Quadgram Statistics as a Fitness Measure
(http://practicalcryptography.com/cryptanalysis/text-characterisation/quadgrams/)

When trying to break ciphers, it is often useful to try deciphering with many different keys, then look at the deciphered text. If the text looks very similar to English (or another language), we consider the key to be a good one.
What we need is a way of determining if a piece of text is very similar to English. This is achieved by counting 'quadgrams' (also known as 'tetragraphs'), or groups of 4 letters e.g. the quadgrams in the text ATTACK are: ATTA, TTAC, and TACK.
Note that single letter frequencies, bigrams and trigrams can also be used for this purpose. My experience shows that quadgram frequencies work slightly better than trigrams, trigrams work slightly better than bigrams etc. but that going higher than 4 letters does not really add any benefit.
To use quadgrams to determine how similar text is to English, we first need to know which quadgrams occur in English. To do this we take a large piece of text, for example several books worth of text, and count each of the quadgrams that occur in them. We then divide these counts by the total number of quadgrams encountered to find the probability of each. If we count the quadgrams in Tolstoy's "War and Peace", we have roughly 2,500,000 total quadgrams after all spaces and punctuation are removed. The probability of a specific quadgram is calculated by dividing the count of that quadgram by the total number of quadgrams in our training corpus. The log probability is simply the logarithm of this number.
A few specific counts:

Quadgram	**Count**	**Log Probability**
AAAA	1	-6.40018764963
QKPC	0	-9.40018764963
YOUR	1132	-3.34634122278
TION	4694	-2.72864456437
ATTA	359	-3.84509320105

The table above shows that some quadgrams occur much more often than other quadgrams, this can be used to determine how similar to English a specific piece of text is. Note that QKPC appears zero times in our training corpus, yet the log probability is not -infinity, this is because we floor all probabilities.
If the text contains QPKC, it is probably not English, but if it contains TION, it is much more likely to be English.
To compute the probability of a piece of text being English, we must extract all the quadgrams, then multiply each of the quadgram probabilities.
For the text ATTACK, the quadgrams are ATTA, TTAC, and TACK. The total probability is

$$p(\mathrm{ATTACK}) = p(\mathrm{ATTA}) \times p(\mathrm{TTAC}) \times p(\mathrm{TACK})$$

where e.g.

$$p(\mathrm{ATTA}) = \frac{\mathrm{count}(\mathrm{ATTA})}{N}$$

In the equation above, count() is the number of times the particular quadgram occured, N is the total number of quadgrams in the training sample.

When multiplying many small probabilities, numerical underflow can occur in floating point numbers. For this reason the logarithm is taken of each probability.

The well known identity log(a*b) = log(a)+log(b) is used, so the final log probability is

$$\log\left(p(\mathrm{ATTACK})\right) = \log\left(p(\mathrm{ATTA})\right) + \log\left(p(\mathrm{TTAC})\right) + \log\left(p(\mathrm{TACK})\right)$$

This log probability is used as the 'fitness' of a piece of text, a higher number means it is more likely to be English, while a lower number means it is less likely to be English.

An Example

In this example we will use the words 'fitness' and 'log probability' interchangeably. This example assumes we have already gathered the log probabilities of all the quadgrams that occur in English text. We now find the likelihood that a new piece of text comes from the same distribution as English text does.

The following text:

ATTACK THE EAST WALL OF THE CASTLE AT DAWN

has a fitness of -129.24. This was achieved by extracting each quadgram, finding the log likelihood of this quadgram in English text (which we calculated earlier), then adding each of the numbers obtained in this way. If we use the Caesar cipher with a key of 5 to encipher this text to

FYYFHP YMJ JFXY BFQQ TK YMJ HFXYQJ FY IFBS

and re-calculate fitness we get -288.10. It is quite clear that the original text has a much higher fitness, which is exactly what we want from a fitness measure. The reason the fitness measure is higher for the first piece of text is that quadgrams like 'EAST' and 'OFTH' are quite common in English, so they contribute a higher score to fitness. The second piece of text contains quadgrams like 'FYYF' and 'BFQQ' which are very rare, these quadgrams contribute a much lower score to the fitness. Since the total fitness is the sum of all these contributions, English like text scores much higher that garbled text.

A Python Implementation

Code for calculating the log probability of a piece of text using quadgram statistics is provided below. Bigram, trigram and quadgram counts are provided in the files bigrams.txt,trigrams.txt and quadgrams.txt respectively. If you are looking for frequencies for languages other than english, see Letter Frequencies for Various Languages.

ngram_score.py is used by other cryptanalysis code provided on this site, for example on the Caesar cipher cracking page. Example use:

```
import ngram_score as ns
fitness = ns.ngram_score('quadgrams.txt')
print fitness.score('HELLOWORLD')
```

- english_monograms.txt
- english_bigrams.txt
- english_trigrams.txt.zip
- english_quadgrams.txt.zip
- english_quintgrams.txt.zip
- ngram_score.py