A quick tutorial on generating a huffman tree

Lets say you have a set of numbers and their frequency of use and want to create a huffman encoding for them:

FREQUENCY	VALUE
5	1
7	2
10	3
15	4
20	5
45	6

Creating a huffman tree is simple. Sort this list by frequency and make the two-lowest elements into leaves, creating a parent node with a frequency that is the sum of the two lower element's frequencies:

```
      12:*
      /  \
   5:1   7:2
```

The two elements are removed from the list and the new parent node, with frequency 12, is inserted into the list by frequency. So now the list, sorted by frequency, is:

```
10:3
12:*
15:4
20:5
45:6
```

You then repeat the loop, combining the two lowest elements. This results in:

```
      22:*
      /   \
  10:3   12:*
         /   \
       5:1   7:2
```

and the list is now:

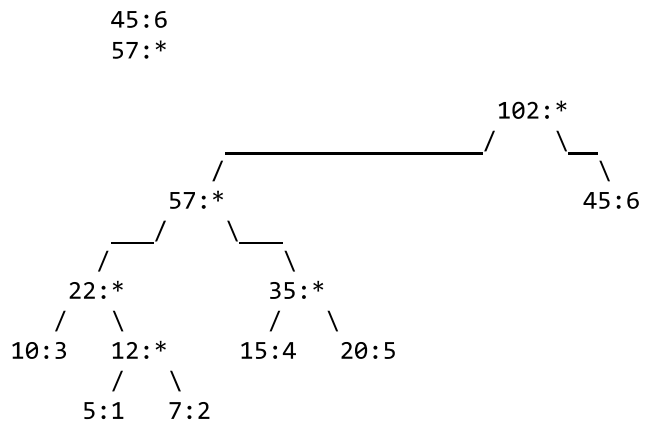
```
15:4
20:5
22:*
45:6
```

You repeat until there is only one element left in the list.

```
      35:*
      /   \
   15:4  20:5

      22:*
      35:*
      45:6

          57:*
       ___/    \___
      /            \
    22:*           35:*
   /   \           /   \
10:3   12:*     15:4   20:5
       /   \
     5:1   7:2
```



Now the list is just one element containing 102:*, you are done.

This element becomes the root of your binary huffman tree. To generate a huffman code you traverse the tree to the value you want, outputing a **0** every time you take a lefthand branch, and a **1** every time you take a righthand branch. (normally you traverse the tree backwards from the code you want and build the binary huffman encoding string backwards as well, since the *first* bit must start from the top).

Example: The encoding for the value **4** (15:4) is **010**. The encoding for the value **6** (45:6) is **1**

Decoding a huffman encoding is just as easy : as you read bits in from your input stream you traverse the tree beginning at the root, taking the left hand path if you read a **0** and the right hand path if you read a **1**. When you hit a leaf, you have found the code.

Generally, any huffman compression scheme also requires the huffman tree to be written out as part of the file, otherwise the reader cannot decode the data. For a static tree, you don't have to do this since the tree is known and fixed.

The easiest way to output the huffman tree itself is to, starting at the root, dump first the left hand side then the right hand side. For each node you output a **0**, for each leaf you output a **1** followed by N bits representing the value. For example, the partial tree in my last example above using 4 bits per value can be represented as follows:

```
000100 fixed 6 bit byte indicates how many bits the value
       for each leaf is stored in.  In this case, 4.
0      root is a node
       left hand side is
10011  a leaf with value 3
       right hand side is
0      another node
       recurse down, left hand side is
10001  a leaf with value 1
       right hand side is
10010  a leaf with value 2
       recursion return
```

So the partial tree can be represented with **000100010011010001100010**, or 23 bits. Not bad!

Letter frequency

From Wikipedia, the free encyclopedia

The **frequency of letters** in text has been studied for use in cryptanalysis, and frequency analysis in particular, dating back to the Iraqi mathematician Al-Kindi (c. 801–873 CE), who formally developed the method (the ciphers breakable by this technique go back at least to the Caesar cipher invented by Julius Caesar, so this method could have been explored in classical times).

Letter frequency analysis gained additional importance in Europe with the development of movable type in 1450 CE, where one must estimate the amount of type required for each letterform, as evidenced by the variations in letter compartment size in typographer's type cases.

California Job Case

Linguists use letter frequency analysis as a rudimentary technique for language identification, where it's particularly effective as an indication of whether an unknown writing system is alphabetic, syllablic, or ideographic. For example, the Japanese Hiragana syllabary contains 46 distinct characters, which is more than most phonetic alphabets, e.g. the Hawaiian alphabet which has a mere 13 letters, or English which has 26.

No exact letter frequency distribution underlies a given language, since all writers write slightly differently. However, most languages have a characteristic distribution which is strongly apparent in longer texts. Even language change as extreme as from old English to modern English (regarded as mutually unintelligible) show strong trends in related letter frequencies: over a small sample of Biblical passages, from most frequent to least frequent, **enaid sorhm tgþlwu (æ)cfy ðbpxz** of old English compares to **eotha sinrd luymw fgcbp kvjqxz** of modern English, with the most extreme differences concerning letterforms not shared.[1]

Linotype machines for the English language assumed the letter order, from most to least common, to be **etaoin shrdlu cmfwyp vbgkjq xz** based on the experience and custom of manual compositors. The equivalent for the French language was **elaoin sdrétu cmfhyp vbgwqj xz**.

Modern International Morse code (generally believed to have been developed by Alfred Vail based on English-language letter frequencies of the 1830s) encodes the most frequent letters with the shortest symbols; arranging the Morse alphabet into groups of letters that require equal amounts of time to transmit, and then sorting these groups in increasing order, yields **e it san hurdm wgvlfbk opxcz jyq**. Similar ideas are used in modern data-compression techniques such as Huffman coding.

Letter frequency was used by other telegraph systems, such as the Murray Code.

Contents

Introduction

Letter frequencies, like word frequencies, tend to vary, both by writer and by subject. One cannot write an essay about x-rays without using frequent Xs, and the essay will have an idiosyncratic letter frequency if the essay is about the frequent use of x-rays to treat zebras in Qatar. Different authors have habits which can be reflected in their use of letters. Hemingway's writing style, for example, is visibly different from Faulkner's. Letter, bigram, trigram, word frequencies, word length, and sentence length can be calculated for specific authors, and used to prove or disprove authorship of texts, even for authors whose styles are not so divergent.

Accurate average letter frequencies can only be gleaned by analyzing a large amount of representative text. With the availability of modern computing and collections of large text corpora, such calculations are easily made. Examples can be drawn from a variety of sources (press reporting, religious texts, scientific texts and general fiction) and there are differences especially for general fiction with the position of 'h' and 'i', with H becoming more common.

Herbert S. Zim, in his classic introductory cryptography text "Codes and Secret Writing", gives the English letter frequency sequence as "ETAON RISHD LFCMU GYPWB VKJXQ Z", the most common letter pairs as "TH HE AN RE ER IN ON AT ND ST ES EN OF TE ED OR TI HI AS TO", and the most common doubled letters as "LL EE SS OO TT FF RR NN PP CC".[2]

Also, to note that different dialects of a language will also affect a letter's frequency. For example, an author in the United States would produce something where the letter 'z' is more common as words like "analyze", "apologize" or "recognize" contain the letter, whereas in British English they do not. This would highly affect the frequency of the letter 'z' as it is a rarely used letter elsewhere in the English language.[3]

The "top twelve" letters constitute about 80% of the total usage. The "top eight" letters constitute about 65% of the total usage. Letter frequency as a function of rank can be fitted well by several rank functions, with the two-parameter Cocho/Beta rank function being the best.[4] Another rank function with no adjustable free parameter also fits the letter frequency distribution reasonably well[5] (the same function has been used to fit the amino acid frequency in protein sequences.[6]) A spy using the VIC cipher or some other cipher based on a straddling checkerboard typically uses a mnemonic such as "a sin to err" (dropping the second "r")[7][8] or "at one sir"[9] to remember the top eight characters.

The use of letter frequencies and frequency analysis plays a fundamental role in cryptograms and several word puzzle games, including Hangman, Scrabble and the television game show *Wheel of Fortune*. One of the earliest descriptions in classical literature of applying the knowledge of English letter frequency to solving a cryptogram is found in E.A. Poe's famous story *The Gold-Bug*, where the method is successfully applied to decipher a message instructing on the whereabouts of a treasure hidden by Captain Kidd.[10]

Letter frequencies had a strong effect on the design of some keyboard layouts. The most-frequent letters are on the bottom row of the Blickensderfer typewriter, and the home row of the Dvorak Simplified Keyboard.

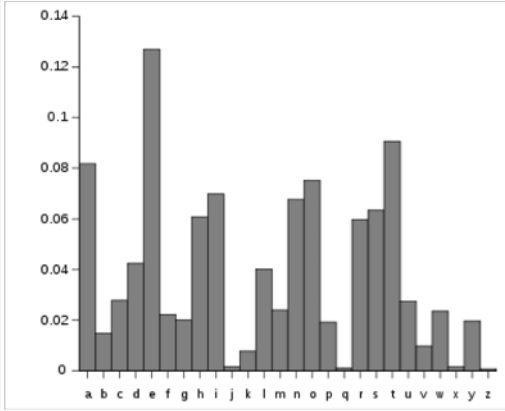
Relative frequencies of letters in the English language

There are three primary ways to count letter frequency that result in very different charts for common letters. The first method, used in the chart below, is to count letter frequency in root words of a dictionary. The second is to include all word variants when counting, such as "abstracts", "abstracted" and "abstracting" and not just the root word of "abstract", which results in letters like "s" appearing much more frequently. A final variant is to count letters based on their frequency of use in actual texts, resulting in certain letter combinations like "th" becoming more common due to the frequent use of common words like "the".

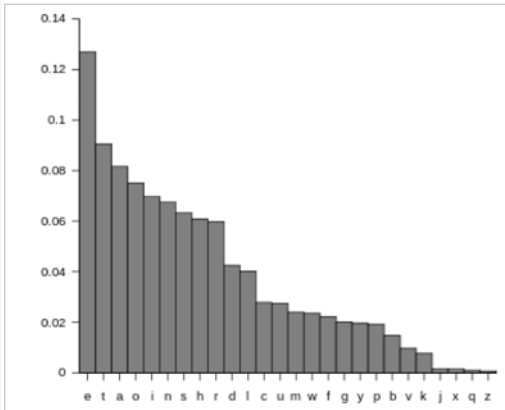
An analysis of entries in the Concise Oxford dictionary, ignoring frequency of word use, gives an order of "EARIOTNSLCUDPMHGBFYWKVXZJQ".[11]

The letter-frequency table below is taken from Pavel Mička's website, which cites Robert Lewand's *Cryptological Mathematics*.[12]

Letter	Relative frequency in the English language	
a	8.167%	
b	1.492%	
c	2.782%	
d	4.253%	
e	12.702%	
f	2.228%	
g	2.015%	
h	6.094%	
i	6.966%	
j	0.153%	
k	0.772%	
l	4.025%	
m	2.406%	
n	6.749%	
o	7.507%	
p	1.929%	
q	0.095%	
r	5.987%	
s	6.327%	
t	9.056%	
u	2.758%	
v	0.978%	
w	2.360%	
x	0.150%	
y	1.974%	
z	0.074%	



Relative frequencies of letters in text.



Relative frequencies ordered by frequency.

According to Lewand, arranged from most to least common in appearance, the letters are: **etaoinshrdlcumwfgypbvkjxqz** Lewand's ordering differs slightly from others, such as Cornell University Math Explorer's Project, which produced a table after measuring 40,000 words.[13]

In English, the space is slightly more frequent than the top letter (e) [14] and the non-alphabetic characters (digits, punctuation, etc.) collectively occupy the fourth position (having already included the space) between *t* and *a*.[15]

Relative frequencies of the first letters of a word in the English language

The frequency of the first letters of words or names is helpful in pre-assigning space in physical files and indexes.[16] Given 26 filing cabinet drawers, rather than a 1:1 assignment of one drawer to one letter of the alphabet, it is often useful to use a more equal-frequency-letter code by assigning several low-frequency letters to the same drawer (often one drawer is labeled VWXYZ), and to split up the most-frequent initial letters — S, A, and C - into several drawers (often 6 drawers Aa-An, Ao-Az, Ca-Cj, Ck-Cz, Sa-Si, Sj-Sz). The same system is used in some multi-volume works such as some encyclopedias. Cutter numbers, another mapping of names to a more equal-frequency code, are used in some libraries.

The first letter of an English word, from most to least common, **t o a w b c d s f m r h i y e g l n p u v j k q z x**[17]

Both the overall letter distribution and the word-initial letter distribution approximately match the Zipf distribution and even more closely match the Yule distribution.[18]

Often the frequency distribution of the first digit in each datum is significantly different from the overall frequency of all the digits in a set of numeric data—see Benford's law for details.

Analysis of a subset of Project Gutenberg text which shows the frequencies which various letters are found at the beginnings of words:[19]