Statistics 116 - Fall 2002
Breaking the Code
Lecture # 3 – Dictionary Ciphers

Jonathan Taylor

In this lecture we talked about a slightly different kind of cipher – one that is not so easily parametrized by one or two parameters. We call them *dictionary ciphers* and they begin with a keyword, for instance "SPAGHETTI." Given the key word we remove all repeated letters in the keyword preserving the order of the remaining letters. In this case we would be left with "SPAGHETI". Using the keyword, a simple substitution is formed as follows:

Cipher:	SPAGHETIBCDFJKLMNOQRUVWXYZ
Plain:	ABCDEFGHIJKLMNOPQRSTUVWXYZ

We observed a weakness in this cipher. That is, any letters appearing after the last letter in the alphabet to appear in the keyword remain the same. That is, since T is the last letter in the alphabet to appear in the above keyword, the letters $UVWXYZ$ are unencrypted. Possible improvements suggested were:

a) Start the remaining letters at the one following the last letter of the keyword

Cipher:	SPAGHETIJKLMNOQRUVWXYZBCDF
Plain:	ABCDEFGHIJKLMNOPQRSTUVWXYZ

b) More generally, we can shift the remaining letters with a shift less than or equal to $25 - k$. For instance, if we used "SPAGHETTI" and a shift of 7 we would have

Cipher:	SPAGHETIMNOQRUVWXYZBCDFJKL
Plain:	ABCDEFGHIJKLMNOPQRSTUVWXYZ

1 *Maximum a Posteriori* estimation

We did not have a systematic attack for this cipher, but suggested ways we might choose keywords that could be reasonable guesses at the keyword. For instance, it is fairly common for people to use simple passwords on their computers. If they were to encrypt a message, then they would most likely use a

keyword similar to their choice of passwords. It is plausible that we could build a probability distribution for common passwords. For instance, if we were going to use first names we could compile the most common first names in the United States.

Given a "dictionary" $\mathcal{D}$– a collection of words with a prior probability for each word q_w for each word w in the dictionary. One way to estimate the TRUE key word $\tilde{w}$ is to look at the posterior probability that the keyword is w given the ciphertext C, i.e.

$$q_{w|C} = \frac{L(C|w)q_w}{\sum_{z\in\mathcal{D}} L(C|z)q_z}.$$

If our priors are appropriately chosen, words with high posterior probability are more likely to be the keyword than those with lower posterior probability. One systematic way to estimate $\tilde{w}$ is to use the word with the highest posterior probability.

(*Maximum a Posteriori Estimation*) Given a prior probability q_w for choice of keyword w, we estimate

$$\widehat{w} = \text{argmax}_w q_{w|C} = \text{argmax}_w \frac{L(C|w)q_w}{\sum_{z\in\mathcal{D}} L(C|z)q_z}.$$

The estimate $\widehat{w}$ is called the $MAP(Maximum\ a\ Posteriori)$ estimate of $\widehat{w}$.

2 Exploring the posterior probability $q_{w|C}$

Another thing we could do is to explore the posterior probabilities $q_{w|C}$. That is, try sampling words from our dictionary $\mathcal{D}$ with probability of choosing the word w given by $q_{w|C}$. Even if $\widehat{w}$ is not the TRUE keyword $\tilde{w}$, $\tilde{w}$ should have a reasonably high value $q_{\tilde{w}|C}$, so if we sample enough words using the posterior probabilities it should not be too long until we actually sample the key used to encrypt the plain text.

At this point, we come to some of the differences between a Bayesian statistician and a frequentist. In the Bayesian's eyes, the enemy chooses a keyword k at random from $\mathcal{D}$ where the probability of choosing the i-th keyword k_i is q_i – the prior probability. The Bayesian cryptanalyst then takes the ciphertext and computes the posterior probability that the i-th keyword was used to encipher the plain text. This posterior probability says nothing explicitly about the *actual* keyword used by the enemy – it just represents the Bayesian's belief about the probability that a particular keyword was used to create the observed cipher text C.

Another point to be wary of in the assuming the Bayesian viewpoint – the conclusions can depend (sometimes heavily) on the choice of prior probabilities q_i. Often, when enough data, or, in this case, cipher text, is observed the effect

of the prior is "washed out" and the conclusions are very similar to those a frequentist would make. To see this effect in an example, consider the following:

Example You, an investigator for the Nevada Gaming Commission, are sent to Bills Casino in Reno to investigate a claim that the casino is cheating at roulette – that is they are using a special roulette wheel that alters the probability of a ball stopping on green to 4/38 instead of 2/38, while keeping every other position equally likely. From experience, you know that, on average, only 10 % of such allegations are true. To decide whether this claim is true or not, you decide to make 10 bets on black, of which you win 7. What is the posterior probability (after playing these 30 games) that the casino is cheating? Not satisfied, you decide to play 490 more, of which you win 114. What is the posterior probability after the 500 games?

If the casino *is* cheating, then the probability that you win when betting on black is

$$\frac{34}{38}\frac{18}{36} = \frac{17}{38}$$

and the likelihood that you win 7 times out of 30 is

$$\left(\frac{17}{38}\right)^7 \left(\frac{21}{38}\right)^{23} = 4.27 \times 10^{-9}.$$

On the other hand, if they are not cheating, the probability that you win when betting on black is 9/19 and the likelihood that you win 7 times out of 30 is

$$\left(\frac{18}{38}\right)^7 \left(\frac{20}{38}\right)^{23} = 2.07 \times 10^{-9}.$$

The posterior probability that the casino is cheating is therefore

$$\frac{0.05 \times 4.27}{0.05 \times 4.27 + 0.95 \times 2.07} = 0.10.$$

Hence, you now believe there is a 10% chance that the casino is cheating.

A frequentist, on the other hand might look at the 7 wins out of 30 trials and compute a Z-score under the hypothesis that the casino is not cheating. For this data, the Z-score is

$$\frac{7 - \frac{18}{38} \times 30}{\sqrt{30 \times \frac{18}{38} \times \frac{20}{38}}} \simeq -2.64.$$

The frequentist, knows that this Z-score should have approximately a Normal distribution (i.e. the "bell-shaped curve") and the frequentist would tell you that if the casino is not cheating, there is approximately 0.4 % chance that you would win 7 or less bets out of 30. The frequentist would then conclude that because this is so unlikely, the casino *must* be cheating.

If you go back and play the rest of the games, in which the ratio of the number of wins to the number of games played is roughly constant and redo the computation, you see that you now believe that there is a 99.99% chance that the casino is cheating. Hence, given enough data the Bayesian's prior beliefs (that is, giving the casinos the benefit of the doubt) are eventually overtaken by the data.

Another issue arises when there are many keys, the denominator in the expression for the posterior probabilities $q_{i|C}$ can be difficult to compute. This means that it can be difficult to truly sample from the posterior distribution. We will see some tricks to do this when we try to break general substitution ciphers.

3 Example

In this section we go over an example of the use of the dictionary cipher and compare the strategy of choosing a key with using the prior probabilities with an exhaustive search technique. I have taken an online version of the King James Bible and chosen a key at random from all the words of length at least 5 in the bible and encrypted Shakespeare's "Merchant of Venice".

The restriction that the key have length at least 5 omits keys such as "and", "why", etc. Since there are repeated words (of length at least 5) in this version of the Bible, choosing a word at random from these words is like choosing a word w_i with prior probability equal to

$$q_{w_i} = \frac{\#\{\text{times } w_i \text{ appears in King James Bible}\}}{\#\{\text{words of length at least 5 in the King James Bible}\}}.$$

In this version, there were 243531 such words, hence to do an exhaustive search of these words would require 243531 likelihood computations. Another approach to break the code is to sample words with probabilities q_{w_i} repeatedly and report all keywords that have a high likelihood, for instance if the (scaled) log-likelihood is bigger than some number $\tilde{\ell}$. The problem with this strategy is that you have to choose $\tilde{\ell}$ so that you will not be wasting too much of your time (if you choose $\tilde{\ell}$ too small – many, many keywords will have higher likelihood).

Remember that the (scaled) log-likelihood of a given keyword of cipher text is

$$\ell(w; C) = \frac{1}{\#C} \sum_{l=A}^{Z} \#\{l\text{'s in the decryption of } C \text{ using keyword } w\} \times \log p_l.$$

I claim that if we only look at keywords w with likelihood $\ell(w; C) > 3.2$ or so we will only have to actually look at very few keywords to decrypt the message.

Where do I get this "crazy" claim? Well, if the string of cipher text is long enough, then the "law of large numbers" tells us that if the plain text that C

represents is "typical", then at the true keyword w

$$\lim_{\#C \to \infty} \ell(w; C) = \sum_{l=A}^{Z} p_l \log p_l.$$

If we use the typical frequencies of English, we see that this expression is approximately equal to -2.85.

Figure 1 plots, for 500 trials, the proportion of trials in which a given number of sample words or less was needed to break the dictionary cipher. For instance, we see that in about 70% of the trials, it took 2000 sampled words (sampled according to the prior probabilities) or less to break the code. Figure 2 plots, for 500 trials, the proportion of times the scaled log-likelihood $\ell(w; C)$ was above 3.2, for all the trials in which the cipher was broken with under 10000 sampled keywords (this was 436 out of 500 trials). For instance, in more than 90 % of the trials, we would only have to inspect 5 decryptions to see if they were correct or not, that is there would only be 4 keywords that had $\ell(w; C) > 3.2$ before the correct keyword was found.

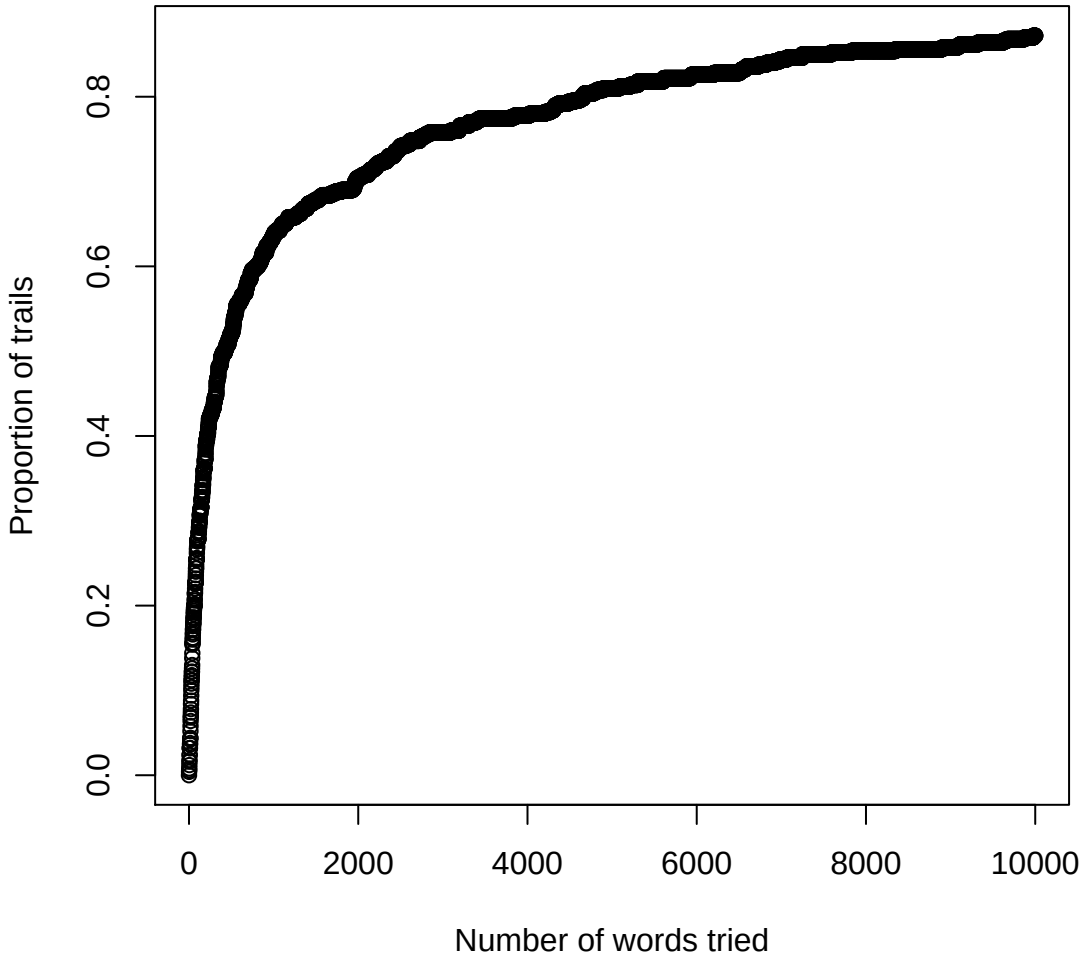

Figure 1: Distribution function of the number of trials needed to solve the dictionary cipher.

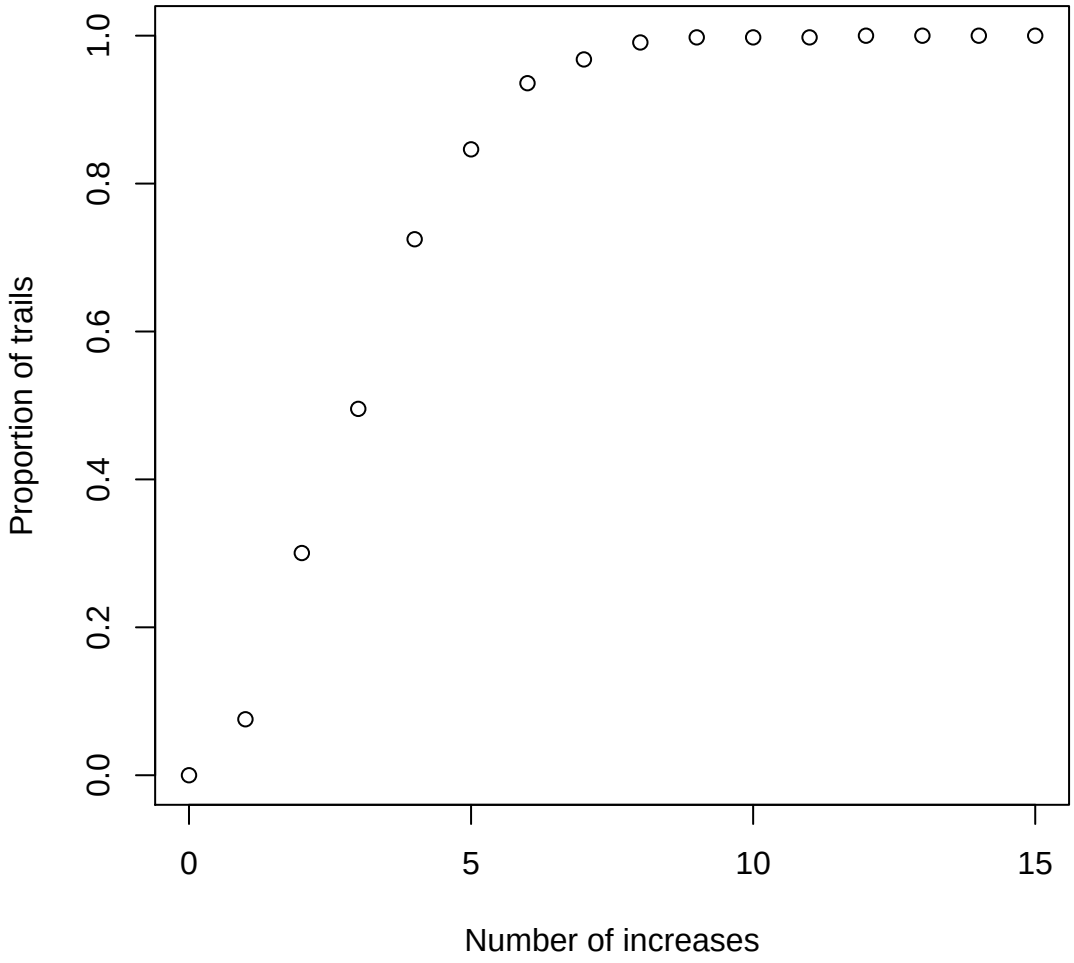

Figure 2: Distribution function of the number of times the (scaled) log-likelihood was above 3.2, when the cipher was broken by under 10000 samples.