Predictive Coding

- The next symbol can be statistically predicted from the past.
  - Code with context
  - Code the difference
  - Move to front, then code
- Goal of prediction
  - The prediction should make the probability of the next symbol high as possible
  - After prediction there is nothing left to know except the probabilities

Bad and Good Prediction

- From information theory – The lower the information the fewer bits are needed to code the symbol.
  \[ \text{inf}(a) = \log_2 \left( \frac{1}{P(a)} \right) \]
- Examples:
  - \( P(a) = \frac{1024}{1024}, \text{inf}(a) = .000977 \)
  - \( P(a) = \frac{1}{2}, \text{inf}(a) = 1 \)
  - \( P(a) = \frac{1}{1024}, \text{inf}(a) = 10 \)

Entropy

- Entropy is the expected number of bit to code a symbol in the model with a, having probability \( P(a) \).
  \[ H = \sum_{a=1}^{m} P(a) \log_2 \left( \frac{1}{P(a)} \right) \]
- Good coders should be close to this bound.
  - Arithmetic
  - Huffman
  - Golomb
  - Tunstall

PPM

- Prediction with Partial Matching
  - Cleary and Witten (1984)
  - Tries to find a good context to code the next symbol

<table>
<thead>
<tr>
<th>context</th>
<th>a...e...i...f...s...y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>the</td>
</tr>
<tr>
<td></td>
<td>h e</td>
</tr>
<tr>
<td></td>
<td>e</td>
</tr>
<tr>
<td>&lt;nil&gt;</td>
<td>50 70 30 35 40 13</td>
</tr>
</tbody>
</table>

- Uses adaptive arithmetic coding for each context

JBIG

- Coder for binary images
  - documents
  - graphics
- Codes in scan line order using context from the same and previous scan lines.

- Uses adaptive arithmetic coding with context
**Issues with Context**

- **Context dilution**
  - If there are too many contexts then too few symbols are coded in each context, making them ineffective because of the zero-frequency problem.
- **Context saturation**
  - If there are too few contexts then the contexts might not be good as having more contexts.
- **Wrong context**
  - Again poor predictors.

**General Differencing**

- Let $x_1, x_2, ..., x_n$ be some numerical data that is correlated, that is $x_i$ is near $x_{i+1}$
- Better compression can result from coding $x_1, x_2 - x_1, x_3 - x_2, ..., x_n - x_{n-1}$
- This idea is used in
  - Signal coding
  - Audio coding
  - Video coding
- There are fancier prediction methods based on linear combinations of previous data, but these can require training.

**Move to Front Coding**

- Non-numerical data
- The data have a relatively small working set that changes over the sequence.
- Example: `ababacbbccbdcc`
- Move to Front algorithm
  - Symbols are kept in a list indexed 0 to m-1
  - To code a symbol output its index and move the symbol to the front of the list

**Example**

```
0 1 2 3
a b c d
```
Example

- Example: `ababaabccbbccccbdc`  
  
  0 1 2 3  
  a b c d  
  ↓  
  0 1 2 3  
  b a c d

Example

- Example: `ababaabccbbccccbdc`  
  
  0 1 2 3  
  b a c d  
  ↓  
  0 1 2 3  
  a b c d

Example

- Example: `ababaabccbbccccbdc`  
  
  0 1 2 3  
  a b c d  
  ↓  
  0 1 2 3  
  b a c d

Example

- Example: `ababaabccbbccccbdc`  
  
  0 1 2 3  
  b a c d  
  ↓  
  0 1 2 3  
  a b c d

Example

- Example: `ababaabccbbccccbdc`  
  
  0 1 2 3  
  b a c d  
  ↓  
  0 1 2 3  
  b a c d

Example

- Example: `ababaabccbbccccbdc`  
  
  0 1 2 3  
  b a c d  
  ↓  
  0 1 2 3  
  b a c d

Example

- Example: `ababaabccbbccccbdc`  
  
  0 1 2 3  
  b a c d  
  ↓  
  0 1 2 3  
  b a c d
Example

• Example: \texttt{abababccbccbccbd}
  
  \begin{tabular}{cccc}
  0 & 1 & 2 & 3 \\
  b & a & c & d \\
  \\
  0 & 1 & 2 & 3 \\
  c & b & a & d \\
  \end{tabular}

Example

• Example: \texttt{abababccbccbccbd}
  
  \begin{tabular}{cccc}
  0 & 1 & 2 & 3 \\
  c & b & d & a \\
  \end{tabular}

Example

• Example: \texttt{abababccbccbccbd}
  
  Frequencies of \{a, b, c, d\}
  a  b  c  d  
  4  7  8  1  

  Frequencies of \{0, 1, 2, 3\}
  0  1  2  3  
  8  9  2  1  

Extreme Example

Input:
\texttt{aaaaaaaaabbbbbbbbbcccccddddd}

Output
\texttt{0000000000100000000200000000030000000}

Frequencies of \{a, b, c, d\}
\begin{tabular}{cccc}
  a & b & c & d \\
  10 & 10 & 10 & 10 \\
\end{tabular}

Frequencies of \{0, 1, 2, 3\}
\begin{tabular}{cccc}
  0 & 1 & 2 & 3 \\
  37 & 11 & 11 & 11 \\
\end{tabular}

Burrows-Wheeler Transform

• Burrows-Wheeler, 1994
• BW Transform creates a representation of the data which has a small working set.
• The transformed data is compressed with move to front compression.
• The decoder is quite different from the encoder.
• The algorithm requires processing the entire string at once (it is not on-line).
• It is a remarkably good compression method.

Encoding Example

• abracadabra
  1. Create all cyclic shifts of the string.
  \begin{tabular}{c}
  0 & abracadabra \\
  1 & bracadabra \\
  2 & cadabraabr \\
  3 & acabradabra \\
  4 & cbracadabra \\
  5 & dbracadabra \\
  6 & ebbradabra \\
  7 & fbracadabra \\
  8 & gbradabra \\
  9 & hbradabra \\
  10 & iabracadabra \\
\end{tabular}
Encoding Example

2. Sort the strings alphabetically into array A

\[
\begin{array}{c|c}
0 & abracadabra \\
1 & bracadabra \\
2 & racadabra \\
3 & cadabra \\
4 & adaabra \\
5 & draabra \\
6 & raabra \\
7 & abaabra \\
8 & aabara \\
9 & arcabra \\
10 & raabra \\
\end{array}
\]

A

A_

Encoding Example

3. L = the last column

\[
\begin{array}{c|c}
0 & abracadabra \\
1 & abracadabra \\
2 & abracadabra \\
3 & abracadabra \\
4 & abracadabra \\
5 & abracadabra \\
6 & abracadabra \\
7 & abracadabra \\
8 & abracadabra \\
9 & abracadabra \\
10 & abracadabra \\
\end{array}
\]

L = radacaabb

X = 2

Why BW Works

- Ignore decoding for the moment.
- The prefix of each shifted string is a context for the last symbol.
  - The last symbol appears just before the prefix in the original.
- By sorting similar contexts are adjacent.
  - This means that the predicted last symbols are similar.

Decoding Example

- We first decode assuming some information. We then show how compute the information.
- Let A' be A shifted by 1

\[
\begin{array}{c|c}
0 & abracadabra \\
1 & abracadabra \\
2 & abracadabra \\
3 & abracadabra \\
4 & abracadabra \\
5 & abracadabra \\
6 & abracadabra \\
7 & abracadabra \\
8 & abracadabra \\
9 & abracadabra \\
10 & abracadabra \\
\end{array}
\]

A

A'

Decoding Example

- Assume we know the mapping T[j] is the index in A of the string i in A.
- T = [2 5 6 7 8 9 10 4 1 0 3]
Decoding Example

• Let $F$ be the first column of $A$, it is just $L$, sorted.
  $F = \begin{array}{cccccccccc}
    0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
    a & a & a & a & b & b & c & d & r & r
  \end{array}$
  $T = \begin{array}{cccccccccc}
    0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
    2 & 5 & 6 & 7 & 8 & 9 & 10 & 4 & 1 & 0 & 3
  \end{array}$

• Follow the pointers in $T$ in $F$ to recover the input starting with $X$.

Decoding Example

• Why does this work?
  • The first symbol of $A[T[i]]$ is the second symbol of $A'[T[i]]$ is the second symbol of $A[i]$ because $A'[T[i]] = A[i]$.

Decoding Example

• How do we compute $T$ from $L$ and $X$?
  0 1 2 3 4 5 6 7 8 9 10
  a a a a a b b c d r r
  $F = \begin{array}{cccccccccc}
    0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
    a & a & a & a & b & b & c & d & r & r
  \end{array}$
  2 5 6 7 8 9 10 4 1 0 3
  $T = \begin{array}{cccccccccc}
    0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
    2 & 5 & 6 & 7 & 8 & 9 & 10 & 4 & 1 & 0 & 3
  \end{array}$
  ab
  a

Decoding Example

• How do we compute $T$ from $L$ and $X$?
  0 1 2 3 4 5 6 7 8 9 10
  a a a a a b b c d r r
  $F = \begin{array}{cccccccccc}
    0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
    a & a & a & a & b & b & c & d & r & r
  \end{array}$
  2 5 6 7 8 9 10 4 1 0 3
  $T = \begin{array}{cccccccccc}
    0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
    2 & 5 & 6 & 7 & 8 & 9 & 10 & 4 & 1 & 0 & 3
  \end{array}$
  abr
  abr
  a

Decoding Example

• How do we compute $T$ from $L$ and $X$?
  0 1 2 3 4 5 6 7 8 9 10
  a a a a a b b c d r r
  $F = \begin{array}{cccccccccc}
    0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
    a & a & a & a & b & b & c & d & r & r
  \end{array}$
  2 5 6 7 8 9 10 4 1 0 3
  $L = r d a r c a a a a b b$
  Note that $L$ is the first column of $A'$ and $A'$ is in the same order as $A$.

  If $i$ is the $k$-th $x$ in $F$ then $T[i]$ is the $k$-th $x$ in $L$. 

Decoding Example

• How do we compute $T$ from $L$ and $X$?
  0 1 2 3 4 5 6 7 8 9 10
  a a a a a b b c d r r
  $F = \begin{array}{cccccccccc}
    0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
    a & a & a & a & b & b & c & d & r & r
  \end{array}$
  2 5 6 7 8 9 10 4 1 0 3
  $L = r d a r c a a a a b b$
  Note that $L$ is the first column of $A'$ and $A'$ is in the same order as $A$.

  If $i$ is the $k$-th $x$ in $F$ then $T[i]$ is the $k$-th $x$ in $L$. 

Decoding Example

0 1 2 3 4 5 6 7 8 9 10
F = a a a a a b b c c d d r r
L = r d a r c a a a a a b b
T = 0 1 2 3 4 5 6 7 8 9 10
2 5 6 7 8

Decoding Example

0 1 2 3 4 5 6 7 8 9 10
F = a a a a a b b c c d d r r
L = r d a r c a a a a a b b
T = 0 1 2 3 4 5 6 7 8 9 10
2 5 6 7 8 9 10

Decoding Example

0 1 2 3 4 5 6 7 8 9 10
F = a a a a a b b c c d d r r
L = r d a r c a a a a a b b
T = 0 1 2 3 4 5 6 7 8 9 10
2 5 6 7 8 9 10
4

Decoding Example

0 1 2 3 4 5 6 7 8 9 10
F = a a a a a b b c c d d r r
L = r d a r c a a a a a b b
T = 0 1 2 3 4 5 6 7 8 9 10
2 5 6 7 8 9 10
4 1

Decoding Example

0 1 2 3 4 5 6 7 8 9 10
F = a a a a a b b c c d d r r
L = r d a r c a a a a a b b
T = 0 1 2 3 4 5 6 7 8 9 10
2 5 6 7 8 9 10
4 1 0 3

Notes on BW

• Alphabetic sorting does not need the entire cyclic shifted inputs. You just have to look at long enough prefixes.
  – A bucket sort will work here.
• There are high quality practical implementations
  – Bzip
  – Bzip2 (seems to be public domain)