If you're bored...

Status: To be marked
Your identity is visible during marking.

Cracking substitution ciphers

This week, we have a debugging challenge, as a pastime while you all wait for the final assignment to be released. It is your task to debug the Python code linked below. I will discuss the result in class next week. Note that this is completely **at your leisure**. It is not part of the course requirements.

The code is supposed to implement a cracking function for a so called substition cipher. That is, the code tries to decrypt messages that have been encrypted by substituting every letter from the alphabet used for the message by another letter from the alphabet, but without having direct access to the decryption key. However, the code is also buggy. It contains a total of 7 locations in which a syntactic error or a runtime error has been made. The challenge is to make the code run to completion by finding and fixing all of the bugs.

In principle, you do not have to understand the implemented algorithm to do this, but below follows some background information for the interested. For general information about subsitution ciphers, you can read more about them at wikipedia.

The algorithm implemented is described in Chen & Rosenthal, 2012, 'Decrypting classical cipher text using Markov chain Monte Carlo'. Assume we have an encrypted message and that we know what language the message is written in. We then proceed by trying out a random mapping of the characters as our decryption key. The result is given a score using a so called *language model*. That is, we try to quantify how good we think the resulting string is as a string of the language we know the message to be in. For instance, if the language is English, we know that *gqoie* not so good as a string, because no (parts of) English words look like that. But on the other hand, even though *egrob* is not an English word, it is a lot better, since it follows the English patterns of where vowels can show up, which consonants can follow eachother, etc. We then try small random changes to the decryption key, and measure whether the the text improves according to the language model or not. By (roughly said) following decryption keys that lead to increasingly better texts, the hope is that we move in the direction of the actual, original message.

The code in mcmc_cipher_cracker.py contains the Python programme you need to debug. In the same directory, you need to put the file lm_dumas_n2.pickle, which contains the statistics needed for the English language model. Start by trying to run the cipher cracker, and look at the error messages

Python raises. When you find an error you cannot understand by simply looking at the error message, considering the assumptions that a bit of code makes, and whether they are actually met by the implementation around it. Ask yourself questions like: Does the object involved have the right type to be used in this way? Does the return value as defined inside the function match the way the function is used when we call it? Do things we assume exist actually exist (are variable names defined, are keys present in the dictionary, are sequences long enough to allow asking for a certain index, do the files we try to open exist under the name we use?)

An effective way of getting answers to such questions is by sprinkling relevant print statements throughout the code when debugging. So for instance, if you get an `IndexError: list index out of range` from some loop, it is helpful to know how long the list in question is, and what index you are trying to get from the list. A call like `print(len(lst),i)`, if the list is `lst` and the index is `i`, will let you track these values through the loop and see where it goes wrong.

When the code runs as it should, the program prints intermediate results of the cracking effort to the screen, and end with its final guess of the decrypted original message after 2000 iterations.

Have fun!

Hemanth Kumar Battula , 5 Oct 2018 15:57

File name: mcmc_cipher_cracker.py (4,6 KB)

Status set to: To be marked

Comment: Debugged and modified all the changes. uploaded the working file.

Submit assignment

Comment

Spell-checking

No file selected.

Select file

Maximum file size: 1024 MB

Submit