Math 4500/6500 Final Project – 2017

Working for a certain government agency, you have intercepted an encrypted message from a mysterious internet newsgroup (we call this the "cyphertext"). Either they are a bunch of harmless fans of 19th century fiction or a dangerous terrorist cell. Only you can decide. In either case, they are cryptographic morons, as it has been determined that they have used a simple substitution cypher to encrypt their message. In this sort of code, each letter of the alphabet is replaced by another letter, so for instance every "A" in the message might be replaced by an "L" (and so forth). The purpose of this homework assignment is for you to use simulated annealing to automatically decrypt the message, recovering the original English text "plaintext". The cyphertext is:

```
RTQIJQRHNZAQFGJAQFBIBQKRGQXJBFBRRTHDJKHSQRGQXBKWFG
ZJJPBAJKRBHKRVGBIGHKJAQMIQEEEJKWFGLZJQPFGQKPFGBIDK
JRRQKPBRQEVQMRPJSBKQLEJLMZJSJZJKIJFHFGZJJTEQKJRJQI
GQFZBWGFQKWEJRFHFGJHFGJZRLNFRHAJTGBEHRHTGBIQETJHTE
JGQXJLJJKQRDBKWVGMFGZJJPBAJKRBHKRTQZFBINEQZEMVGMKH
FQKHFGJZPBZJIFBHKQFZBWGFQKWEJRFHFGJHFGJZFGZJJQKPGQ
XJJXJKFZBJPFHIHKRFZNIFQSHNZPBAJKRBHKWJHAJFZMTZHSJR
RHZRBAHKKJVIHALVQRJYTHNKPBKWFGBRFHFGJKJVMHZDAQFGJA
QFBIQERHIBJFMHKEMQAHKFGHZRHQWHMHNDKHVGHVHKQSEQFRNZ
SQIJVGBIGGQRHKEMFVHPBAJKRBHKRVJIQKZJTZJRJKFQSBWNZJ
HSQFGZJJPBAJKRBHKQERHEBPQKPRBABEQZEMFGJMFGBKDFGQFL
MAHPJERHSFGZJJPBAJKRBHKRFGJMIHNEPZJTZJRJKFHKJHSSHN
ZBSFGJMIHNEPAQRFJZFGJTJZRTJIFBXJHSFGJFGBKWRJJBFGBK
DRHANZANZJPFGJTZHXBKIBQEAQMHZQKPDKBFFBKWGBRLZHVRGJ
EQTRJPBKFHQKBKFZHRTJIFBXJRFQFJGBREBTRAHXBKWQRHKJVG
HZJTJQFRAMRFBIVHZPRMJRBFGBKDBRJJBFKHVGJRQBPQSFJZRH
AJFBAJLZBWGFJKBKWBKQONBFJFZQKRBFHZMAQKKJZVJEEBPHKH
FABKPFJEEBKWMHNBGQXJLJJKQFVHZDNTHKFGBRWJHAJFZMHSSH
NZPBAJKRBHKRSHZRHAJFBAJRHAJHSAMZJRNEFRQZJINZBHNRSH
ZBKRFQKIJGJZJBRQTHZFZQBFHSQAQKQFJBWGFMJQZRHEPQKHFG
JZQFSBSFJJKQKHFGJZQFRJXJKFJJKQKHFGJZQFFVJKFMFGZJJQ
KPRHHKQEEFGJRJQZJJXBPJKFEMRJIFBHKRQRBFVJZJFGZJJPBA
JKRBHKQEZJTZJRJKFQFBHKRHSGBRSHNZPBAJKRBHKJPLJBKWVG
BIGBRQSBYJPQKPNKQEFJZQLEJFGBKWRIBJKFBSBITJHTEJTZHI
JJPJPFGJFBAJFZQXJEEJZQSFJZFGJTQNRJZJONBZJPSHZFGJTZ
HTJZQRRBABEQFBHKHSFGBRDKHVXJZMVJEEFGQFFBAJBRHKEMQD
BKPHSRTQIJGJZJBRQTHTNEQZRIBJKFBSBIPBQWZQAQVJQFGJZZ
JIHZPFGBREBKJBFZQIJVBFGAMSBKWJZRGHVRFGJAHXJAJKFHSF
GJLQZHAJFJZMJRFJZPQMBFVQRRHGBWGMJRFJZPQMKBWGFBFSJE
EFGJKFGBRAHZKBKWBFZHRJQWQBKQKPRHWJKFEMNTVQZPFHGJZJ
RNZJEMFGJAJZINZMPBPKHFFZQIJFGBREBKJBKQKMHSFGJPBAJK
RBHKRHSRTQIJWJKJZQEEMZJIHWKBCJPLNFIJZFQBKEMBFFZQIJ
PRNIGQEBKJQKPFGQFEBKJFGJZJSHZJVJANRFIHKIENPJVQRQEH
KWFGJFBAJPBAJKRBHK
```

Here is the basic idea. We know that some permutation σ_* of 26 letters will convert the cyphertext to the plaintext. We need to find σ_*. Let's think about this as an optimization problem which we'll solve by simulated annealing:

- State space Ω: permutations σ of 26 letters (size $26! \sim 4 \times 10^{26}$ = really big).
- Moves: Transpositions (which swap 2 letters)
- Energy function: E – measures how close we are to a successful decryption.

There are many possible energy functions which could measure how "Englishlike" a string of letters are, but we'll focus on a very simple one: the relative probabilities of letters and two-letter combinations in English text. Here is a table of letter probabilities in English compiled from about 4.5 billion characters of English text: we'll call this probability distribution Q^{letters}:

E	0.120965
T	0.0893813
A	0.0855169
O	0.0746727
I	0.0732512
N	0.0717218
S	0.067282
R	0.0633271
H	0.0495571
L	0.0420646
D	0.0387118
C	0.0316444
U	0.0268158
M	0.0252632
F	0.0218151
G	0.0208634
P	0.0206617
W	0.0182536
Y	0.0172136
B	0.016048
V	0.0105935
K	0.00808698
J	0.00219779
X	0.0019135
Z	0.00113756
Q	0.00104025

A table of counts for 2 letter sequences in a lot of English text is found on the course page (`english_bigrams.txt`). That file describes a probability distribution on the $26^2 = 676$ two-letter combinations which we'll call Q^{digraphs}. Note that $Q^{\text{digraphs}}(i) \neq 0$ regardless of the digraph

i because we're ignoring spaces. So even "xp" might occur in a sentence referring to "anthra**x** **p**roduction"[1].

In any permutation σ of our cyphertext, each letter occurs with a certain frequency: for instance if $\sigma = e$ is the identity permutation (which leaves the letters alone), then J occurs about 13.363% of the time. Together, these frequencies give us empirical distributions of letter probabilities in the proposed decryption $\sigma(\text{cyphertext})$, which we call $P^{\text{letters}}(\sigma)$ and $P^{\text{digraphs}}(\sigma)$

We want to score σ as "close" to σ_* if $P^{\text{letters}}(\sigma)$ is close to Q^{letters} and $P^{\text{digraphs}}(\sigma)$ is close to Q^{digraphs}. But how do we measure "close" for probability distributions?

At this point, I implore you to take MATH 4600 (Probability)! You're welcome to experiment with different measurements for this (you can start reading by looking up "hypothesis testing"). But one good option is to measure the difference between distributions using the "Kullback-Leibler divergence" or "relative entropy".

$$\text{KL}(P, Q) = -\sum_i P(i) \log \frac{Q(i)}{P(i)}$$

Here Q is always the reference distribution for English text and $P(\sigma)$ is the distribution in the permuted cyphertext. If $P(i) = 0$ (that is, some letter or digraph doesn't occur in $\sigma(\text{cyphertext})$, the contribution of that term is 0. (Since every letter and digraph occurs eventually in enough English text, $Q(i)$ is never 0.)

If you go this direction, the final score function should look like:

$$E(\sigma) = \text{KL}(P^{\text{letters}}(\sigma), Q^{\text{letters}}) + \text{KL}(P^{\text{digraphs}}(\sigma), Q^{\text{digraphs}})$$

Your job is to decrypt the text. You'll have to insert spaces and punctuation manually since you just get a long string of letters out. Can you identify it or tell anything about the authors?

1. Implementation notes

I did the project in a few hours one afternoon; here are some notes from my code which might save you some time.

- Mathematica has a built-in "CharacterCounts" function which computes the frequency distribution of letters or digraphs in a text. It returns an Association, so you might want to read (again) about Associations.
- You don't need to run CharacterCounts every time you swap letters as you try random moves; the counts are always going to be the same, only their labels are changing. So you really only need to permute the letters which label the various probabilities in P^{letters} and P^{digraphs}.
- When I ran the code, I needed about 60,000 annealing steps at temperatures ranging from 100 to 0.001. I saw scores as high as 6 or so, and got a final score around 0.22.

[1]Or experience points!

2. Challenge + Extra Credit

There's a completely different methodology which can be used here, which is based on the assumption that the text contains valid English words. Mathematica has a DictionaryLookup function which enables you to find all the words which match a given pattern; in principle, this should let you make some initial guesses and very quickly either expand them to a full decryption or rule your initial assumptions out.

3. Acknowledgements

Thanks to the 2017 class for helping fine-tune this, including Biyao Liang and Daniel Harper, who caught that my (original) definition for Kullback-Leibler divergence had the wrong sign!