COMP3721 Week Two Lab Synopsis

Notation

- In the context of data communications, the unit modifiers kilo, mega and giga are always interpreted as SI (International System of Units[1]) prefixs:
  - k = 10^3
  - M = 10^6
  - G = 10^9
- We also have the following common abbreviations:
  - (b)it – a logarithmic measure of information – one bit is equivalent to two (linear) possibilities (0 or 1).
  - (B)yte = 8 bits = 1 octet

Logarithms

- Logarithms tend to show up in the context of data communications frequently since we are dealing with bits (and for other reasons to be seen later)
- The logarithm is defined in terms of its relation to exponents

$$x^y = z$$
$$\log_x z = y$$

- Logarithms and exponents are then, by definition, complementary (just as addition and subtraction or multiplication and division are complementary) – one 'undoes' the other

$$\log_x x^y = y$$
$$x^{\log_x y} = y$$

- Bits are simply the logarithmic measure of something, information in the form of possibilities, where the base used is 2
  - For example, a single letter can take on one of 26 possible values. The amount of information in that single letter, measured in bits, is then $\log_2 26 \approx 4.7$ bits.
- We can verify our answer above fairly easily by entering it into our calculator, $2^{4.7} = 25.99$. What is a little more complicated is

1 See http://physics.nist.gov/cuu/Units/prefixes.html

getting the answer in the first place.

- Most scientific calculators will only calculate logarithms for base 10 (generally labeled LOG) and base e (generally labeled LN).
- Then to calculate logarithms for other bases (commonly base 2), we need the identity:

$$\log_x y = \frac{\log_n y}{\log_n x}$$

where n is any base.

- With this identity then, we can solve the base 2 logarithm of 26 as follows:

$$\log_2 26 = \frac{\log_{10} 26}{\log_{10} 2} = \frac{1.414973348}{0.301029996} \approx 4.7$$

Information versus Data

- For the sake of discussing data communications, we will consider the term *data* to refer to the number of bits to be moved from one location to another
- An interesting application of logarithms is in considering how much *information* (in bits) something (a picture, a book, a digitized movie) really has – this is often less than the data we move
- As an example of this contrast, consider that you have to move the the word “example” across a communications channel where the machines on either end use extended ASCII encoding for characters.
- **Approach One:** The data-oriented approach would then say:

  6 extended ASCII characters * 8 bits / character = 48 bits

  - We have 48 bits of data to move in this case.
- **Approach Two:** But it easy to argue that there is less information than this (48 bits) in the word “example”, *if we consider the specific nature of the data being moved*. If we assume we are only moving English words[1], we could put

1 This assumption is an important part of the argument. For example, without it, we could not restrict the possible range of characters in use. Information tends to be lesser than the data moved precisely because we are considering the specific characteristics of a particular set of data.

together an argument as follows:

- The average English word has 5.5 letters. To simplify things, we'll assume 5 letters instead.
- Each letter can take on one of 26 possible values (a-z)
- Then the number of unique words that can be formed is at most $26^5 \approx 11.9$ million. In bits this is

$$\log_2 26^5 \approx 23.5\, bits$$

- But even this seems somewhat contrary to instinct – there are not 11.9 million words in the English language
- The above is still an over-estimate of the real information involved (under the assumption that we are only moving English words) because it assumes each letter is independent of each other letter. Obviously this is not true, for example, our model above allows the word "aaaaa" (as does the extended ASCII, data-oriented model).
- **Approach Three:** We can take an alternate approach then and consider the number of possible words that exist
  - the largest English dictionary contains approximately 250 000 words, though a much smaller vocabulary of 30 000 words is sufficient for most tasks such as reading a newspaper
  - Then a single word, such as "example" is just one of the 250 000 possibilities and, all words being equally probable[1], represents just

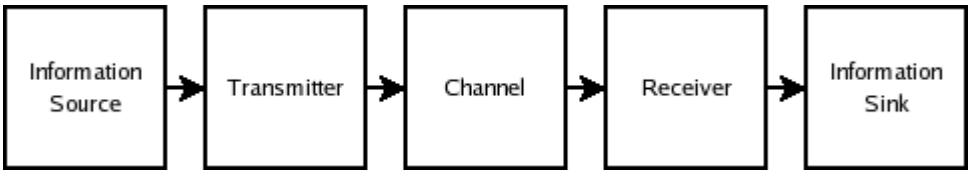
$$\log_2 250000 \approx 18\, bits$$

Where Information Matters – Application One

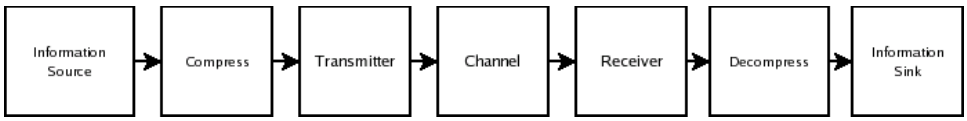
- While the above discussion of information is perhaps a good way to practice logarithms, it also provides an excellent opportunity to consider the simple and extended communication models introduced in lecture:

1 This assumption keeps things simple. In a more realistic example of moving words, for example, the transfer of a book, all words would not be equally probable - "the" is more probable and frequent than "antidisestablishmentarianism". In that case, we can actually come up with even better, lower estimate of the information, based on the probability associated with each word. This approach leads to a measure called *entropy* which is a core concept in the general study of *information theory*.

- The simple communication model is shown as a block diagram here:



- If our information source and sink do encode data using extended ASCII format, then the system above would move approximately 5.5 characters / word * 8 bits / character = 44 bits / word.
- Given that the cost of moving bits through a communications system is still relatively expensive, we have reason to consider whether there is room to improve the efficiency of the system
- In this case, our arguments above regarding the information involved say that we should be able to do better (i.e. 18 bits / word instead of 44 bits / word).
- So we might consider the following, extended communication model instead



- Here we expect the compression mechanism to shrink the data from approximately 44 bits per word on average to something closer to 18 bits per word.

Where Information Matters – Application Two

- The field of information theory is interested with the exact question we have considered here – how much information really exists in a given block of data
- We saw an example where the data consisted of English words – there was less information than data because each letter in a word is not entirely independent of its neighbouring letters. But all manner of other examples exist
  - images – while in theory each pixel could be completely different from its neighbouring pixels, this is not true of most images where neighbouring pixels are normally very similar
  - audio – sound consists of waveforms. While the amplitude of

a wave may be completely unrelated to the amplitude of the waveform just before or after, this is not the common case.

  - movies – not only are the pixels related spatially, as in images, but also temporally. The image in frame one is often very similar to the image in frame two.

- The example sited here all consider the specific nature of the data, but some arguments can be made regarding limits that exist in all information sources
- Claude Shannon, the father of information theory, put together just such a general argument[1] to establish the important result today known as Shannon's Theorem. We will often use this theorem later in the course.

Exercises

1. Many computer architectures in use today have 32-bit words. Computer languages such as C or C++ specify primitive data types such as integers in relation to the word size of a machine. What can we say about the range of integer values on a 32-bit machine when using a language such as C? How is the statement being made related to logarithms?
2. How many bits of information in a single decimal digit? In a hexadecimal digit?
3. Public key cryptography uses large composite numbers as keys – typically 1024 digits or more. How many bits of information are there in a 1024 digit key?
4. You are going to multiply together two 30 digit hexadecimal values. What is the maximum number of hexadecimal digits possible in the result?
5. Estimate the space saved (in bits) by converting a 800x600 24-bit colour image to a gray-scale image with 64 unique shades of gray.

1 This is well outside the scope of this course, but for those interested, see http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html

Solutions to Exercises

1. 32-bits is equivalent to 2^{32} possible values. This relationship (32 bits = 2^{32} possible values) is simply the logarithm (bits are a base 2 logarithmic measure) to exponent relationship. So an integer in C on a 32-bit platform can have a value of between
   - 0 and (2^{32}-1=) 4294967295 if interpreted as unsigned, or
   - -2147483648 and 2147483647 if interpreted as signed
2. A digit can take on one of 10 possible values (0-9), we can determine that a digit contains $\log_2 10 \approx 3.3$ bits of information. A hexadecimal digit can take on one of 16 possible values – $\log_2 16 = 4$ bits.
3. We could try to solve this by calculating $\log_2 10^{1024}$ but most calculators will choke on trying to solve 10^{1024}. A more sophisticated approach is to recognize that a digit is also a logarithmic measure of information – it is simply a base 10 measure as opposed to the bit which is a base 2 measure.

   Using our knowledge from the prior question, we can then calculate that 1024 digits are equivalent to 1024 * 3.321928095 bits/digit ≈ 3402 bits.

   It is also helpful to realize that if we know the number of bits in a digit (3.321928095 bits / 1 digit), then we also know the number of digits in a bit (1 digit / 3.321928095 bits = 0.301029996 digits/bit). This implies another characteristic of logs:

   $$\log_x y = \frac{1}{\log_y x}$$

   Note that then, the calculation we did here is

   $$\log_2 1024 = \frac{\log_{10} 10^{1024}}{\log_{10} 2} = \log_{10} 10^{1024} \times \log_2 10 \approx 1024 \times 3.3 \approx 3402\, bits$$

   which is just an application of our second logarithm identity.
4. One of the useful characteristics of logarithms is the fact that where you would multiply the linear values, you simply need to add the logarithms to get the same answer. You implicitly know this from high school algebra where you would have learned that $x^y * x^z = x^{y+z}$ – with logs we simply deal with the y and z variables and need not bother with the base of x.

Hex digits are just a base 16 logarithmic measure (like decimal digits or bits). So we can think of multiplying two 30 digit hexadecimal values as simply

$16^{30} * 16^{30} = 16^{30+30} = 16^{60}$

and conclude that the result will have at most 60 hexadecimal digits.

5. The original image contains 800 * 600 pixels * 24 bits/pixel = 11 520 000 bits of data. 64 shades of gray are equivalent to $\log_2 64 = 6$ bits/pixel so the converted image will require ¼ the space.