Huffman Coding

Darin Critchlow
CSIS 2430-001

Objective:

Implement Huffman Coding

1. **Use this file: Metamorphosis.txt**
2. **Convert the file to binary and measure the size.**
3. **Implement Huffman Coding to compress the file. (You will calculate the frequencies of all punctuations, etc., build a tree, then use the tree to compress the file (remember, it will be in BINARY - LEAVE IT in this format.)**
4. **Measure the newly compressed file.**
5. **Answer the question: What is the delta shrinkage (Entropy) from the first file to the second?**
6. **Use the frequencies found on page 771 #27 and reimplement.**
7. **Measure the compression again.**
8. **Compare/contrast (what did you find?, why was there much of a difference?, etc.)**

What Worked:

Everything worked properly

What Didn't Work:

Everything worked properly

Comments:

As per our discussion in class, you need to follow these steps to make Huffman coding work as follows:

1. As a baseline, make a copy of the Metamorphosis file and save it as a binary file.
2. With your programming language, read in the file and calculate the frequency of all of the letters and characters (including spaces).
3. Build your tree (good example in the book – chapter 11.2?) from the frequencies.
4. From your text file, use the tree you just created to recode the letters to the new variable bits from your tree in #3 and save it as a second binary file.
5. Compare binary file in #1 to the one in #4 and you should see a reduction (compression) of $x\%$. Remember, all of your answers should be about the same within a small error.

Code:

```
#!/usr/bin/python
from heapq import heappush, heappop, heapify
from collections import defaultdict

with open('Metamorphosis.bin', 'rb') as f:
    data = f.read()

def encode(symb2freq):
    """Huffman encode the given dict mapping symbols to weights"""
    heap = [[wt, [sym, ""]] for sym, wt in symb2freq.items()]
    heapify(heap)
    while len(heap) > 1:
        lo = heappop(heap)
        hi = heappop(heap)
        for pair in lo[1:]:
            pair[1] = '0' + pair[1]
        for pair in hi[1:]:
            pair[1] = '1' + pair[1]
        heappush(heap, [lo[0] + hi[0]] + lo[1:] + hi[1:])
    # print sorted(heappop(heap)[1:], key=lambda p: (len(p[-1]), p))
    return sorted(heappop(heap)[1:], key=lambda p: (len(p[-1]), p))

# txt = "this is an example for huffman encoding"
symb2freq = defaultdict(int)
for ch in data:
    symb2freq[ch] += 1
# print "Symbol\tWeight\tHuffman Code"
huff = encode(symb2freq)
# for p in huff:
#     print "%s\t%s\t%s" % (p[0], symb2freq[p[0]], p[1])

# with open("huffman_values.txt", 'wb') as f4:
#     f4.write(str(huff))

# meta = open('amt.png', 'rb')
# print meta.mode
# print meta.name
# print meta.encoding
# data = meta.read()
# print data
# print type(data)
# print meta.tell()
# print len(data)
# bina = open('Metamorphosis.bin', 'wb')
```

```
# print bina.encoding
# data1 = bina.read()
# print type(data1)
# bina.write(''.join(format(ord(x), 'b') for x in data1))

# import binascii
# with open('Metamorphosis.txt', 'rb') as f:
#     data = f.read()
# # print data
# data_bin = bin(int(binascii.hexlify(data),16))
# # print data_bin
# # n = int(data_bin, 2)
# # data_txt = binascii.unhexlify('%x' %n)
# # print data_txt

# with open('Metamorphosis_original_copy.txt', 'rb') as f2:
#     data = f2.read()
# print data

# with open('Metamorphosis.bin', 'wb') as f3:
#     f3.write(data)
```