Fast Text Compression with Neural Networks

Matthew Mahoney
Florida Institute of Technology

http://cs.fit.edu/~mmahoney/compression/

- How text compression works
- Neural implementations have been too slow
- How to make them faster

How Text Compression Works

Common character sequences can have shorter codes

Morse Code
e = .
z = --..

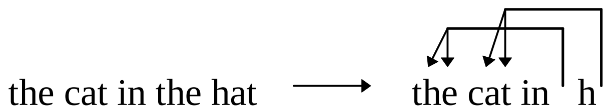
Shorter code	**Longer code**
e	*z*
dog	*dgo*
of the	*the of*
roses are red	*roses are green*

Text compression is an AI problem

Types of compression

From fast but poor...
to slow but good

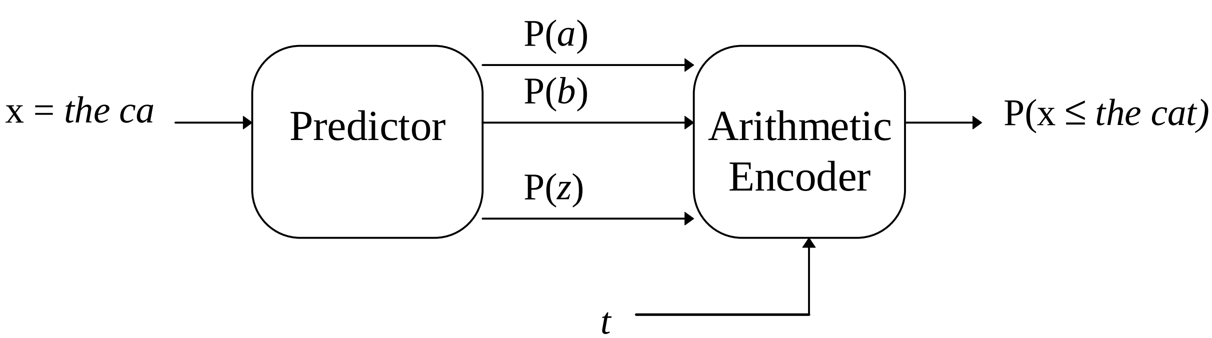
Limpel-Ziv (*compress, zip, gzip, gif*)



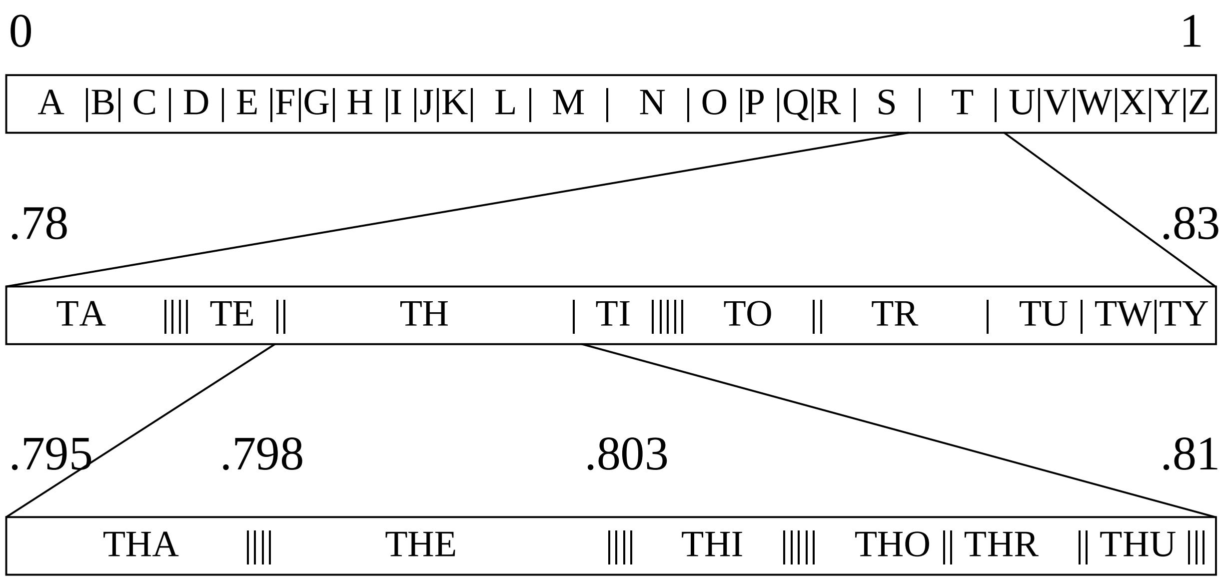
Context Sorting (*Burrows-Wheeler (szip)*)

```
the ca|t  --->  2t 1a 2_ 2e
the ha|t
 the c|a
in the|_
at the|_
 in th|e
hat th|e
```
(*run-length code*)

Predictive Arithmetic (*PPMZ (boa, rkive) and neural network*)



Arithmetic Encoding

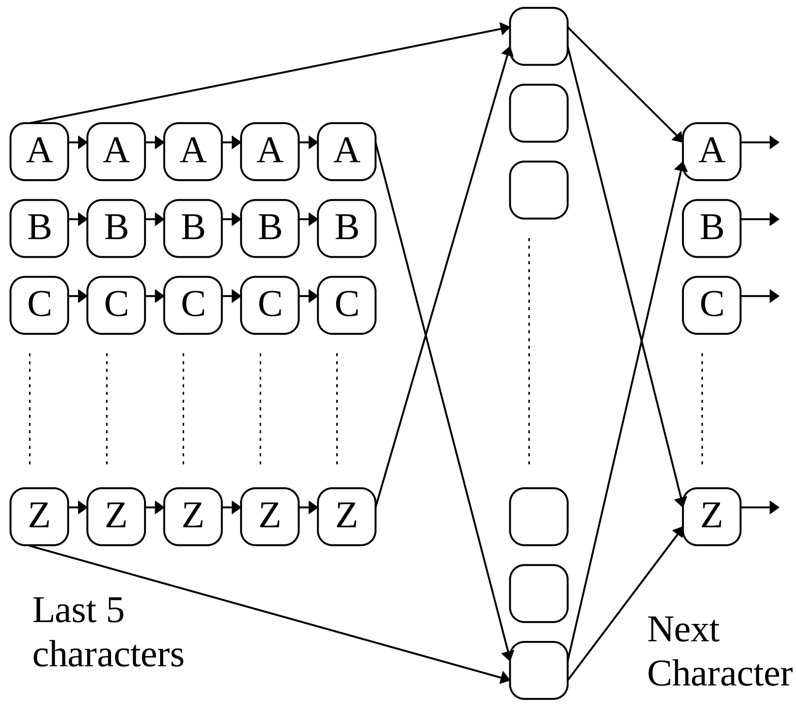


P("THE") = 0.005
Compress("THE") = .8

Binary code for *x* is within 1 bit of $\log_2 1/P(x)$
(Theoretical limit, Shannon, 1949)

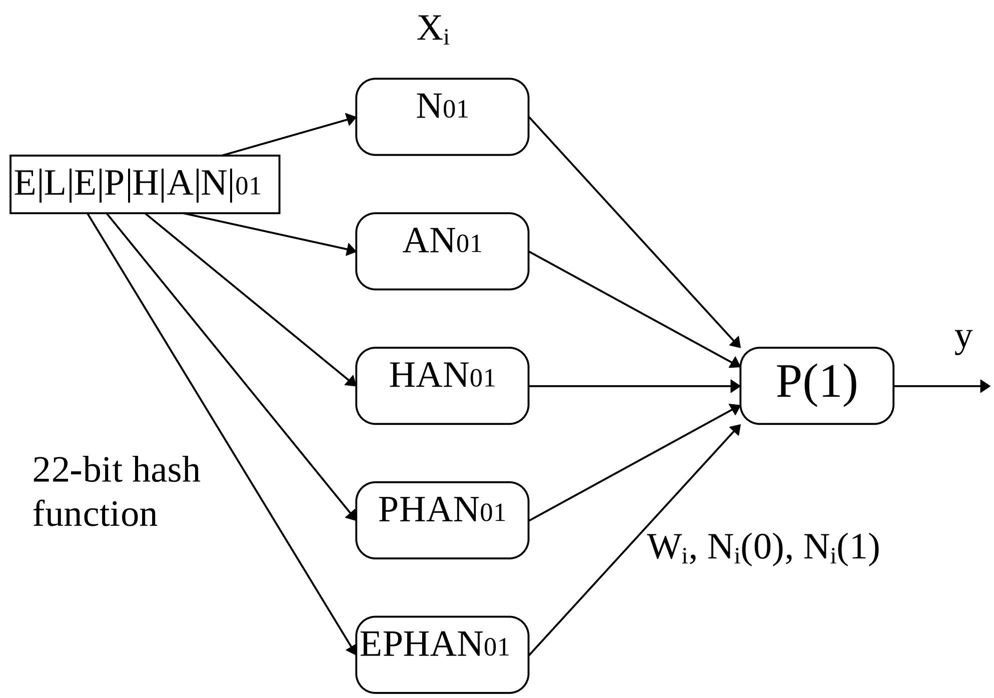
Compression depends entirely on accuracy of P.

Schmidhuber and Heil (1994)
Neural Network Predictor



- 80 character alphabet
- 3 layer network
- 400 input units (last 5 characters)
- 430 hidden units
- 80 output units
- Trained off line in 25 passes by back propagation
- Training time: 3 days on 600KB of text (HP-700)
- 18% better compression than *gzip -9*

Fast Neural Network Predictor



- Predicts one bit at a time
- 2 layer network
- 2^{22} (about 4 million) input units
- One output unit
- Hash function selects 5 or 6 inputs = 1, all others 0
- Trained on line using variable learning rate
- Compresses 600KB in 15 seconds (475 MHz P6-II)
- 42-47% better compression than *gzip -9*

Prediction

$P(1) = g(\Sigma_i w_i x_i)$ *Weighted sum of inputs*

$g(x) = 1/(1 + e^{-x})$ *Squashing function*

Training

$N_i(y) \leftarrow N_i(y) + x_i$ *Count 0 or 1 in context i*

$E = y - P(1)$ *Output error*

$w_i \leftarrow w_i + (\eta_S + \eta_L/\sigma^2{}_i)x_iE$ *Adjust weight to reduce error*

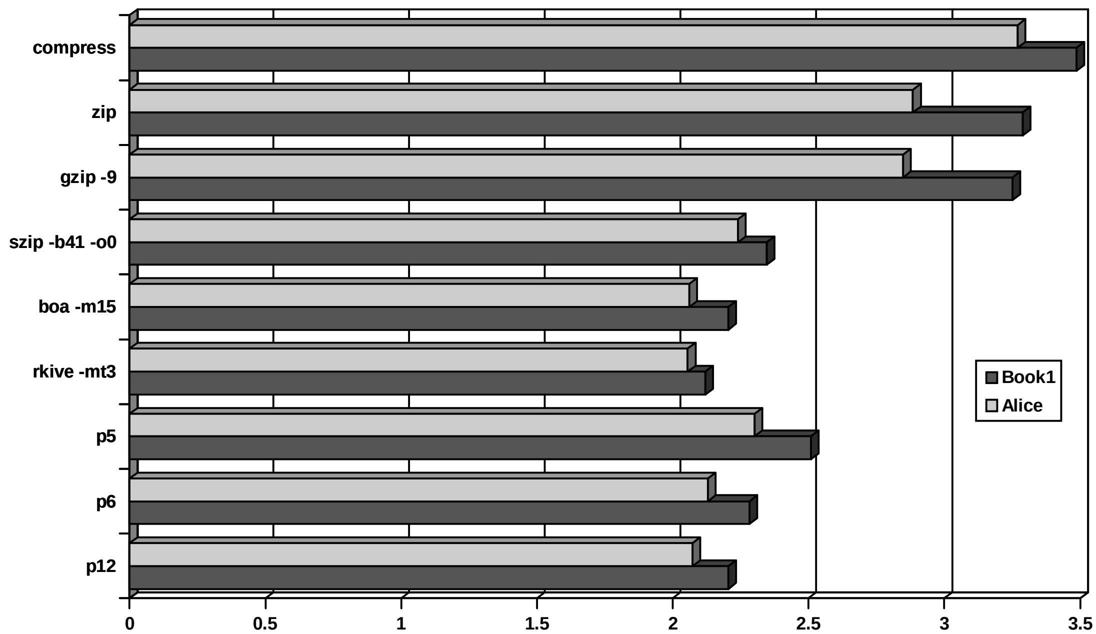
$\sigma^2{}_i = (N_i(0) + N_i(1) + 2d)/(N_i(0) + d)(N_i(1) + d)$
Variance of data in context i

$d = 0.5$ *Initial count*

$\eta_S = 0$ to 0.2 *Short term learning rate*

$\eta_L = 0.2$ to 0.5 *Long term learning rate*

Compression Results

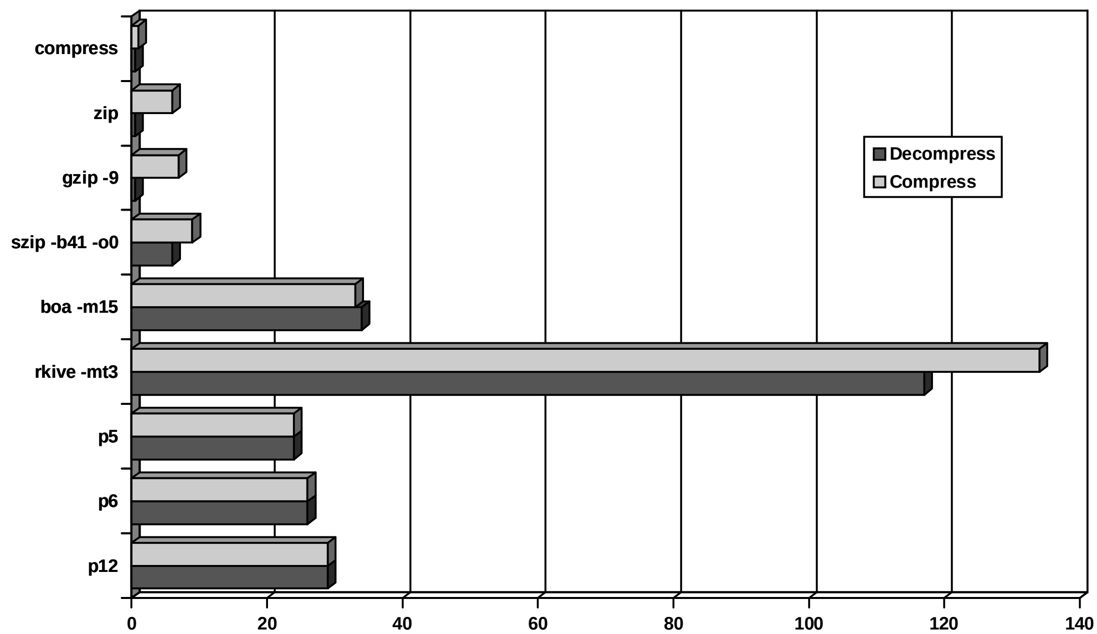


Compression in bits per character

- η_S and η_L tuned on *Alice in Wonderland*
- Tested on *book1* (Far from the Madding Crowd)

- P5 - 256K neurons, contexts of 1-4 characters
- P6 - 4M neurons, contexts of 1-5 characters
- P12 - 4M neurons, contexts of 1-4 characters and 1-2 words (unpublished)

Compression Time



Seconds to compress and decompress *Alice*
(152KB file on 100 MHz 486)

Summary

Compression within 2% of best known, at similar speeds

50% better (but 4x-50x slower) than *compress*, *zip*, *gzip*

Fast because

- Fixed representation - only output layer is trained (5x faster)
- One pass training by variable learning rate (25x faster)
- Bit-level prediction (16x faster)
- Sparse input activation (5-6 of 4 million, 80x faster)

Implementation available at
http://cs.fit.edu/~mmahoney/compression/