Huffman Code

Jochen Ziegenbalg

electronic mail : ziegenbalg@ph – karlsruhe.de
homepage : http : // www.ziegenbalg.ph – karlsruhe.de

Basics

■ The alphabet

The Huffman Code is based on an alphabet and on the frequencies of the letters in this alphabet. In this notebook we will mostly work with the German alphabet (however without "Umlauts" ans "Eszet"). According to Huffman's method of coding, each letter is coded by a binary word. But unlike in the case of ASCII-coding the word length may differ from letter to letter. Letters with a high (low) frequency are coded by "short" ("long") binary words, respectively. This can only work if the code is a prefix-code.

Definition: A code is called a *prefix-code* if none of its codewords is equal to the initial segment of another codeword.

```
In[4]:= Alphabet = {"A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L",
          "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z"}

Out[4]= {A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z}
```

■ Frequencies

Table of frequencies of the letters in the German alphabet (values in percent):

A	6.108
B	2.527
C	2.832
D	5.324
E	15.844
F	1.463
G	2.633
H	4.517
I	8.238
J	0.222
K	1.626
L	3.300
M	2.949
N	9.771

O	2.644
P	0.655
Q	0.012
R	7.700
S	6.704
T	6.471
U	4.727
V	1.041
W	1.346
X	0.023
Y	0.023
Z	1.299

Next, we construct a list consisting of pairs {letter, frequency}:

In[68]:=
```
German = {{"A", 6.109}, {"B", 2.527}, {"C", 2.832}, {"D", 5.324},
  {"E", 15.844}, {"F", 1.463}, {"G", 2.633}, {"H", 4.517},
  {"I", 8.238}, {"J", 0.222}, {"K", 1.626}, {"L", 3.300}, {"M", 2.949},
  {"N", 9.771}, {"O", 2.644}, {"P", 0.655}, {"Q", 0.012}, {"R", 7.700},
  {"S", 6.704}, {"T", 6.471}, {"U", 4.727}, {"V", 1.041},
  {"W", 1.346}, {"X", 0.023}, {"Y", 0.023}, {"Z", 1.299}}
```

Out[68]=
```
{{A, 6.109}, {B, 2.527}, {C, 2.832}, {D, 5.324}, {E, 15.844},
 {F, 1.463}, {G, 2.633}, {H, 4.517}, {I, 8.238}, {J, 0.222},
 {K, 1.626}, {L, 3.3}, {M, 2.949}, {N, 9.771}, {O, 2.644},
 {P, 0.655}, {Q, 0.012}, {R, 7.7}, {S, 6.704}, {T, 6.471}, {U, 4.727},
 {V, 1.041}, {W, 1.346}, {X, 0.023}, {Y, 0.023}, {Z, 1.299}}
```

```
In[69]:= % // TableForm
```

```
Out[69]//TableForm=
        A      6.109
        B      2.527
        C      2.832
        D      5.324
        E      15.844
        F      1.463
        G      2.633
        H      4.517
        I      8.238
        J      0.222
        K      1.626
        L      3.3
        M      2.949
        N      9.771
        O      2.644
        P      0.655
        Q      0.012
        R      7.7
        S      6.704
        T      6.471
        U      4.727
        V      1.041
        W      1.346
        X      0.023
        Y      0.023
        Z      1.299
```

The next command isolates the frequencies (given as percentages).

```
In[70]:= (Transpose[German])[[2]]
```

```
Out[70]= {6.109, 2.527, 2.832, 5.324, 15.844, 1.463, 2.633, 4.517,
          8.238, 0.222, 1.626, 3.3, 2.949, 9.771, 2.644, 0.655, 0.012,
          7.7, 6.704, 6.471, 4.727, 1.041, 1.346, 0.023, 0.023, 1.299}
```

A test for consistency:

```
In[71]:= Apply[Plus, Transpose[German][[2]]]
```

```
Out[71]= 100.
```

The following list is sorted by frequencies. It will be the basic list by whihc we will construct the Huffman tree and the Huffman code.

```
In[72]:= PairSwitch[Pairlist_] := {Pairlist[[2]], Pairlist[[1]]}
```

```
In[73]:= PairSwitch[{A, B}]
```

```
Out[73]= {B, A}
```

In[189]:=
```
Map[PairSwitch, German]
```

Out[189]=
```
{{6.109, A}, {2.527, B}, {2.832, C}, {5.324, D}, {15.844, E},
 {1.463, F}, {2.633, G}, {4.517, H}, {8.238, I}, {0.222, J},
 {1.626, K}, {3.3, L}, {2.949, M}, {9.771, N}, {2.644, O},
 {0.655, P}, {0.012, Q}, {7.7, R}, {6.704, S}, {6.471, T}, {4.727, U},
 {1.041, V}, {1.346, W}, {0.023, X}, {0.023, Y}, {1.299, Z}}
```

In[190]:=
```
% // Sort
```

Out[190]=
```
{{0.012, Q}, {0.023, X}, {0.023, Y}, {0.222, J}, {0.655, P},
 {1.041, V}, {1.299, Z}, {1.346, W}, {1.463, F}, {1.626, K},
 {2.527, B}, {2.633, G}, {2.644, O}, {2.832, C}, {2.949, M},
 {3.3, L}, {4.517, H}, {4.727, U}, {5.324, D}, {6.109, A}, {6.471, T},
 {6.704, S}, {7.7, R}, {8.238, I}, {9.771, N}, {15.844, E}}
```

In[191]:=
```
FTGerman = Sort[Map[PairSwitch, German]]
(* FT for Frequency Table *)
```

Out[191]=
```
{{0.012, Q}, {0.023, X}, {0.023, Y}, {0.222, J}, {0.655, P},
 {1.041, V}, {1.299, Z}, {1.346, W}, {1.463, F}, {1.626, K},
 {2.527, B}, {2.633, G}, {2.644, O}, {2.832, C}, {2.949, M},
 {3.3, L}, {4.517, H}, {4.727, U}, {5.324, D}, {6.109, A}, {6.471, T},
 {6.704, S}, {7.7, R}, {8.238, I}, {9.771, N}, {15.844, E}}
```

Constructing the Huffman Code

Remarks on the implementation: The construction of the Huffman code is based on a given alphabet with frequencies; more precisely on a list of pairs {frequency, letter} - see for instance the list FTGerman above. (Having the frequency value in the first and the letter in the second position is only a matter of convenience.) This list is referred to as the underlying frequency table FT - which is the only input parameter of the function HuffmanTree.

The Huffman tree is constructed as a binary root-tree. Its subtrees are of the form {value, {left child, right child}} or {value, letter}. The latter ones are the "leaves" of the Huffman tree. The construction of the Huffman tree is done by a series of transformations on FT as follows:

* FT is given in sorted form (sorted by the values of the subtrees).
* The two subtrees with the lowest values (i.e. the first and the second subtree of FT are merged into another subtree, its value being the sum of the latter two.)
* This new subtree replaces the two subtrees from which it was constructed.
* This procedure is continued until there is only one subtree.

In[192]:=

```
LowestValue1[T_] := First[T[[1]]];
LowestValue2[T_] := First[T[[2]]];

HuffmanTree[FT_] :=
 If[Length[FT] == 1, FT,
  HuffmanTree[Sort[Prepend[Delete[FT, {{1}, {2}}],
     {LowestValue1[FT] + LowestValue2[FT], {FT[[1]], FT[[2]]}}]]]]
```

In[195]:=

```
HuffmanTree[FTGerman]
```

Out[195]=

```
{{100., {{41.131, {{19.015, {{9.244, {{4.517, H}, {4.727, U}}}, {9.771, N}}},
      {22.116, {{10.601, {{5.277, {{2.633, G}, {2.644, O}}}, {5.324, D}}},
        {11.515, {{5.477, {{2.645, {{1.299, Z}, {1.346, W}}}, {2.832, C}}},
          {6.038, {{2.949, M}, {3.089, {{1.463, F}, {1.626, K}}}}}}}}}}},
   {58.869, {{26.984, {{12.58, {{6.109, A}, {6.471, T}}},
       {14.404, {{6.704, S}, {7.7, R}}}}},
     {31.885, {{15.844, E}, {16.041, {{7.803, {{3.3, L},
           {4.503, {{1.976, {{0.935, {{0.28, {{0.058, {{0.023, Y}, {0.035,
                     {{0.012, Q}, {0.023, X}}}}}, {0.222, J}}}, {0.655, P}}},
               {1.041, V}}}, {2.527, B}}}}}, {8.238, I}}}}}}}}}
```

In[196]:=

```
% // TableForm
```

Out[196]//TableForm=

			4.517	H						
		19.015	9.244	4.727	U					
			9.771	N						
				2.633	G					
			10.601	5.277	2.644	O				
				5.324	D					
	41.131					1.299	Z			
		22.116		5.477	2.645	1.346	W			
						2.832	C			
			11.515			2.949	M			
				6.038	3.089	1.463	F			
						1.626	K			
100.			12.58	6.109	A					
		26.984		6.471	T					
			14.404	6.704	S					
				7.7	R					
			15.844	E						
				3.3	L					
	58.869									0.0
						0.935	0.28	0.058		0.0
		31.885	16.041	7.803	4.503	1.976			0.222	J
								0.655	P	
							1.041	V		
						2.527	B			
				8.238	I					

In[203]:=

```
LeftChild[HT_] := { ((First[HT])[[2]])[[1]] };

RightChild[HT_] := { ((First[HT])[[2]])[[2]] };

LeafP[HT_] := Not[ListQ[(HT[[1]])[[2]]]] ;    (* Leaf-Property *)

Letter[Pair_] := Pair[[2]];

HuffmanCodeTable[HT_] :=
  If[LeafP[HT], {Letter[First[HT]], "1"}, Sort[HCT[HT, ""]]];

HCT[HT_, code_] :=
 Which[HT == {}, {},
  LeafP[HT], Return[{{Letter[First[HT]], code}}],
  True,
  Join[
   HCT[LeftChild[HT], StringJoin[code, "0"]],
   HCT[RightChild[HT], StringJoin[code, "1"]]] ]
```

In[209]:=

```
HuffmanCodeTable[{{100, {{40, "U"}, {60, "V"}}}}]
```

Out[209]=

```
{{U, 0}, {V, 1}}
```

In[210]:=

```
HuffmanCodeTable[HuffmanTree[FTGerman]]
```

Out[210]=

```
{{A, 1000}, {B, 111011}, {C, 01101}, {D, 0101}, {E, 110}, {F, 011110},
 {G, 01000}, {H, 0000}, {I, 1111}, {J, 111010001}, {K, 011111},
 {L, 11100}, {M, 01110}, {N, 001}, {O, 01001}, {P, 11101001},
 {Q, 11101000010}, {R, 1011}, {S, 1010}, {T, 1001}, {U, 0001},
 {V, 1110101}, {W, 011001}, {X, 11101000011}, {Y, 1110100000}, {Z, 011000}}
```

Some further utility functions

In[211]:=

```
FTGerman
```

Out[211]=

```
{{0.012, Q}, {0.023, X}, {0.023, Y}, {0.222, J}, {0.655, P},
 {1.041, V}, {1.299, Z}, {1.346, W}, {1.463, F}, {1.626, K},
 {2.527, B}, {2.633, G}, {2.644, O}, {2.832, C}, {2.949, M},
 {3.3, L}, {4.517, H}, {4.727, U}, {5.324, D}, {6.109, A}, {6.471, T},
 {6.704, S}, {7.7, R}, {8.238, I}, {9.771, N}, {15.844, E}}
```

In[212]:=

```
Cases[FTGerman, {_, "D"}]
```

Out[212]=

```
{{5.324, D}}
```

In[93]:=
```
Frequency[FT_, symb_] := First[First[Cases[FT, {_, symb}]]]
```

In[157]:=
```
Frequency[FTGerman, "D"]
```

Out[157]=
```
5.324
```

In[213]:=
```
HuffmanCode[FT_, symb_] :=
 Last[First[Cases[HuffmanCodeTable[HuffmanTree[FT]], {symb, _}]]]
```

In[159]:=
```
HuffmanCode[FTGerman, "D"]
```

Out[159]=
```
0101
```

In[214]:=
```
MeanCodeLength[FT_] :=
 Module[{alph, freq, codes, mean = 0},
  alph = Sort[(Transpose[FT])[[2]]];
  freq = Map[Function[x, Frequency[FT, x]], alph];
  codes = Map[Function[x, HuffmanCode[FT, x]], alph];
  (* Print[{alph, freq, codes}]; *)
  Do[mean = mean + freq[[i]] * StringLength[codes[[i]] ], {i, 1, Length[alph]} ];
  mean = 0.01 * mean;
  Return[mean]]
```

In[215]:=
```
MeanCodeLength[FTGerman]
```

Out[215]=
```
4.12501
```

Some Examples

■ Example: ABRAKADABRA

In[224]:=
```
ExABRAKADABRA = Sort[{{100 * 5 / 11, "A"}, {100 * 2 / 11, "B"},
    {100 * 2 / 11, "R"}, {100 * 1 / 11, "K"}, {100 * 1 / 11, "D"}}] // N
```

Out[224]=
```
{{9.09091, D}, {9.09091, K}, {18.1818, B}, {18.1818, R}, {45.4545, A}}
```

In[225]:=
```
HuffmanTree[ExABRAKADABRA]
```

Out[225]=
```
{{100., {{45.4545, A}, {54.5455, {{18.1818, {{9.09091, D}, {9.09091, K}}},
     {36.3636, {{18.1818, B}, {18.1818, R}}}}}}}}
```

In[226]:=
```
% // TableForm
```

Out[226]//TableForm=

	45.4545	A		
		18.1818	9.09091	D
100.			9.09091	K
	54.5455	36.3636	18.1818	B
			18.1818	R

In[227]:=
```
HuffmanCodeTable[HuffmanTree[ExABRAKADABRA]]
```

Out[227]=
```
{{A, 0}, {B, 110}, {D, 100}, {K, 101}, {R, 111}}
```

In[228]:=
```
MeanCodeLength[ExABRAKADABRA]
```

Out[228]=
```
2.09091
```

■ Example: MISSISSIPPISCHIFF

In[229]:=
```
ExMISSISSIPPISCHIFF =
 Sort[{{100 * 1 / 17, "M"}, {100 * 5 / 17, "I"}, {100 * 5 / 17, "S"}, {100 * 2 / 17,
     "P"}, {100 * 1 / 17, "C"}, {100 * 1 / 17, "H"}, {100 * 2 / 17, "F"}}] // N
```

Out[229]=
```
{{5.88235, C}, {5.88235, H}, {5.88235, M},
 {11.7647, F}, {11.7647, P}, {29.4118, I}, {29.4118, S}}
```

In[219]:=
```
Apply[Plus, Map[First, %]]
```

Out[219]=
```
100.
```

In[230]:=
```
HuffmanTree[ExMISSISSIPPISCHIFF]
```

Out[230]=
```
{{100., {{41.1765, {{17.6471, {{5.88235, M}, {11.7647, F}}},
      {23.5294, {{11.7647, P}, {11.7647, {{5.88235, C}, {5.88235, H}}}}}}},
   {58.8235, {{29.4118, I}, {29.4118, S}}}}}}
```

In[231]:=
```
% // TableForm
```

Out[231]//TableForm=

		17.6471	5.88235	M
			11.7647	F
	41.1765		11.7647	P
100.		23.5294	11.7647	5.88235 C
				5.88235 H
		29.4118	I	
	58.8235	29.4118	S	

In[232]:=

```
HuffmanCodeTable[HuffmanTree[ExMISSISSIPPISCHIFF]]
```

Out[232]=

```
{{C, 0110}, {F, 001}, {H, 0111}, {I, 10}, {M, 000}, {P, 010}, {S, 11}}
```

In[223]:=

```
MeanCodeLength[ExMISSISSIPPISCHIFF]
```

Out[223]=

```
2.52941
```