hfinrdfecdr n g  dern rn'`egn,hsdhi  c' enieyrs ,eWaeceEtn
 l e,eendy,ctovoW,h asio.ieh rt nto g antrhrrro
oehin  T'nqa,or vi  fe Ceptitrn,cfd'etwu  l   thrre  d 'oo;tfvl!stMao!e, eottih'reeotutlatttepnl ykr aIalhuwaril'y i-  n`
ng?npu . ,sedenttemei'hna
 nw tyronatulgagehw stehud n
obnoe!edrutkh mVertodTeiYiahdrywst! Fbwena:enQc essKhMrAta,lohshrQw nhQsr ietetddgs d p hyaeotnanewnnt wltv ek ohitygn aeIhna e o
Q,r'tHhnicIhehe  qat eeerweettnryuvlohtoh ahd pipmwydnu tc w ni ehwhtfehfaIe  eioht wbe 'tochtrjdnss
hra toefNl
eain kritu Aneoladtwar c aocsly  M
ohnr o'myatahsior  ld' t'r vuc'ctdrut Xwo Mna'lc
on soo'ebA ssea tn ,a   d t c leo  cIhsolg a Gie
h   Geddasu`` a
 weoildo
Easas nosaathihsing e eiPSnyli `k
dgoo - r ,;yoHr dw lrac swtositt yfr'r veh  etahduH K t u iwa dntuotn e-  cqIeett m,otearme'H raldrpismey'rrdyied eenen,  dm peesk,' nyb    ebe e`ur loe.oac meshiRtdunsijdtg
 d eos  nidnc ahA K. sg
t!ru p
ehthrerke eaancp rd an iavcn,b cl; s a  )o 8 s ar, oen*se eoitfm Toh    boa e
lE wlcbsoeslwaeowg`osa eor-o a rHs t ie
ilte
?sd wf h
```

Test case 2

Attempt to decrypt large ciphertext from text case 1.

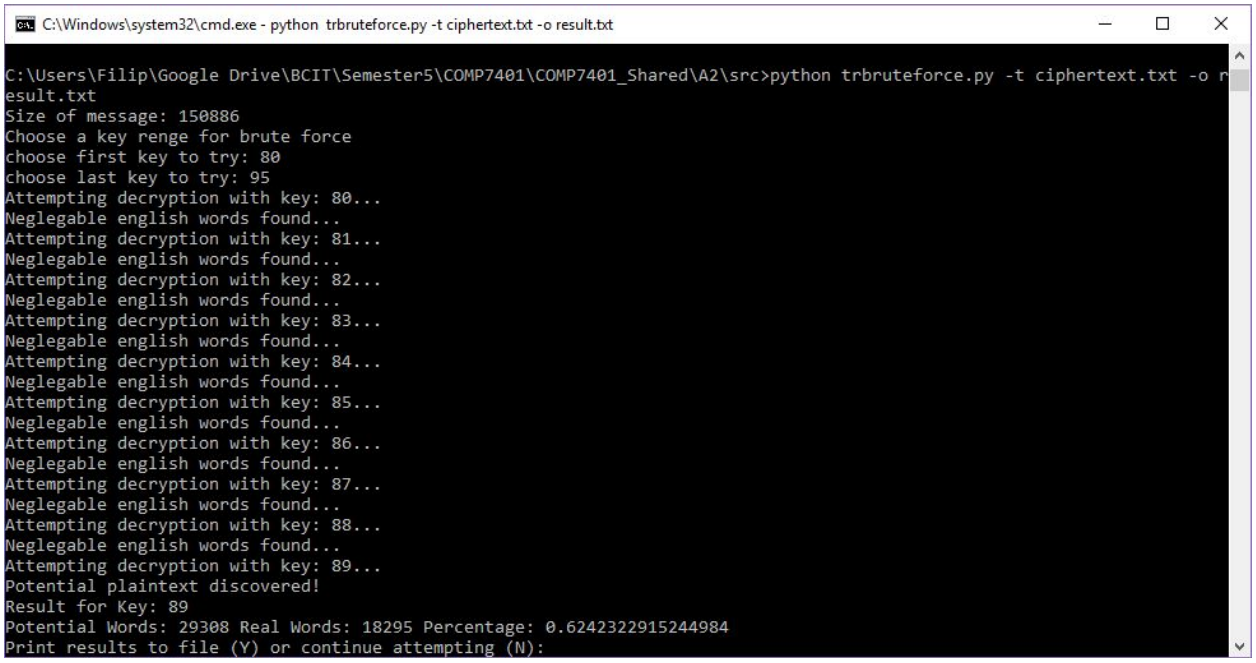


```
C:\Windows\system32\cmd.exe - python  trbruteforce.py -t ciphertext.txt -o result.txt

C:\Users\Filip\Google Drive\BCIT\Semester5\COMP7401\COMP7401_Shared\A2\src>python trencode.py -t alice.txt -k 89 -o ciph
ertext.txt

C:\Users\Filip\Google Drive\BCIT\Semester5\COMP7401\COMP7401_Shared\A2\src>python trbruteforce.py -t ciphertext.txt -o r
esult.txt
Size of message: 150886
Choose a key renge for brute force
choose first key to try: 80
choose last key to try: 95
```

Program will continue trying to decrypt the file, with each key specified in the range until more than 50% english word are found in the resulting output.

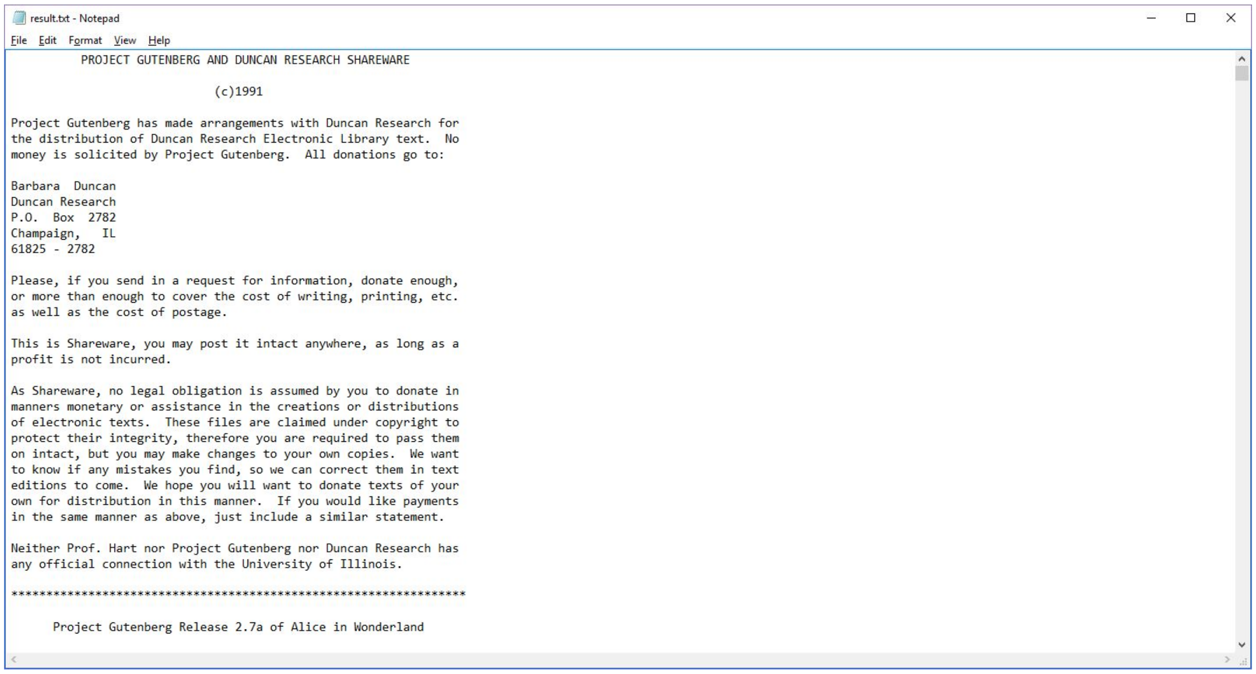


```
C:\Windows\system32\cmd.exe - python  trbruteforce.py -t ciphertext.txt -o result.txt

C:\Users\Filip\Google Drive\BCIT\Semester5\COMP7401\COMP7401_Shared\A2\src>python trbruteforce.py -t ciphertext.txt -o r
esult.txt
Size of message: 150886
Choose a key renge for brute force
choose first key to try: 80
choose last key to try: 95
Attempting decryption with key: 80...
Neglegable english words found...
Attempting decryption with key: 81...
Neglegable english words found...
Attempting decryption with key: 82...
Neglegable english words found...
Attempting decryption with key: 83...
Neglegable english words found...
Attempting decryption with key: 84...
Neglegable english words found...
Attempting decryption with key: 85...
Neglegable english words found...
Attempting decryption with key: 86...
Neglegable english words found...
Attempting decryption with key: 87...
Neglegable english words found...
Attempting decryption with key: 88...
Neglegable english words found...
Attempting decryption with key: 89...
Potential plaintext discovered!
Result for Key: 89
Potential Words: 29308 Real Words: 18295 Percentage: 0.6242322915244984
Print results to file (Y) or continue attempting (N):
```

Test case 3

Results are outputted to specified output file, and as we can see in the previous screenshots this is the result from key size 89 which was used to encrypt the file. The result is plaintext.



```
          PROJECT GUTENBERG AND DUNCAN RESEARCH SHAREWARE

                           (c)1991

Project Gutenberg has made arrangements with Duncan Research for
the distribution of Duncan Research Electronic Library text.  No
money is solicited by Project Gutenberg.  All donations go to:

Barbara  Duncan
Duncan Research
P.O.  Box  2782
Champaign,   IL
61825 - 2782

Please, if you send in a request for information, donate enough,
or more than enough to cover the cost of writing, printing, etc.
as well as the cost of postage.

This is Shareware, you may post it intact anywhere, as long as a
profit is not incurred.

As Shareware, no legal obligation is assumed by you to donate in
manners monetary or assistance in the creations or distributions
of electronic texts.  These files are claimed under copyright to
protect their integrity, therefore you are required to pass them
on intact, but you may make changes to your own copies.  We want
to know if any mistakes you find, so we can correct them in text
editions to come.  We hope you will want to donate texts of your
own for distribution in this manner.  If you would like payments
in the same manner as above, just include a similar statement.

Neither Prof. Hart nor Project Gutenberg nor Duncan Research has
any official connection with the University of Illinois.

*****************************************************************

      Project Gutenberg Release 2.7a of Alice in Wonderland
```

Summary

The Transposition may have been very strong and useful back when it was first started however with modern day computing with a little bit of optimization such as multithreading or multiprocessing it’s very easy to defeat within a minute.

Learning to produce ciphertext and crack it provides a bit of insight in the minds of the mathematical geniuses that had developed them during war and worked on overcoming their enemies in the heat of the pressure.