Auto TTP Breaker report

This is a python program that break two times pad automatically with n-gram language model

Requirements

- Python 3.7
- dill 0.3.1.1
- Pattern 3.6
- blist 1.3.6
- nltk 3.5

How to use

```
pip install requirements.txt
```

or

```
pip3 install requirements.txt
```

modify `decrypt.py` accordingly

```
#   For test
p0 = 'After two and a half years with her parents'
p1 = 'The term Internet when used to refer to the'
LENGTH_OF_BATCH = LENGTH_OF_FILE = p0.__len__()
for i in range(LENGTH_OF_FILE):
    x.append(ord(p0[i]) ^ ord(p1[i]))

# For real use
c0 = open("ctext0", 'rb').read(LENGTH_OF_FILE)
c1 = open("ctext1", 'rb').read(LENGTH_OF_FILE)
for i in range(LENGTH_OF_FILE):
    x.append(c0[i] ^ c1[i])
```

set the length of plaintext you want to get

```
LENGTH_OF_BATCH = 43  # If you want to decrypt in batch, set it different
LENGTH_OF_FILE = 43
```

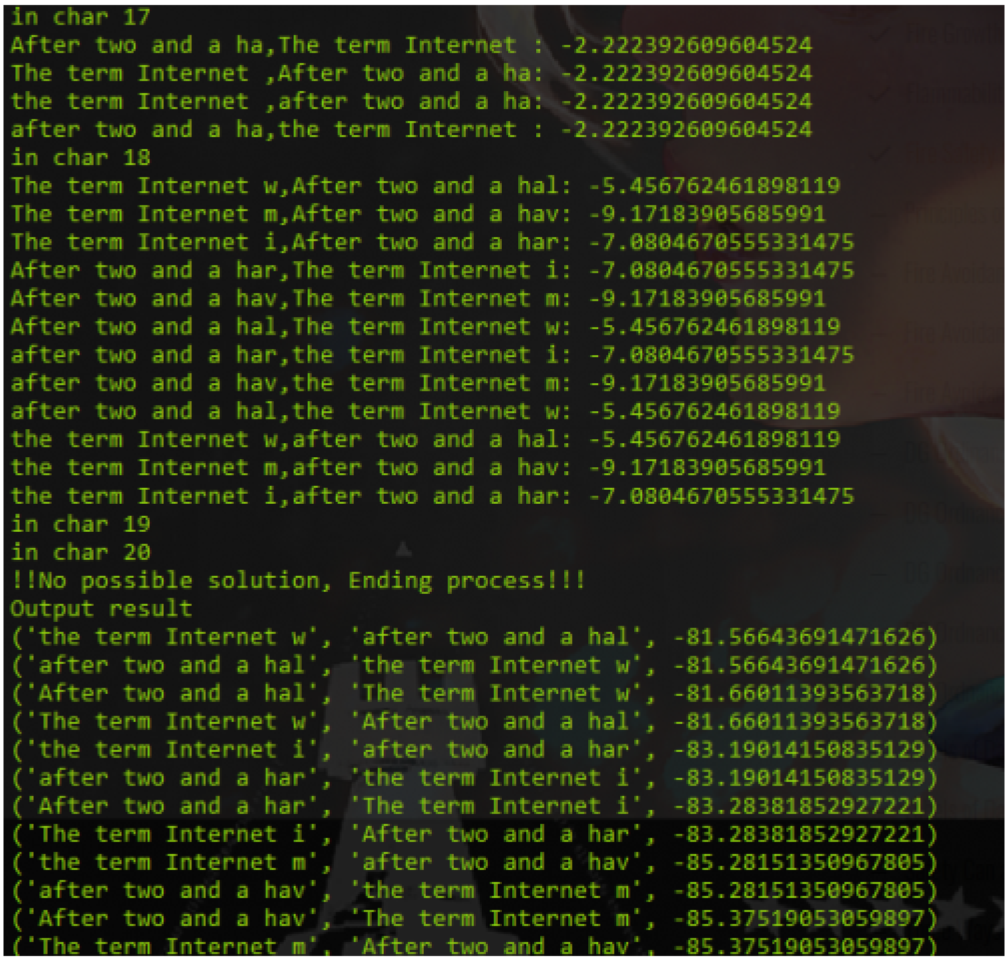
then run

```
py decrypt.py
```

or

```
python3 decrypt.py
```

Expected output



```
in char 17
After two and a ha,The term Internet : -2.222392609604524
The term Internet ,After two and a ha: -2.222392609604524
the term Internet ,after two and a ha: -2.222392609604524
after two and a ha,the term Internet : -2.222392609604524
in char 18
The term Internet w,After two and a hal: -5.456762461898119
The term Internet m,After two and a hav: -9.17183905685991
The term Internet i,After two and a har: -7.0804670555331475
After two and a har,The term Internet i: -7.0804670555331475
After two and a hav,The term Internet m: -9.17183905685991
After two and a hal,The term Internet w: -5.456762461898119
after two and a har,the term Internet i: -7.0804670555331475
after two and a hav,the term Internet m: -9.17183905685991
after two and a hal,the term Internet w: -5.456762461898119
the term Internet w,after two and a hal: -5.456762461898119
the term Internet m,after two and a hav: -9.17183905685991
the term Internet i,after two and a har: -7.0804670555331475
in char 19
in char 20
!!No possible solution, Ending process!!!
Output result
('the term Internet w', 'after two and a hal', -81.56643691471626)
('after two and a hal', 'the term Internet w', -81.56643691471626)
('After two and a hal', 'The term Internet w', -81.66011393563718)
('The term Internet w', 'After two and a hal', -81.66011393563718)
('the term Internet i', 'after two and a har', -83.19014150835129)
('after two and a har', 'the term Internet i', -83.19014150835129)
('After two and a har', 'The term Internet i', -83.28381852927221)
('The term Internet i', 'After two and a har', -83.28381852927221)
('the term Internet m', 'after two and a hav', -85.28151350967805)
('after two and a hav', 'the term Internet m', -85.28151350967805)
('After two and a hav', 'The term Internet m', -85.37519053059897)
('The term Internet m', 'After two and a hav', -85.37519053059897)
```

Description

- Input: 2 string(With only number and ASCII characters) xored with same key
- Output: most possible plain texts

`dataScrapper.py`

This program first collect title of "Featured articles" (high quality articles) from Wikipedia. Then, it visit and download content of each article. A data file name `wikiData.txt` will be generated.

For this project, I collected 2000 articles(about 53.0 MB text file)

`training.py`

This program train an n-gram model using functions from NLTK package. The model is using Witten-Bell back-off Smoothing, which may be helpful dealing with Overfitting of the model.

An n-gram model is basically a model that can predicting the probability of a word or character appears according to the previous n word or character

First, it generate a set of "vocab" which is the characters that needed to be consider by the model when fitting. Any character that is not in the vocab list will not be put into the model and will output meaningless probability if asked to predict. The model will be save from time to time, currently is to every 2000 line. After training, a character level n-gram language model will be outputted using dill package which is a better way than native python pickle to output object with lambda functions.

`test()` is provided to test the model with input you want.

`HelperClasses.py`

This file contains major part of the project's data structure.

To set the threshold of dropping branches, simply modify this

```
POSSIBLE_BRANCH_THRESHOLD = -10
```

The number is log value. -10 is tested to be reasonable value, value too small will result in huge amount of possible path thus lengthen the compute time to very long(very, very long),and use large amount of RAM and disk space (Virtual RAM in Windows)

`class Layer`

The object of this class will store the `Node`s, with same length of decrypted plain text.

`class Node`

The object of this class will represent a possible plain texts and its possible next stages in `Edge`

```
generateNodes(self, nextLayer: Layer, cypherChar, model: WittenBellInterpolated)
```

This function will generate all possible n+1 length plaintext base on this node. It enumerate all possible character and drop 70% of possible branches(leave at least 5)

```
addNodeandEdge(self, a, b, nextLayer, model)
```

This function will calculate the possibility of each given combination of valid next characters and add if above threshold.

Notice that character after space is treated as single character with no context. Becuse the character after space should have little relationshiop with the previous word. Also, in one version of the model, the probability of next character is unreasonable if don't do so.

```
if score > POSSIBLE_BRANCH_THRESHOLD or (self.p.__len__() < 3 and score > -20):
```

If plaintext that is short, the score will be low due to lack of context, so I widen the acceptable range to avoid over cutting of possible results. -20 is tested to be a reasonable boundary.

`class Edge`

`Edge`s connects `Node`s. It store the two new char that may be the next char in both plain text. When deleted, it will also delete the two `Node`s it connected.

`decrypt.py`

This file is the file that directly used by the user as described above. After enumerating all all possible result or end earlier for no possible next character, it will output the top 10 most likely results.

Limitation

- For some unknown reason, the process may end early in some situation. After testing, I found that the direct cause is that the score of all result drop below threshold. But I have no idea what the root cause is and how to solve this.
- The model have difficulty dealing with symbols, so the program do not deal with symbols. I have tried several ways like treat symbols as space, retraining the model, train another word level n-gram model. However, none of these works. I suspect that symbol is more a syntax and grammar related element, thus n-gram model is not good at dealing with it.

Reference

Major sources of my idea and implementation

Mason, J., Watkins, K., Eisner, J., & Stubblefield, A. (2006, October). A natural language approach to automated cryptanalysis of two-time pads. In Proceedings of the 13th ACM conference on Computer and communications security (pp. 235-244).

https://www.cs.jhu.edu/~jason/papers/mason+al.ccs06.pdf

N-gram Language Model with NLTK

https://www.kaggle.com/alvations/n-gram-language-model-with-nltk