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 distribution of probabilities of the next symbol as skewed as possible
  - After prediction there is no way to predict more so we are in the first order entropy model

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:
  - \( P(a) = \frac{1023}{1024} \), \( \text{inf}(a) = 0.000977 \)
  - \( P(a) = \frac{1}{2} \), \( \text{inf}(a) = 1 \)
  - \( P(a) = \frac{1}{1024} \), \( \text{inf}(a) = 10 \)

Entropy

- Entropy is the expected number of bits to code a symbol in the model with a having probability \( P(a_i) \).

  \[ H = \sum_{i=1}^{n} P(a_i) \log_2 \left( \frac{1}{P(a_i)} \right) \]

- 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>good</th>
<th>context</th>
<th>a...e...i...f...s...y</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0 0 5 7 4 7</td>
<td></td>
</tr>
<tr>
<td>he</td>
<td>10 1 7 10 9 7</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>12 2 10 15 10 10</td>
<td></td>
</tr>
<tr>
<td>&lt;nil&gt;</td>
<td>50 70 30 35 40 13</td>
<td></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
JBIG Example

<table>
<thead>
<tr>
<th>next bit</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>frequency</td>
<td>100</td>
<td>10</td>
</tr>
</tbody>
</table>

\[ H = -\frac{10}{110} \log_2 \frac{110}{110} - \frac{10}{110} \log_2 \frac{110}{100} = .44 \]

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 as good as having more contexts.
- Wrong context
  - Again poor predictors.

Prediction by Differencing

- Used for Numerical Data
- Example: 2 3 4 5 6 7 8 7 6 5 4 3 2

\[
\begin{align*}
2 &\quad 1 \quad 1 \\
3 &\quad 2 \\
4 &\quad 1 \\
5 &\quad 1 \\
6 &\quad 1 \\
7 &\quad 1 \\
8 &\quad 0 \\
7 &\quad 0 \\
6 &\quad 0 \\
5 &\quad 0 \\
4 &\quad 0 \\
3 &\quad 0 \\
2 &\quad 0
\end{align*}
\]

- Transform to 2 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1
  - much lower first-order entropy

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 may require training.

Move to Front Coding

- Non-numerical data
- The data have a relatively small working set that changes over the sequence.
- Example: a b a b a b c b b c c b c b c c
- 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

- Example: a b a b a b c b b c c b c b c c

\[
\begin{align*}
0 &\quad 1 \\
1 &\quad 2 \\
3 &\quad 0 \\
2 &\quad 0 \\
a &\quad b \\
c &\quad d
\end{align*}
\]
Example
• Example: a b a b a a b c b b c c b d b c c
  0 1 2 3
  a b c d
  |
  0 1 2 3
  b a c d

Example
• Example: a b a b a b a a b c c b d b c c
  0 1 2 3
  b a c d
  |
  0 1 2 3
  a b c d

Example
• Example: a b a b a b a b a a b c c b c c b d b c c
  0 1 2 3
  b a c d
  |
  0 1 2 3
  a b c d

Example
• Example: a b a b a b a b a b a a b c c b c c b d b c c
  0 1 2 3
  b a c d
  |
  0 1 2 3
  a b c d
Example

- Example: \texttt{abababccbbcccbdbcc} \\
  \texttt{0 1 1 1 0 1 2} \\
  \texttt{b a c d} \\
  \texttt{0 1 2 3} \\
  \texttt{c b a d}

Example

- Example: \texttt{abababccbbcccbdbcc} \\
  \texttt{0 1 1 1 0 1 2 0 1 0 1 0 0 1 3 1 2 0} \\
  \texttt{c b d a}

Example

- Example: \texttt{abababccbbcccbdbcc} \\
  \texttt{0 1 1 1 0 1 2 0 1 0 1 0 0 1 3 1 2 0} \\
  Frequencies of \{a, b, c, d\} \\
  \texttt{a b c d} \\
  4 7 8 1 \\
  Frequencies of \{0, 1, 2, 3\} \\
  \texttt{0 1 2 3} \\
  8 9 2 1

Extreme Example

Input: \texttt{aaaaaaaaabbbbbbbcccccddddd}
Output: \texttt{0000000000100000000020000000003000000000}

Frequencies of \{a, b, c, d\} \\
\texttt{a b c d} \\
10 10 10 10

Frequencies of \{0, 1, 2, 3\} \\
\texttt{0 1 2 3} \\
37 1 1 1

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.

| 0 | abracadabra |
| 1 | brazacabra |
| 2 | cacababra |
| 3 | dacababra |
| 5 | adacabra |
| 6 | dabababra |
| 7 | abracadab |
| 9 | rasabracadab |
| 10 | aabracadab |
Encoding Example

2. Sort the strings alphabetically into array A

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>abracadabra</td>
</tr>
<tr>
<td>1</td>
<td>abraabracad</td>
</tr>
<tr>
<td>2</td>
<td>radacabra</td>
</tr>
<tr>
<td>3</td>
<td>acadabraabr</td>
</tr>
<tr>
<td>4</td>
<td>cadabraabr</td>
</tr>
<tr>
<td>5</td>
<td>abraabracad</td>
</tr>
<tr>
<td>6</td>
<td>abraabraabr</td>
</tr>
<tr>
<td>7</td>
<td>abracaabra</td>
</tr>
<tr>
<td>8</td>
<td>abraabraa</td>
</tr>
<tr>
<td>9</td>
<td>raabraabbr</td>
</tr>
<tr>
<td>10</td>
<td>abraabraa</td>
</tr>
</tbody>
</table>

A

Encoding Example

3. L = the last column

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>abracadabra</td>
</tr>
<tr>
<td>1</td>
<td>abraabracad</td>
</tr>
<tr>
<td>2</td>
<td>radacabra</td>
</tr>
<tr>
<td>3</td>
<td>acadabraabr</td>
</tr>
<tr>
<td>4</td>
<td>cadabraabr</td>
</tr>
<tr>
<td>5</td>
<td>abraabracad</td>
</tr>
<tr>
<td>6</td>
<td>abraabraabr</td>
</tr>
<tr>
<td>7</td>
<td>abracaabra</td>
</tr>
<tr>
<td>8</td>
<td>abraabraa</td>
</tr>
<tr>
<td>9</td>
<td>raabraabbr</td>
</tr>
<tr>
<td>10</td>
<td>abraabraa</td>
</tr>
</tbody>
</table>

L = rdarcaaaabb

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 to compute the information.
- Let $A'$ be $A$ shifted by 1

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>abracadabra</td>
</tr>
<tr>
<td>1</td>
<td>abraabracad</td>
</tr>
<tr>
<td>2</td>
<td>radacabra</td>
</tr>
<tr>
<td>3</td>
<td>acadabraabr</td>
</tr>
<tr>
<td>4</td>
<td>cadabraabr</td>
</tr>
<tr>
<td>5</td>
<td>abraabracad</td>
</tr>
<tr>
<td>6</td>
<td>abraabraabl</td>
</tr>
<tr>
<td>7</td>
<td>abracaabra</td>
</tr>
<tr>
<td>8</td>
<td>abraabraa</td>
</tr>
<tr>
<td>9</td>
<td>raabraabbr</td>
</tr>
<tr>
<td>10</td>
<td>abraabraa</td>
</tr>
</tbody>
</table>

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.

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

$$T = 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10$$
$$2 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 4 \ 1 \ 0 \ 3$$

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

Decoding Example

- Why does this work?

```
A    T    A'
0    aabracadabr 2    raabracadab
1   abraabradc 5    dbraabraca
2    abracadabra 6    2aaabraabdr
3    acadabraabra 7    3racaabraab
4    adabraabra 8    4cadaabraab
5    braabracada 9    5abraabraad
6    bracadabraa 10   6abracadabra
7    cadabraabra 4    7acadabraabra
8    dababraabra 1    8adabraabra
9    raabraabracad 0   9braabraabra
10   racadabraabra 3   10bracadabra
```

Decoding Example

- How do we compute $F$ and $T$ from $L$ and $X$?
- $F$ is just $L$ sorted

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

```
T = 0 1 2 3 4 5 6 7 8 9 10
2 5 6 7 8 9 10 4 1 0 3
```

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

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

Decoding Example

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

Decoding Example

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

Decoding Example

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

Notes on BW

- Alphabetic sorting does not need the entire cyclic shifted inputs.
  - Sort the indices of the string
  - Most significant symbols first radix sort works
- There are high quality practical implementations
  - Bzip
  - Bzip2 (seems to be w/o patents)
Encoding Exercise

Encode the string ababababababababab = (ab)^8
1. Find L and X
2. Do move-to-front coding of L.
3. Estimate the length of the code using first order entropy.

Decoding Exercise

Decode L = baaaaaba, X = 6
1. First Compute F and T
2. Use those to decode.