

ESE250: Digital Audio Basics

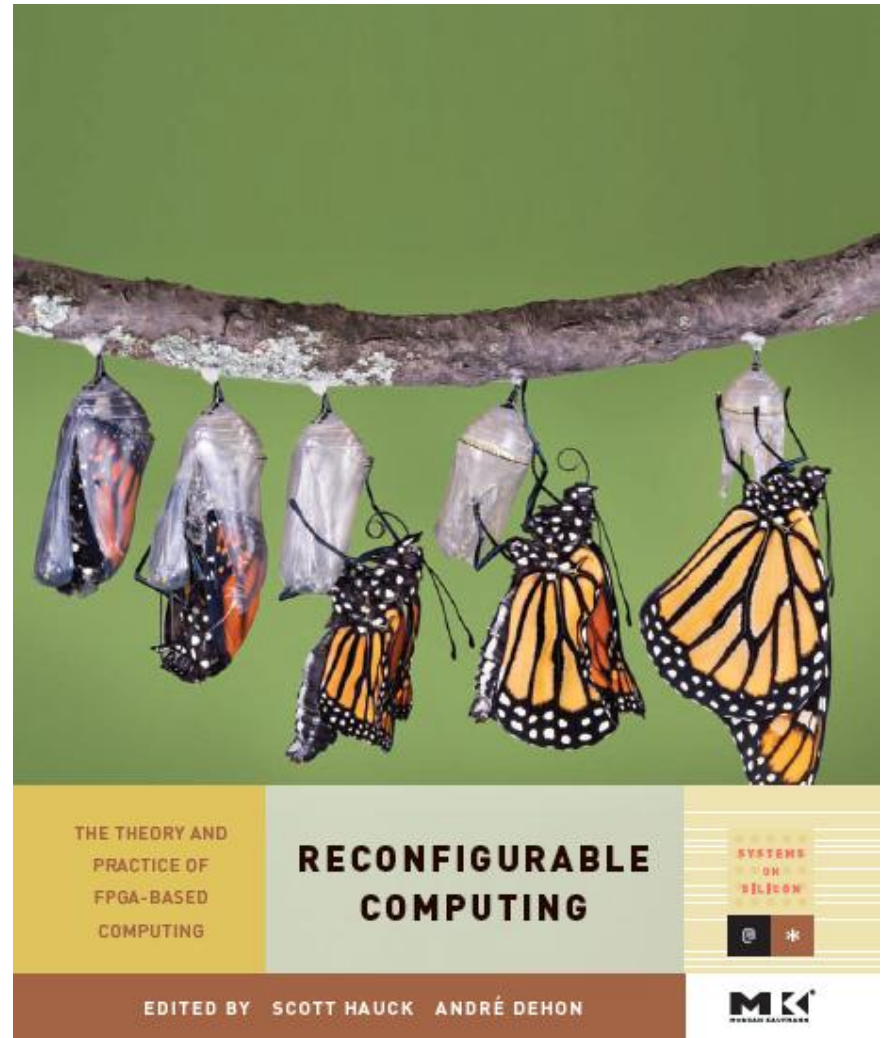
Week 3: January 26, 2012
Lossless Compression

Note: Lecture Worksheet



DeHon's Book

- 900+ pages
- 1.5 inches thick
- 3.7 lbs.



Kindle 2

- Claims 1,500 books
- 0.33 inches thick
- 0.6 lbs (10.2 oz)
- 2GB Flash
- 1.4GB user content

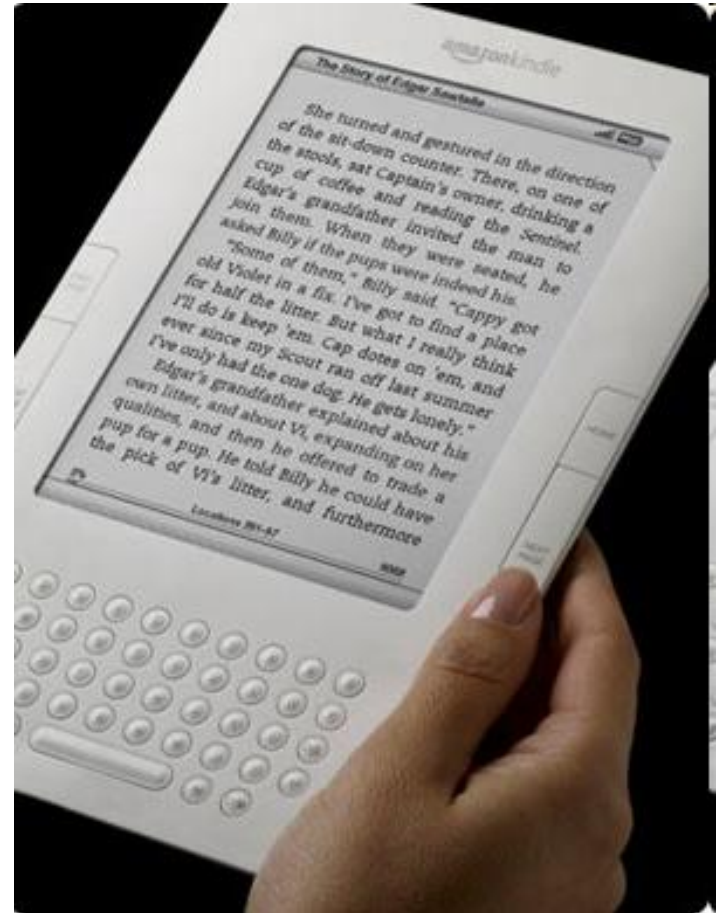


Image Source: http://www.amazon.com/Kindle-Amazons-Wireless-Reading-Generation/dp/B00154JDAI/ref=sr_1_1?ie=UTF8&s=electronics&qid=1253891340&sr=8-1

Evaluate Claim

- 1.4GB (let's approximate 1.5GB)
- Holds 1500 books
- How much “information” in a book?

$$1.5\text{GB} = 1500 \text{ MB}$$

$$1500\text{MB} / 1500 = 1\text{MB}/\text{book}$$

Evaluate Claim

- 1MB per book
- Assume 500p book
- How many Bytes per page?

(1MB = 1000 KB)

1000KB / 500 pages = 2KB/page

How much information on a page?

- Assume
 - 1 screen = 1 page
 - (May be poor assumption)
 - 1 bit / pixel (black and white)
 - 600x800 screen
- How many Bytes to represent?

$$600 \times 800 \text{ px/pg} \cdot 1/8 \text{ Byte/px} = 60,000 \text{ Byte/pg} \\ = 60 \text{ KB/pg}$$

- How compare with previous claim (2KB/pg)?

$$60\text{KB} / 2\text{KB} \Rightarrow 30 \text{ times larger!}$$

What do we do?

- How do we narrow 30× gap?

Structure

- Book doesn't contain arbitrary patterns of pixels
- There is structure
- Most pages are text
 - A limited set of pixel patterns

Exploit Structure

- How do we exploit this structure?

Page Structure

- Organized into lines of text
 - 80 characters / line
 - 50 lines / page
 - Limited number of characters
 - 26 letters, 9 digits, ...
- Assume 8b/character
 - How many bytes/page?

80 chars/line × 50 lines/page = 4000 chars/page

8bits/char = 1 Byte/char

4000 chars/page × 1 Byte/char = 4000 Bytes/page

Structure Saved Space

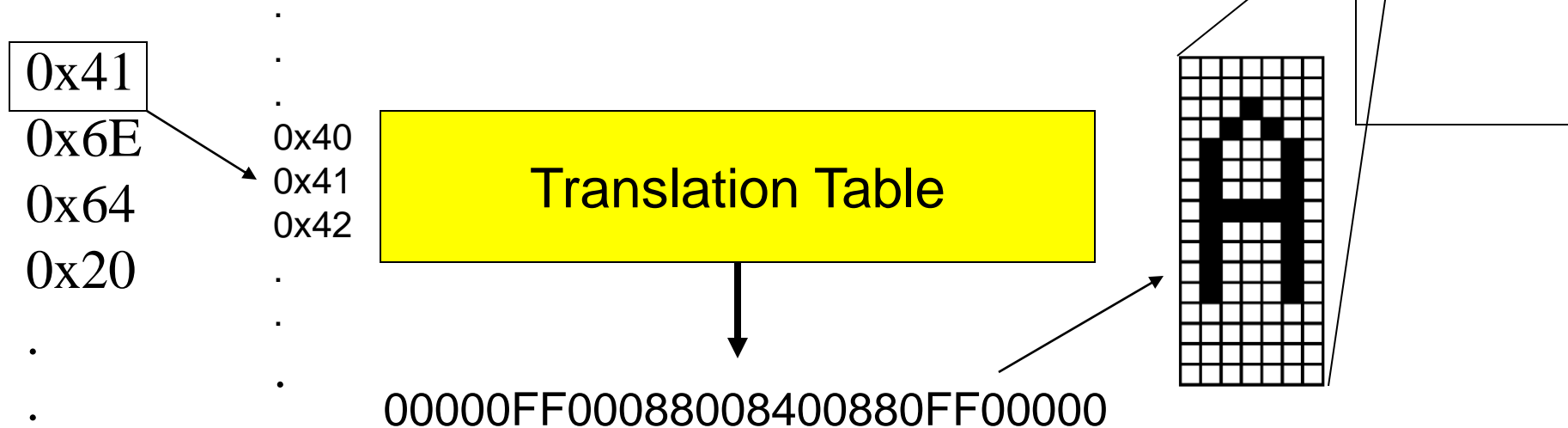
600 pixels/80 chars wide = 7 pixels/char

800 pixels/50 chars tall = 16 pixels/char

- 600x800 pixels to represent 80x50 characters
- How big is each character?
- $7 \times 16 \text{b} = 14\text{B}$ $7 \times 16 \text{ bits} = 7 \times 2 \text{ Bytes} = 14 \text{ Bytes}$
- But we only need 256 different characters
 - 1B worth of distinct “names”
 - 14B worth of px per pattern to represent with those names
- How resolve?
 - Give each desired character pattern a short name (1B long)
 - Use to lookup the 14B bit pattern

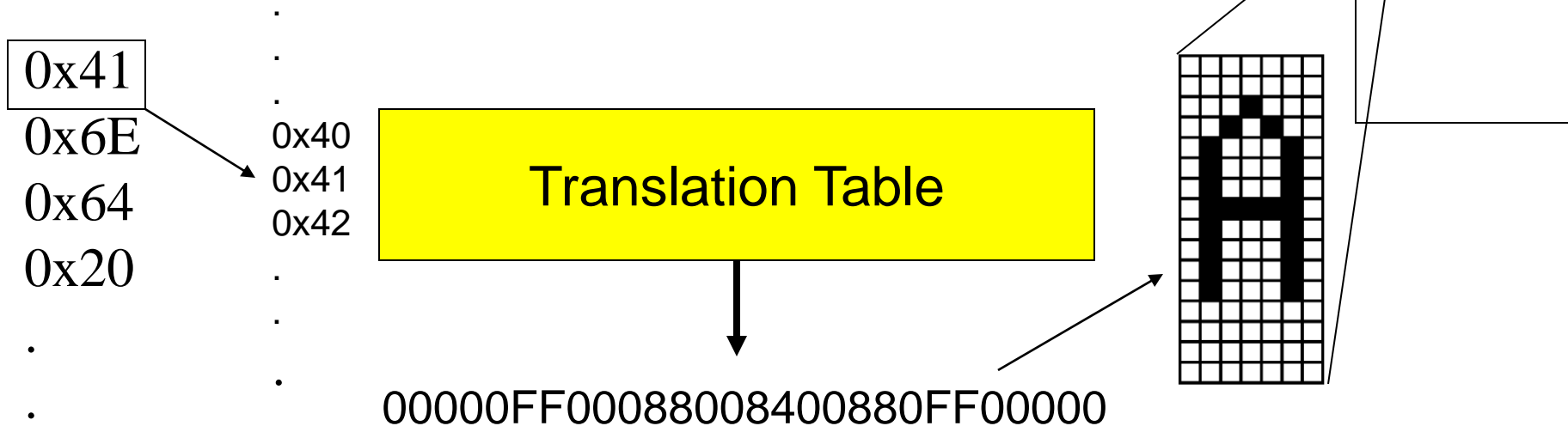
Decoding

- Give each character pattern a short name (1B long)
- Use to lookup the 14B bit pattern



Decoding Table

- 256 characters, 14B bit pattern
- How big is table? $256 \times 14B = 1024 \times 3.5 B = 3.5KB$
- Compared to page? $4KB$ with this encoding



Character Encoding

- Why 8b/character?
 - ASCII = American Standard Code for Information Interchange
 - Only 95 are printable
 - How many bits does that require? $\lceil \log_2(95) \rceil = 7$
 - Now how many Bytes/page?
[recall 4000 char/page (from slide 10)]
- $(4000 \text{ chars/page} \times 7 \text{ bits/char}) / (8 \text{ bits/Byte}) \approx 3.5 \text{ KB/page}$

Bits/character

- How many bits / character would we need to get to 2KB / page?

- Still 80 lines x 50 characters / line

2KB/page = 2000 Bytes / page

$(2000 \text{ Bytes/page}) / (4000 \text{ char/page}) = 0.5 \text{ Bytes/char} = 4 \text{ bits/char}$

- How many characters does this give us?

$2^4 = 16$ characters

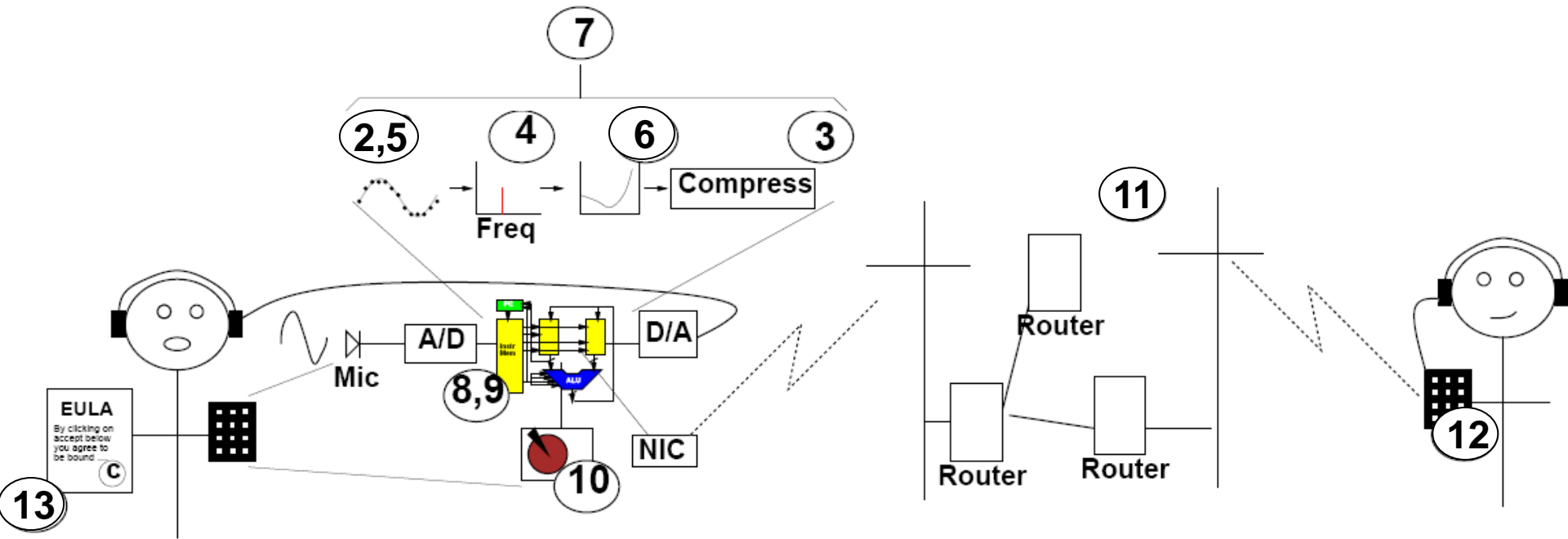
Claim

- We can encode English text with 4-5b/character on *average*.

Outline

- Translation: we can give
 - more frequent characters a shorter name
 - less frequent characters a longer name
- Exploiting Statistics
- Interlude
- Compressibility
- Entropy (getting formal)
- Larger building blocks

Course Map



Numbers correspond to course weeks

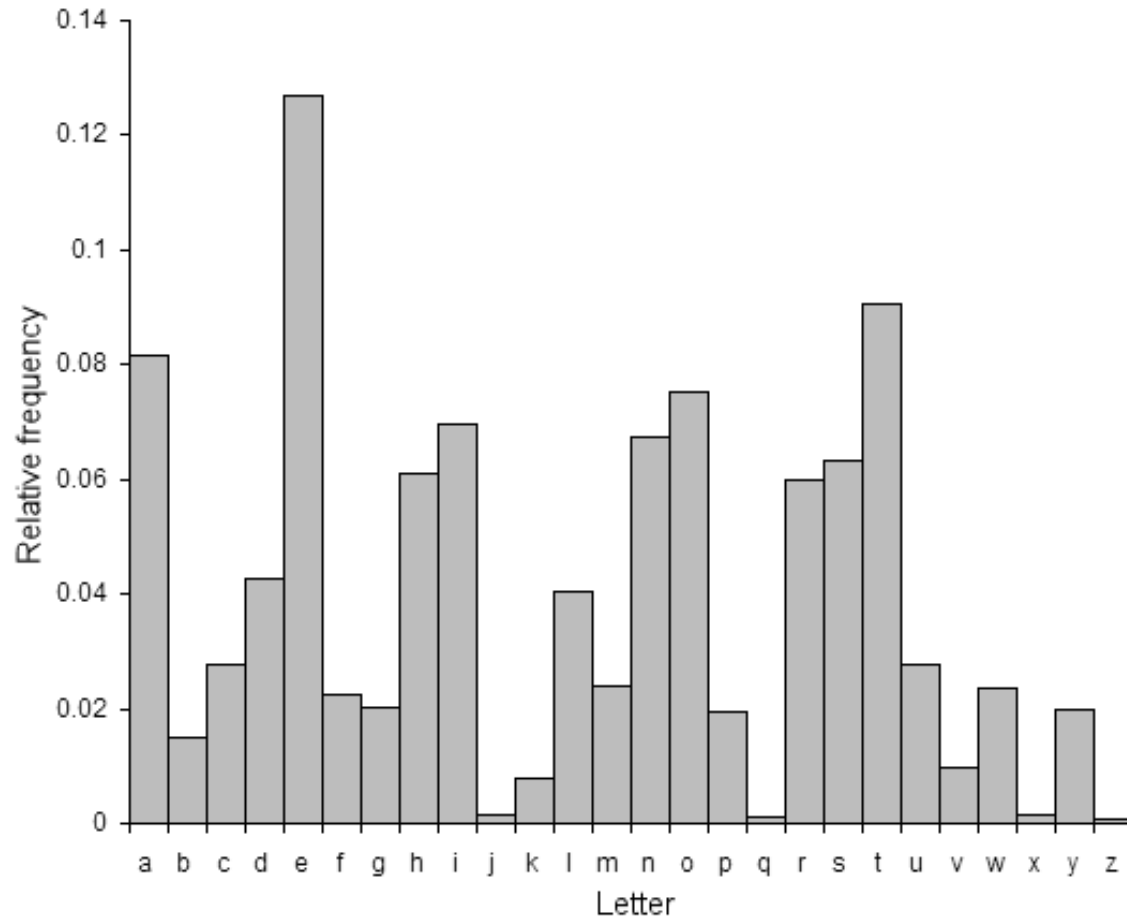
Information Content

- Does each character contain the same amount of “information”?

Statistics

- How often does each character occur?
 - Capital letters versus non-capitals?
 - How many e's in a line of text?
 - How many z's?
 - How many q's?
- Characters do not occur with equal frequency.

English Letter Frequency



<http://en.wikipedia.org/wiki/File:English-slf.png>

Exploiting Frequency

- How can we exploit statistics (frequency) to pick character encodings?

Encoding

symbol	bits	encode	symbol	bits	encode
spc	3	111	a	4	0111
,	8	01101010	b	6	000101
,	6	101000	c	6	110001
-	9	101001110	d	5	10011
.	7	0110001	e	3	001
1	10	0001001111	f	6	110000
4	13	1010011110110	g	6	011011
7	11	01101001001	h	5	11001
9	13	0001001011101	i	4	0100
;	9	011010111	j	10	0110100110
E	10	0110000011	k	8	10100110
F	10	0001001001	l	5	00011
I	7	0001000	m	6	110101
J	12	011010011100	n	4	0101
M	10	0110100001	o	4	1000
P	10	0110100101	p	6	100100
R	11	10100111100	q	10	0110100010
T	9	011000011	r	4	0000
W	11	10100111110	s	5	11011
Y	12	011010011110	t	4	1011
			u	6	110100
			v	7	1010010
			w	6	100101
			x	9	011000000
			y	6	011001

Calculating Bits/Page

- No longer: 80 lines/page
 ×50 characters/line
 ×7 bits/character

$$TotalBits = \sum occurrences[c] \times bits[c]$$

Line Example

Line 1:

Peace being concluded, and the association business therefore at an

Line Example

Line 1:

Peace being concluded, and the association business therefore at an

Peace being concluded 21

Line Example

Line 1:

Peace being concluded, and the association business therefore at an

Peace being concluded 21

1

0

Line Example

Line 1:

Peace being concluded, and the association business therefore at an

Peace being concluded 21

1

03

Line Example

Line 1:

Peace being concluded, and the association business therefore at an

Peace being concluded 21

1

034633634463644656535

Line Example

Line 1:

Peace being concluded, and the association business therefore at an

Peace being concluded 21

1

034633634463644656535 99

$$\text{Bits/char} = 99/21 = 4.71$$

Excerpt

- 2137 characters
- 9589 bits encoded
- 4.5 bits/character
- Compare to your per line calc.

symbol	bits	count	symbol	bits	count
cr	6	32	c	6	44
spc	3	336	d	5	71
'	8	3	e	3	186
,	6	41	f	6	42
-	9	3	g	6	39
.	7	9	h	5	78
1	10	1	i	4	130
4	13	1	j	10	2
7	11	1	k	8	6
9	13	1	l	5	49
;	9	5	m	6	36
E	10	1	n	4	133
F	10	1	o	4	134
I	7	13	p	6	40
J	12	1	q	10	1
M	10	2	r	4	88
P	10	4	s	5	111
R	11	1	t	4	163
T	9	5	u	6	61
W	11	1	v	7	12
Y	12	1	w	6	27
a	4	124	x	9	2
b	6	38	y	6	25

Idea

- Make encoding for common events (frequent characters) short
- Recurring System Engineering principal:
 - Make the common case fast

Interlude

Cryptograms

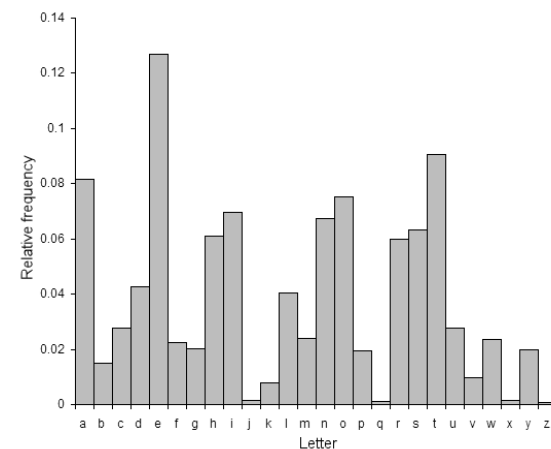
- Decipher:

Rit sdvry dmt ljagzhmrdjr. Ar'y cidr ritq
dmt ztmvtaftk rh pt ridr ktrtmgajty rit
vhlmyt hs tftjry.

- Knowing
 - English phrase, substitution code

<http://www.rinkworks.com/brainfood/p/crypts1.shtml>

Decipher Cryptogram



- Rit sdvry dmt ljagzhmrdjr. Ar'y cidr ritq dmt ztmvtaftk rh pt ridr ktrtmgajty rit vhlmyt hs tftjry.
- Letter frequency
 - t r m d y l
- English letter frequency
 - e t a o i n s r h l d c u m f p g w y b v k x j q z

Decipher Cryptogram

- Rit sdvry dmt ljagzhmrdjr. Ar'y cidr ritq dmt ztmvtaftk rh pt ridr ktrtmgajty rit vhlmyt hs tftjry.
- Guess: Rit = The
- THE sdvTy dmE ljagzhmTdjT. aT'y cHdT THEq dmE zEmvEafEk Th pE THdT kETEmgajEy THE vhlmyE hs EfEjTy.

Decipher Cryptogram

- THE sdvTy dmE ljagzhmTdjT. aT'y cHdT
THEq dmE zEmvEafEk Th pE **THdT**
kETEmgajEy THE vhlmyE hs EfEjTy.
- Needs a vowel – what works?
 - d=A
- THE sAvTy AmE ljagzhmTAjT. aT'y cHAT
THEq AmE zEmvEafEk Th pE THAT
kETEmgajEy THE vhlmyE hs EfEjTy.

Decipher Cryptogram

- THE sAvTy AmE ljagzhmTAjT. aT'y
cHAT THEq AmE zEmvEafEk Th pE
THAT kETEmgajEy THE vhlmyE hs
EfEjTy.
- 4 words end in y
- Half of all words end in: E T D or S
 - Used T, E
- Which makes sense here?

Decipher Cryptogram

- THE sAvTS AmE ljagzhmTAjT. aT'S cHAT
THEq AmE zEmvEafEk Th pE THAT
kETEmgajES THE vhlmSE hs EfEjTS.
- What makes sense in these context?
 - a=l, c=W, h=O
- THE sAvTS AmE ljlgzOmTAjT. IT'S WHAT
THEq AmE zEmvElfEk TO pE THAT
kETEmgljES THE vOlmsE Os EfEjTS.

Decipher Cryptogram

- THE sAvTS **AmE** ljlgzOmTAjT. IT'S WHAT THEq **AmE** zEmvElfEk TO pE THAT kETEmgljES THE vOImSE Os EfEjTS.
- What makes sense here?
 - $m=R$
- THE sAvTS ARE ljlgzORTAjT. IT'S WHAT THEq ARE zERvElfEk TO pE THAT kETERgljES THE vOIRSE Os EfEjTS.

Decipher Cryptogram

- THE sAvTS ARE ljlgzORTAjT. IT'S WHAT **THEq** ARE zERvElfEk TO **pE** THAT kETERgljES THE vOIRSE Os EfEjTS.
- Again, context limits
 - q=Y, p=B
- THE sAvTS ARE ljlgzORTAjT. IT'S WHAT THEY ARE zERvElfEk TO BE THAT kETERgljES THE vOIRSE Os EfEjTS.

Decipher Cryptogram

- THE sAvTS ARE ljlgzORTAjT. IT'S WHAT THEY ARE zERvEIfEk TO BE THAT kETERgljES THE vOIRSE **Os** EfEjTS.
- Most common 2 letter words:
 - **of** to in it is be as at so we he by **or on** do if me my up an go no us am
- s=F
- THE FAvTS ARE ljlgzORTAjT. IT'S WHAT THEY ARE zERvEIfEk TO BE THAT kETERgljES THE vOIRSE OF EfEjTS.

Decipher Cryptogram

- THE **FAvTS** ARE ljlgzORTAjT. IT'S WHAT THEY ARE zERvEIfEk TO BE THAT kETERgljES THE vOIRSE OF EfEjTS.
- What makes sense in this context?
 - $v=C$
- THE FACTS ARE ljlgzORTAjT. IT'S WHAT THEY ARE zERCEIfEk TO BE THAT kETERgljES THE COIRSE OF EfEjTS.

Decipher Cryptogram

- THE FACTS ARE UNIMPORTANT. IT'S WHAT THEY ARE PERCEIVED TO BE THAT DETERMINES THE COURSE OF EVENTS.

Cryptogram Lesson?

- Frequency information sufficient?
- Gives enough information
 - Combined with the structure of English and context
- ...to determine the letter.

- Substitution Cypher not very secure.
 - Nonetheless, Ketterer combo on your worksheet

Returning to Compression

Lossless Compression

- We discard no information
- $\text{Decode}(\text{Encode}(\text{msg})) = \text{msg}$
- Contrast with: $n_q(t) = s(t) - q_L[s(t)]$
- Invertible
- Translation in Symbol space

Lossless vs. Lossy

- Lossless
 - Encode(The) → 01100001111001001
 - Decode(01100001111001001) → The
 - Perfect restoration possible
- Lossy: lose information
 - E.g. drop case in encoding
 - Encode(The) → 101111001001
 - Decode(101111001001) → the
 - E.g. quantize sound waveform to 3-bits/sample

Compressibility

Compressibility

- Compressibility depends on non-randomness (uniformity)
 - Structure
 - Non-uniformity

Uniform

- If every character occurred with the same frequency,
 - There's nothing to give the short encoding
 - For everything we give a short encoding,
 - Something else gets a longer encoding

Highly Non-Uniform

- Extreme case:
 - One character occurs 99.999% of the time
 - Everything else less
 - Give it encoding of length 1 (say 0)
 - Everything else length 8 (1xxxxxxx)
 - Avg. encoding length
 - = $1 * 0.99999 + 0.00001 * 8$
 - = 1.00007

Notion

- Compare:
 - Uniform – 7 b/character (95 characters)
 - Previous slides – 1.00007 b/char
 - Previous calculation – 4.5 b/char
- The less uniformly random,
 - the more opportunity for compression

Formalizing Uniformity

Entropy

Formalizing Randomness

- We can quantify randomness
 - Measure structure
- Doesn't require selecting a code
- Forms a useful Lower Bound

Lower Bounds

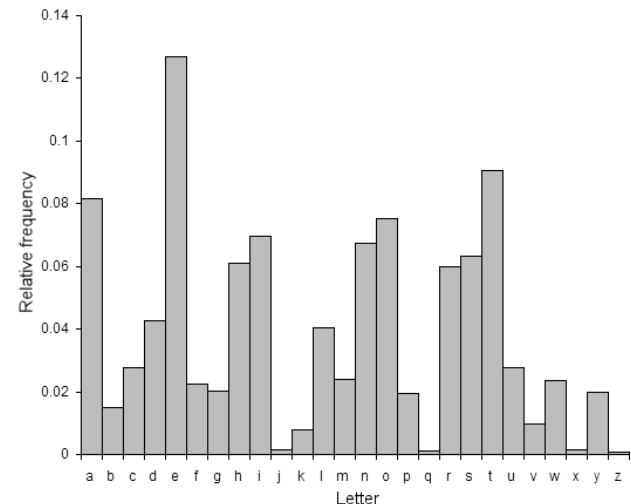
- Useful tool for Engineers
- Tell us when to stop optimizing
- How much headroom do we have to improve if we work harder?

Information / Symbol

- What is the “information” associated with each symbol?
- Related to its probability of occurrence.
- If everything equally likely,
 - The probabilities are all the same
 - All give same information

Information / Symbol

- What is the “information” associated with each symbol?
- Related to its probability of occurrence.
- $\text{Prob}('e') > \text{Prob}('z')$
 - Seeing an ‘e’ conveys less information than seeing a ‘z’



Information “Rate” of a Symbol

- Notion of information “rate”
 - “bits per character”
 - to be “expected”
- Depends on probability of occurrence
- Examples
 - Generic: $p_{sym} = 1/256 = 2^{-8}$
 $\Rightarrow I_{sym} = \log_2 (1/p_{sym}) = \log_2 (2^8) = 8$
 - Letter “e”: $p_e = 0.127$ ($\approx 1/8 = 2^{-3}$)
 $\Rightarrow I_e = \log_2 (1/p_e) = 2.98$ ($\approx \log_2 (2^3) = 3$)
- In general $I_{sym} = \log_2 (1/p_{sym})$

Information “Rate” of a Message

- Total Information content

if sym occurs c_{sym} times in Msg then

$$\begin{aligned} I_{Msg} &= \sum_{sym \in Msg} c_{sym} \times I_{sym} \\ &= \sum_{sym \in Msg} c_{sym} \times \log_2(1/p_{sym}) \end{aligned}$$

- Average information rate

if total of symbols in Msg is $C_{Msg} = \sum_{sym \in Msg} c_{sym}$ then

$$\begin{aligned} I_{Msg} / C_{Msg} &= \left(\sum_{sym \in Msg} c_{sym} \times I_{sym} \right) / C_{Msg} \\ &= \sum_{sym \in Msg} \left(c_{sym} / C_{Msg} \right) \times \log_2(1/p_{sym}) \end{aligned}$$

Entropy

- As Msg becomes very long we expect that

$$\text{Lim}_{Msg \rightarrow \infty} (c_{sym} / C_{Msg}) = p_{sym}$$

- Leads to new definition

$$\textit{Entropy}(Msg) = \sum \left(p_i \times \log_2 \left(\frac{1}{p_i} \right) \right)$$

- Tells us how random the information is
 - E.g., how far we are from a uniform distribution

Uniform distribution ($p=1/256$) \rightarrow reduce to TotalChars \times 8

Smaller values \rightarrow less random \rightarrow more compressible

Comparing Entropy with Table

Franklin excerpt

- 2137 characters
- Entropy=9434
 - 4.41 b/char
- codedbits=9589
 - 4.49 b/char

sym	C-bits	Inform.
spc	3	2.6
e	3	3.5
t	4	3.7
o	4	4.0
n	4	4.0
i	4	4.0
a	4	4.1
q	10	11.06

Questions Left Open

- How prove this is the lower bound?
- How construct codes that come close to this?
 - Prove they are optimal?
- How close can we get?

[Have answers, but beyond today's lecture]

Larger Building Blocks

Previous Model Limitations

- Previous discussion ignores symbol context
- Simplified model
 - Assumes uniform probability of symbol occurrence in all contexts
 - Assumes must have code for single symbol
 - In fact “optimum” and “lower bound” are defined assuming that model

Context

- Probability of a symbol depends on where it shows up.
 - What's the probability of second/missing letter?

t_e

Predecessor Context

- Simple model:
 - Depends on previous letter.
 - What's the probability of symbols given immediate prior symbol?
 - Probability of symbols following:
 - Q
 - Period
 - e

Letter Frequencies

- in the English Language
 - e t a o i n s r h l d c u m f p g w y b v k x j q z
- Letters to Follow the "e"
 - r s n d

http://www.deafandblind.com/word_frequency.htm

Predecessor Context

- Use $P(c_i|c_{i-1})$ instead of $P(c_i)$
 - When computing entropy
- Use a separate Encoding table for each context
 - $\text{Encode}_c(x)$

Position in Word Context

Letter Frequencies

- English Language
 - e t a o i n s r h l d c u m f p g w y b v k x j q z
- 1st Letter in Words
 - t o a w b c d s f m r h i y e g l n o u j k
- 2nd Letter in Words
 - h o e i a u n r t
- 3rd Letter in Words
 - e s a r n i
- Last Letter in Words
 - e s t d n r y f l o g h a k m p u w
- More than half of all words end with:
 - e t d s

http://www.deafandblind.com/word_frequency.htm

Repeated Phrase Model

- Often words/phrases are repeated
 - Consider lyrics for a song:
 - She loves you, yeah, yeah, yeah
 - She loves you, yeah, yeah, yeah
 - She loves you, yeah, yeah, yeah, yeah
- You think you lost your love...

["She Love You" – Beatles]

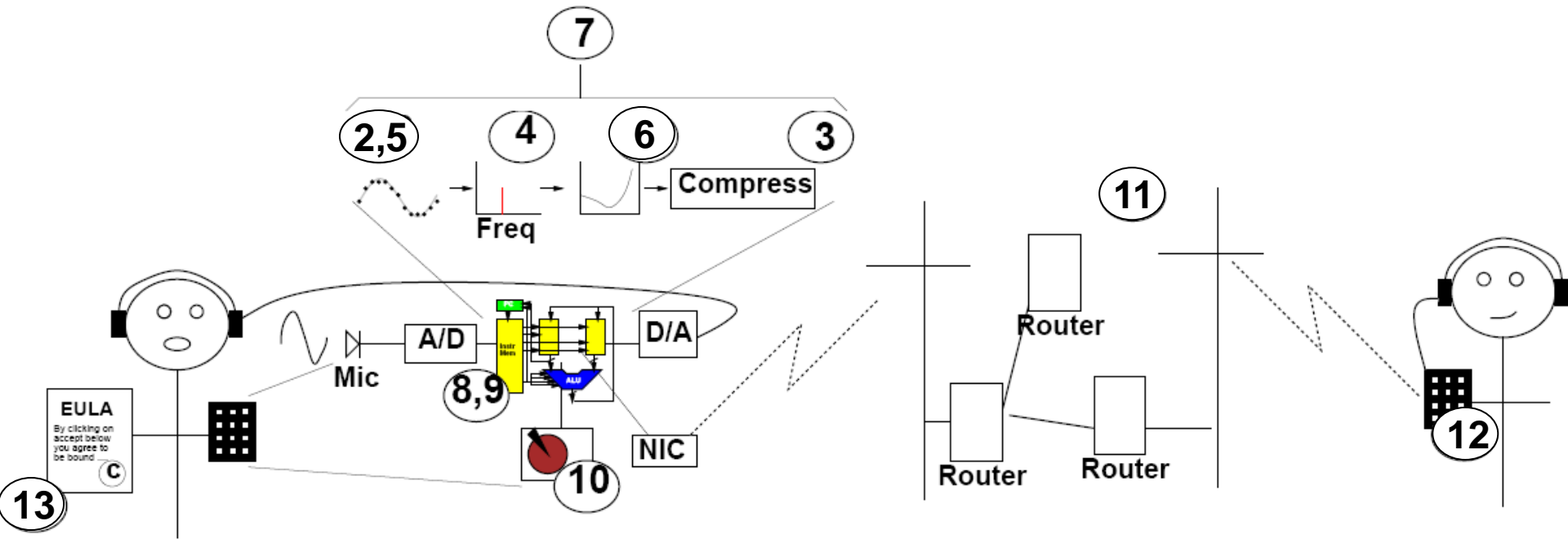
Repeated Phrase Model

- Often words/phrases are repeated
 - Consider lyrics for a song:
 - She loves you, yeah, yeah, yeah
<copy 0:31>
<copy 0:30><copy<25:31>
- You think <copy 10:12> lost <copy 10:12>r
<copy 4:7>...

[“She Love You” – Beatles]

Wrapup

Course Map



Numbers correspond to course weeks

Learn More

- Readings linked to today's lecture
 - One in blackboard
- ESE 674 – Information Theory

Admin

- Lab Tuesday
 - In Ketterer (Moore 204)

Big Ideas

- Translation – give things compact names
- Real world symbol sequences are **not** uniformly random
- Non-uniformity → compression opportunity
 - We can compress rw symbol seq. significantly
 - Exploit the common case, statistics
- Look for right granularity and structure