Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
Department of Mechanical Engineering

6.050J/2.110J **Information and Entropy** **Spring 2003**

Problem Set 5 Solutions

Solution to Problem 1: The Registrar's Worst Nightmare

Table 5–1 summarizes our knowledge at the beginning of the problem.

Probability	Value	Corresponding Statement
$p(U \mid A)$	0.20	20% of the Add Forms got lost in the mail
$p(E \mid A)$	0.80	[80% of the Add Forms were received properly]
$p(U \mid R)$	0.15	15% of the Registration Forms were illegible because of ink smears
$p(E \mid R)$	0.85	[85% of the Registration Forms were legible]
$p(E \mid N)$	0.02	errors in typing enrolled 2% of the people who didn't want to take the course
$p(U \mid N)$	0.98	[98% of the people who didn't want to take the course were not accidentally enrolled]

Table 5–1: Our Knowledge of the Probabilities

Solution to Problem 1, part a.

Reading from Table 5–1, we know that the probability that Alice is enrolled given that she submitted an Add form is $p(E \mid A) = 0.80$.

Solution to Problem 1, part b.

This part gives us the information that the *probability a student submitted an Add form given that the student is enrolled* is 25%. In probability-speak, this is $p(A \mid E) = 0.25$, the answer to the question.

Solution to Problem 1, part c.

Here we are given the information that $p(E) = 0.05$. From Bayes' theorem we know that $p(A, E) = p(A)p(E \mid A) = p(E)p(A \mid E)$. Since we have the last two numbers, we can calculate our answer as $p(A, E) = 0.05 \times 0.25 = 0.0125$, or 1.25%.

Solution to Problem 1, part d.

From part b. we know that $p(A \mid E) = 0.25$. Thus, from part c. we have the equation:

$$p(A) = \frac{p(E)p(A \mid E)}{p(E \mid A)} \tag{5–1}$$

which is equal to $\frac{0.05 \times 0.25}{0.8} = 0.016$, or 1.6%.

Solution to Problem 1, part e.

As before, $p(A \mid E) = 0.25$, or 25%.

Solution to Problem 1, part f.

We are looking for $p(N)$. We can write the following equations:

$$p(E) = p(E \mid A)p(A) + p(E \mid R)p(R) + p(E \mid N)p(N) \tag{5–2}$$

$$1 = p(A) + p(R) + p(N) \tag{5–3}$$

We can substitute Equation 5–3 into Equation 5–2 to eliminate $p(R)$ and obtain:

$$\begin{aligned} p(N) &= \frac{p(E) - p(E \mid A)p(A) - p(E \mid R) + p(E \mid R)p(A)}{p(E \mid N) - p(E \mid R)} \\ &= \frac{0.05 - 0.80 \times 0.16 - 0.85 + 0.85 \times 0.16}{0.02 - 0.85} \\ &= \frac{0.7992}{0.83} \\ &= 0.963 \end{aligned} \tag{5–4}$$

So the answer is 96.3%.

Solution to Problem 1, part g.

Here the probability is $p(E) = 0.05$, or 5%.

Solution to Problem 1, part h.

Here we are looking for $p(E|N)$, which we've known from the start to be 2%.

Solution to Problem 2: Special Orders Don't Upset Us

Solution to Problem 2, part a.

This is a noisy channel with the same probabilities for mixing up Z and B. Channel capacity is defined as the maximum mutual information (for any possible input probability) times the rate W. The rate of error is $\epsilon = 0.08$. Channel capacity for this channel is given by Equation 6.26.

$$\begin{aligned} C &= M_{max}W \\ &= 1 - \epsilon \log_2\left(\frac{1}{\epsilon}\right) - (1-\epsilon)\log_2\left(\frac{1}{(1-\epsilon)}\right) \\ &= 1 - 0.08 \log_2\left(\frac{1}{0.08}\right) - (0.92)\log_2\left(\frac{1}{0.92}\right) \\ &= 0.5978 \text{ bits/second} \end{aligned} \tag{5–5}$$

(5–6)

Solution to Problem 2, part b.

Information per symbol is defined as

$$\begin{aligned} I &= p\log_2\left(\frac{1}{p}\right) + (1-p)\log_2\left(\frac{1}{1-p}\right) \\ &= 0.1\log_2\left(\frac{1}{0.1}\right) + (0.9)\log_2\left(\frac{1}{0.9}\right) \\ &= 0.1\log_2\left(\frac{1}{0.1}\right) + (0.9)\log_2\left(\frac{1}{0.9}\right) \\ &= 0.469 \text{ bits} \end{aligned} \tag{5–7}$$

(5–8)

Solution to Problem 2, part c.

Since the information per symbol, I, is less than the channel capacity, C, we can send our information reliably.

Solution to Problem 2, part d.

Our maximum rate is the channel capacity (bits/second) divided by the information content per order (bits/order), to give us orders per second of 1.27.

Solution to Problem 2, part e.

Table 5–2 and 5–3 shown the solution to this problem.

Character	Code
Burger-Burger	B
Zucchini-Burger	ZB
Burger-Zucchini	ZZB
Zucchini-Zucchini	ZZZ

Table 5–2: Huffman code for Buzz

Order	Probability	Bits	Average # of Bits
Burger-Burger	0.81	1	0.81
Zucchini-Burger	0.09	2	0.18
Burger-Zucchini	0.09	3	0.27
Zucchini-Zucchini	0.01	3	0.03
Total	1.00		1.29

Table 5–3: Information content of Huffman code for Buzz

The channel capacity is 0.5978 bits/second, or 1.1956 bits every two seconds. This is lower than our needed capacity, so the scheme will not work.

Solution to Problem 3: New, "Improved" Web Protocol

Solution to Problem 3, part a.

The client should resend the request, under the assumption that its request was lost, or the sever did not respond. Also, the client should check that the server is not down, in which case the client should abort.

Solution to Problem 3, part b.

If the client's initial request for an entire page gets lost, then the server will receive nothing from the client, and therefore the server will do nothing. Meanwhile, the client will not know that its request was lost initially, and will wait for the server to begin sending packets (of course, this won't happen). After ten seconds have gone by without the client receiving any packets, it will follow its timeout rule, which tells it to start over again by sending the request again. If it gets through that time,everything will proceed as expected, and if it gets lost again, this process will keep repeating until it gets through. Thus, the protocol handles loss of initial requests properly.

Solution to Problem 3, part c.

All packets received by the client from the server will be correct because of the CRC error detection. Thus, the problems must be because of something other than corrupted packets. Here are a few:

1. Consider what happens if packets are lost – in most cases, this is fine because the missing packets will be requested by the client. However, if the last data packet (before the End of Page packet) gets lost, the client will never know that it was missing. This is because the client figures out which packets are missing based on the greatest numbered packet it has received; if the last packet (which had the greatest number) is dropped, then the client will use an incorrect greatest number and will incorrectly believe it has received the whole page even though the latter part of the page is missing.

2. Consider what happens if the content of a web page is updated before a client successfully receives every packet: the client will receive packets corresponding to new content that may not properly align with old content.

Solution to Problem 3, part d.

Fixes for these issues:

1. Include the total number of packets within each data packet (this may seem a bit wasteful, but perhaps you've seen a similar scheme used in package shipping, where it's common to see boxes labelled "Box 1 of 3," for example). Another fix for might be to make a special "End of Page" Packet, which specifically signifies the end of the transmission. The client should check for this packet, and if it doesn't have it, request all the packets between the highest-numbered packet received and this packet.

2. Include a last-modified-time field in every packet. If the client receives a packet that is "newer" than the other packets received, the client should discard all data and request the page from the beginning, and also discard any subsequently received older packets. This fix may result in slow downloads in the case of dynamic pages, because it would require error-free transmission of all packets of the same request, i.e., retransmissions would never work.