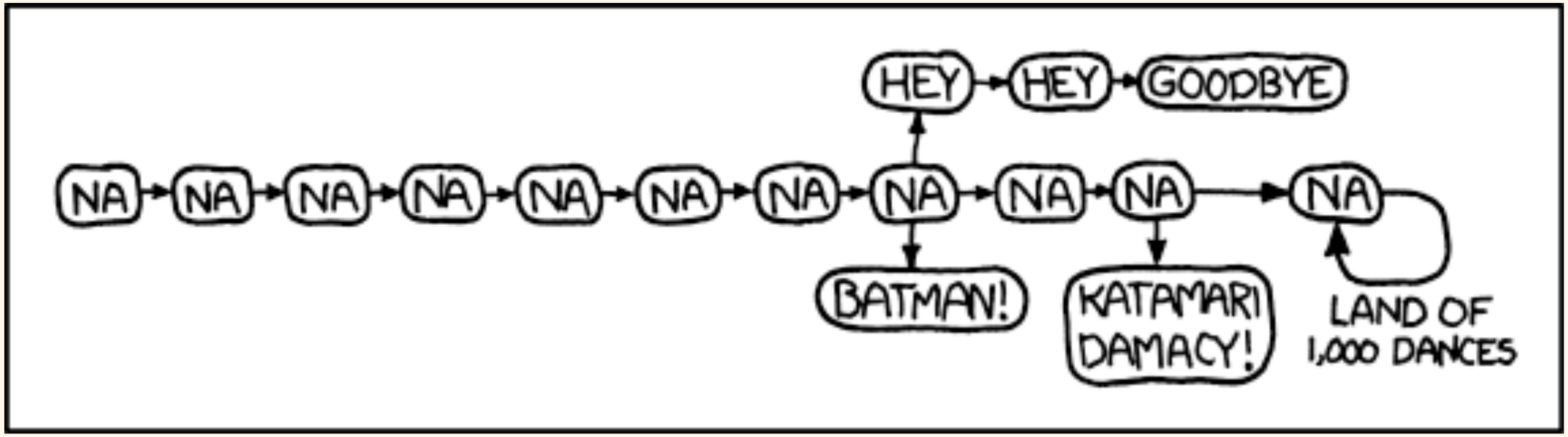
Analyzing Sequences:

Course overview, and some first steps



Steven Bedrick

CS/EE 5/655, 9/29/14

xkcd #851, "Na", by Randall Munroe

Plan for the day:

1. What are we doing here?
2. Course logistics
3. What is a sequence?
4. Probability refresher
5. Overview of encodings

What are doing here?

Many types of data occur sequentially:

Words in a sentence .

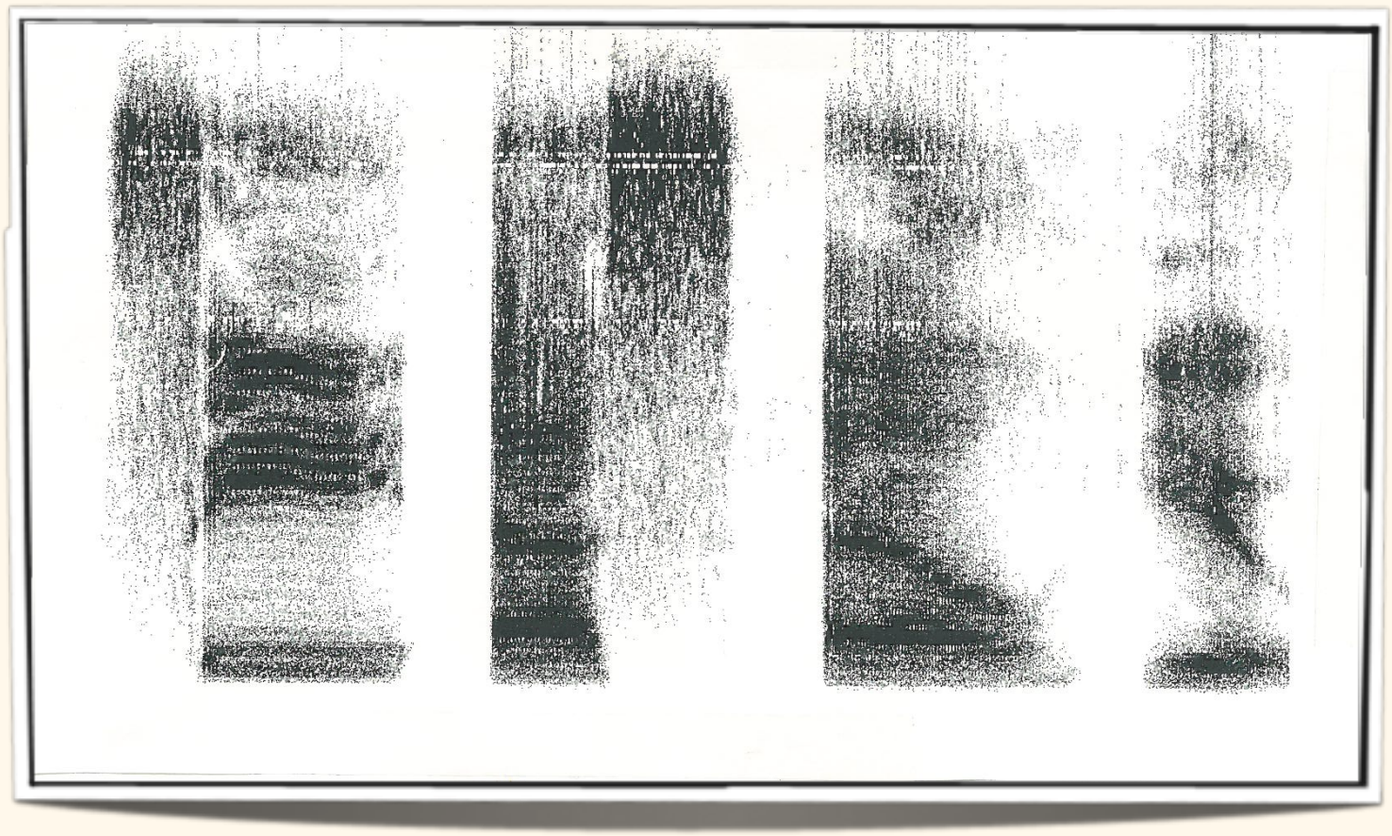
L e t t e r s in a word.

Turns

taken

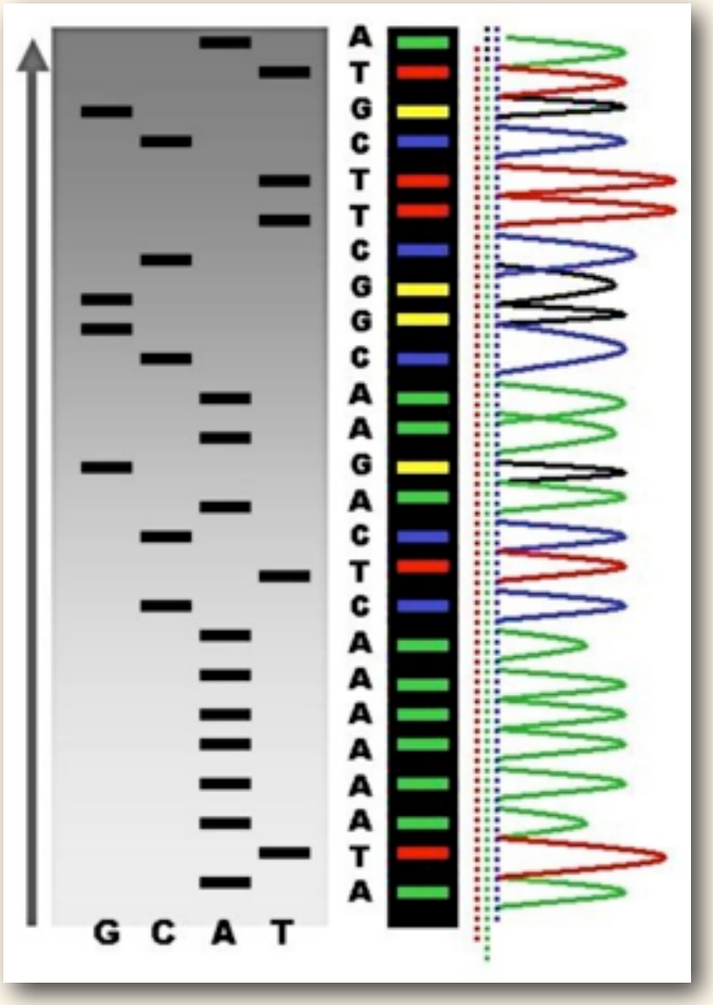
in a

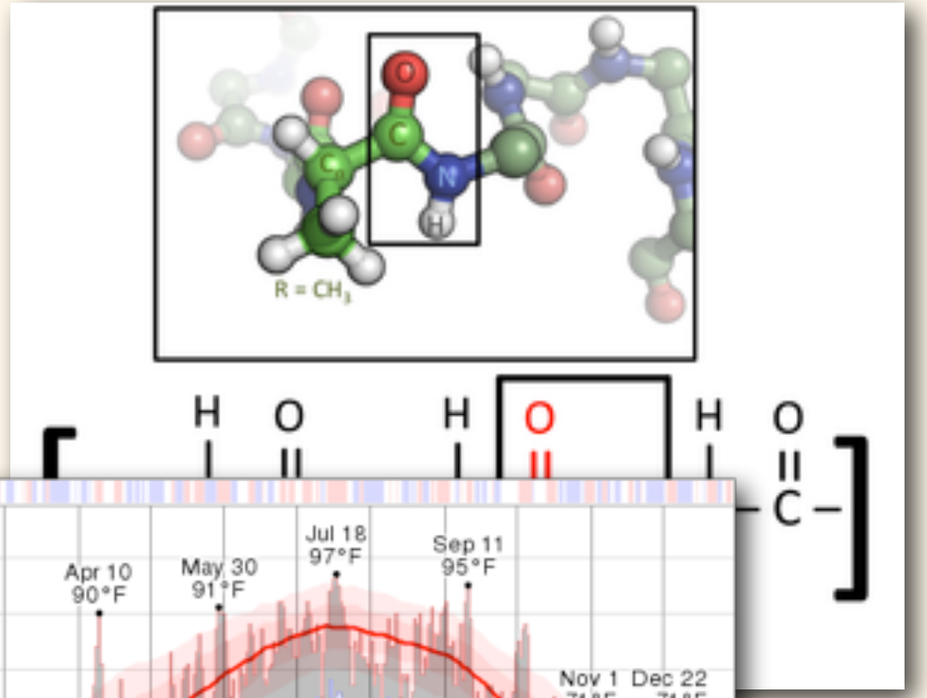
conversation.

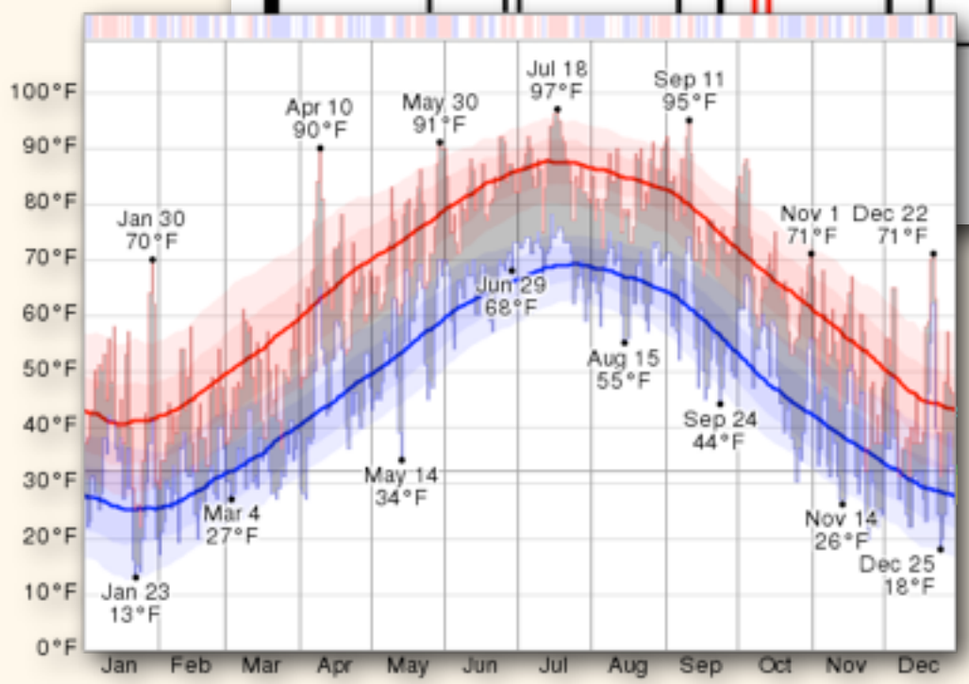


What are doing here?

Many types of data occur sequentially:







What are doing here?

Things we might want to do
with sequential data:

Characterize what we've seen...

Guess what might come next...

Label/Classify observations...

Compare (match?) two sequences...

Can you think of more?

Logistics:

Mondays & Wednesdays, 4:00 to 5:30 in GH5

Course Website:

http://cslu.ohsu.edu/~bedricks/courses/cs655/

Grading: 70% homework, 30% participation

Logistics:

Homework will be due at 11:59 PM the night of its due date.

If you think you will need more time for a given assignment, don’t be afraid to ask...

... but don’t wait until the last minute to do so!

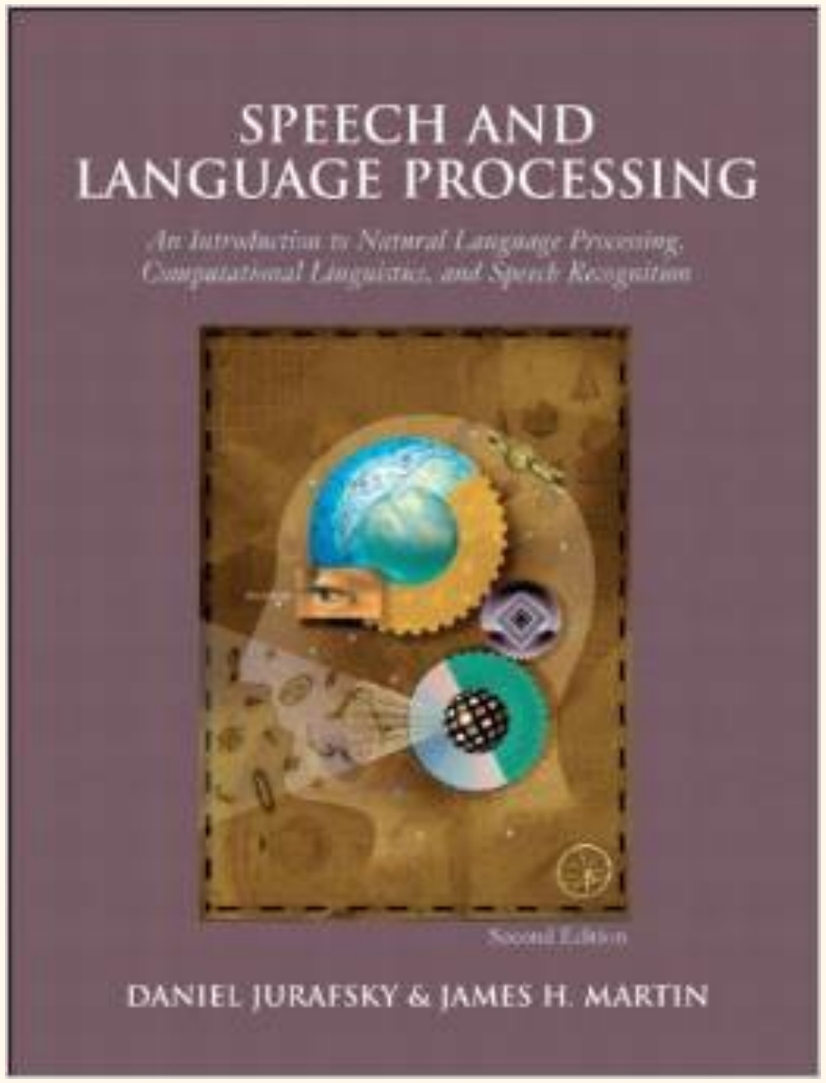
Logistics:

You'll need access to a computer running a modern UNIX-like OS...

Let me know ASAP if that will be a problem!

Logistics:

The textbook we'll be using:



Later in the course, I will provide other readings.

Plan for the day:

1. What are we doing here?
2. Course logistics
3. What is a sequence?
4. Probability refresher
5. Overview of encodings

What is a sequence?

Simply put, a sequence is an *ordered* list of (usually, related) elements.

A sequence of elements can be either finite (“h”,”e”,”l”,”l”,”o”) or infinite (“all even integers”)

Order matters!

{1,3,5} != {3,5,1}

What is a sequence?

Key vocabulary:

"Alphabet": The set of distinct elements ("types") that can occur in a given sequence

For DNA: "A","C","T","G"

For proteins: "Ala","Arg","Asn","Asp", "Cys", etc.

For (some) English: "A","B","C","D", etc.

For Jane Austen titles: "Sense", "and", "Sensibility", "Pride", "Prejudice", etc...

What is a sequence?

Key vocabulary:

“n-gram”: sub-sequences of n-adjacent tokens in a sequence

“It is a truth universally acknowledged…”

1-gram (unigram)	“It”, “is”, “a”, “truth”, “universally”, etc.
2-gram (bigram)	“It is”, “is a”, “a truth”, “truth universally”, etc.
3-gram (trigram)	“It is a”, “is a truth”, “a truth universally”, etc.

Plan for the day:

1. What are we doing here?
2. Course logistics
3. What is a sequence?
4. Probability refresher
5. Overview of encodings

Glossary:

- argmax
- Joint probability
- Conditional probability
- Statistical independence
- Bayes' theorem
- Noisy channel model

argmax:

$$\arg\max_x f(x)$$

The value(s) of x at which f(x) is at its maximum.

$$\arg\max_x \sin x = \frac{\pi}{2}$$

Not to be confused with max:

$$\max_x \sin x = 1$$

Joint probability:

The probability that two or more random variables will take on certain respective values.

$$p(red + blue = 11)$$

$$= p(red = 6) \times p(blue = 5) + p(red = 5) \times p(blue = 6)$$

$$= \frac{1}{6} \times \frac{1}{6} + \frac{1}{6} \times \frac{1}{6}$$

$$= \frac{1}{36} + \frac{1}{36} = \frac{1}{18}$$

Conditional probability:

The probability that one or more random variables have a respective value...

... given that one or more *other* random variables have a respective value.

$$p(A|B) = \frac{p(A \cap B)}{p(B)}$$

$$p(red + blue = 11 | red = 6) = \frac{(1/6)(1/6)}{1/6}$$

$$= \frac{1}{6}$$

Statistical Independence:

Random variables A and B are independent iff:

$$p(A|B) = p(A)$$

$$p(B|A) = p(B)$$

Bayes' Theorem

Used for lots of things, but particularly helpful when we need to relate prior information to current observations.

$$P(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

Thomas Bayes
1701(?) – 1761

Image: Wikipedia. As per statute, it is unlawful to discuss Bayes' theorem without showing this image.

Bayes' Theorem

Example: Testing for a rare mutation.

In a population of 1,000,000 people, 1 in 10,000 (0.0001) carry a certain mutation.

A test exists for this mutation, and the test is 95% accurate (FPR and FNR are both 0.05).

You test positive for the mutation. What are the chances that you *actually* carry it?

Bayes' Theorem

Example: Testing for a rare mutation.

A: "has mutation"
B: "tests positive"
We want to know P(A | B) (i.e., p(mutation | positive).

p(A) = 0.0001

p(B | A) = 0.95

p(B) = p(B|A)p(A) + p(B|!A)p(!A)

p(B) = 0.95*0.0001+0.5*0.9999

p(B) = 0.5009

Bayes' Theorem

p(A) = 0.0001

p(B | A) = 0.95

p(B) = p(B|A)p(A) + p(B|!A)p(!A)

p(B) = 0.95*0.0001+0.5*0.9999

p(B) = 0.5009

$$P(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

$$p(A|B) = \frac{0.95 \times 0.0001}{0.5009} \approx 0.0002$$

Conclusion: With a rare disease, you need a very accurate test!

“Noisy Channel” model

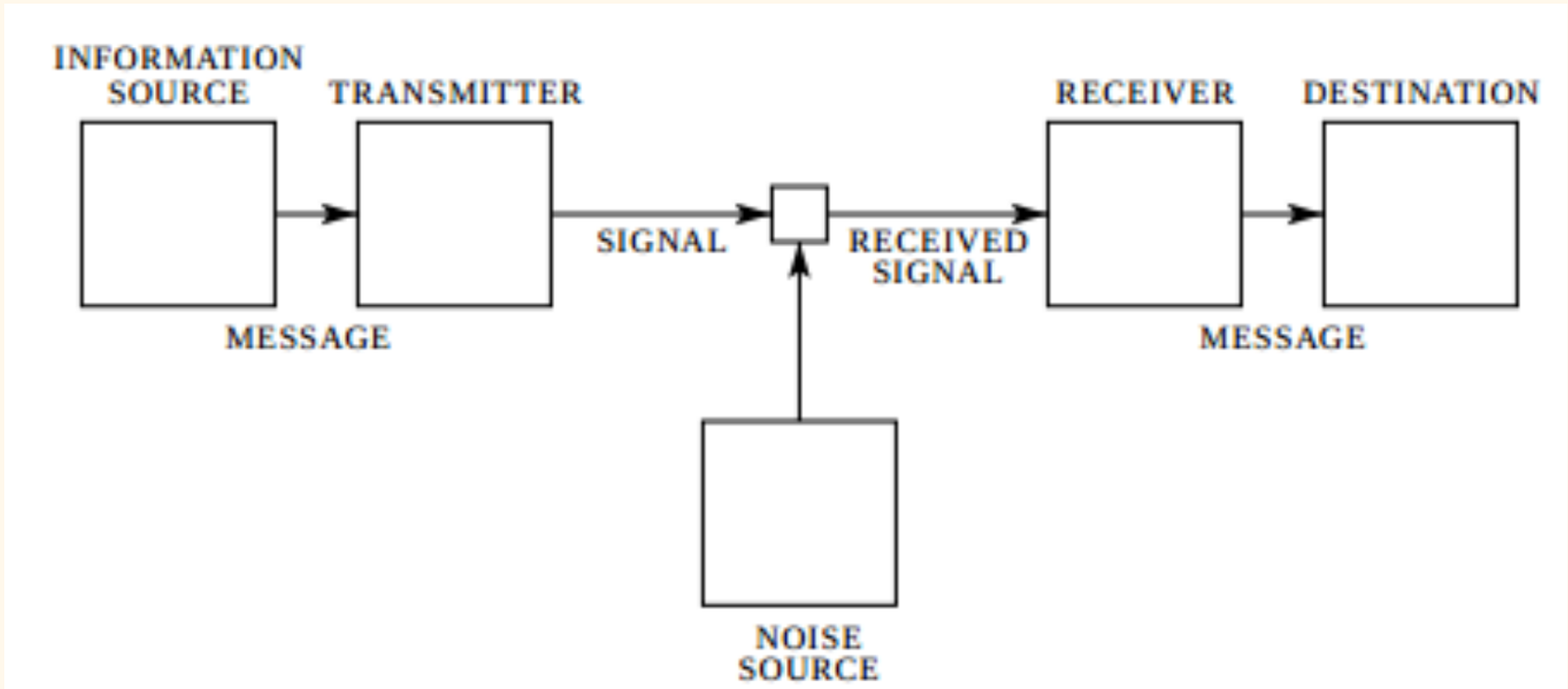


Fig. 1—Schematic diagram of a general communication system.

Claude Shannon
1916 – 2001

1. Shannon C. A Mathematical Model of Communication. The Bell System Technical Journal. 1948 Jul 28;27:379–423, 623–56.

“Noisy Channel” model

Warren Weaver, 1949 Rockefeller Foundation memorandum
Translation:

“When I look at an article in Russian, I say: this is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.”

Warren Weaver
1894 – 1978

"Noisy Channel" model

Given a ciphertext *ct*, select plaintext *pt* that maximizes the conditional probability of the plaintext:

$$\arg\max_{pt} p(pt|ct)$$

Using Bayes' Theorem:

$$\arg\max_{pt} \frac{p(ct|pt)}{p(ct)}$$

The denominator is constant, so:

$$\frac{p(ct|pt)}{p(ct)} \propto p(ct|pt)p(pt)$$

So at the end:

$$\arg\max_{pt} p(ct|pt)p(pt)$$

Plan for the day:

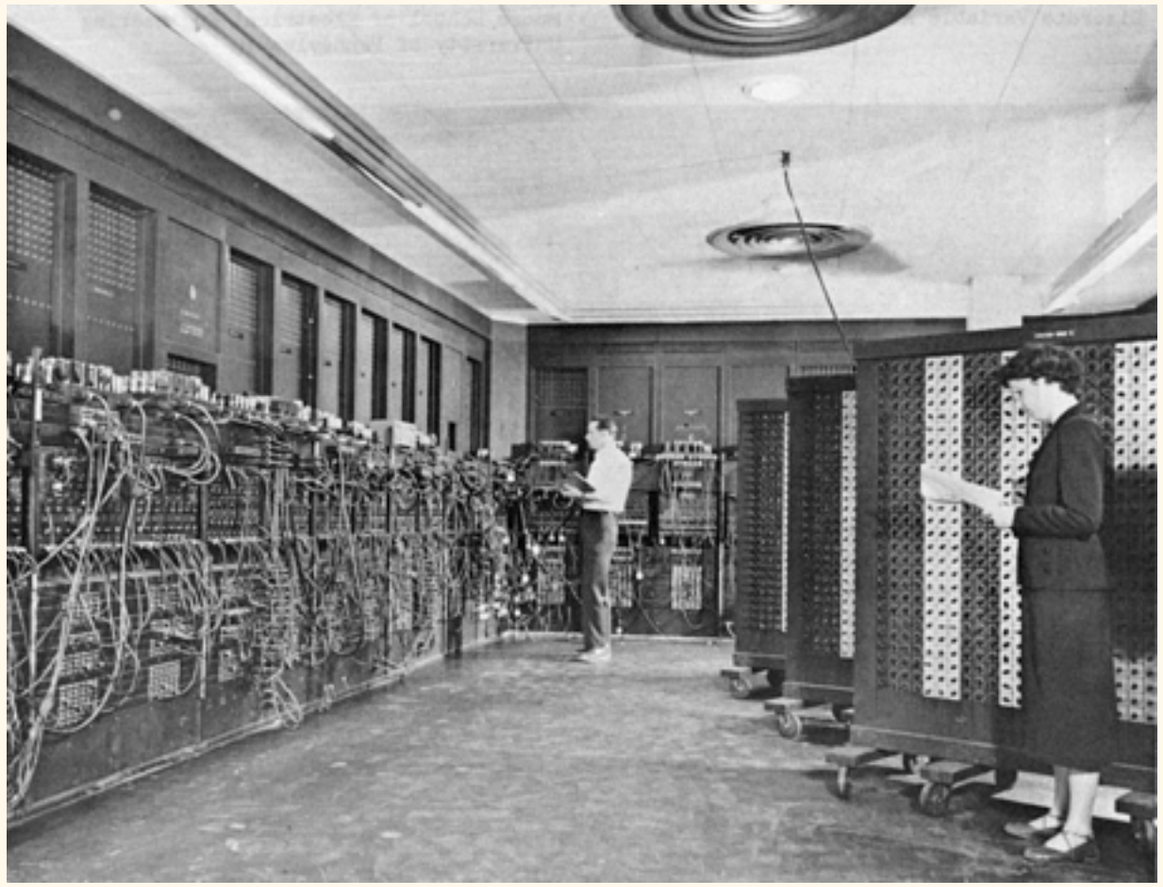
1. What are we doing here?
2. Course logistics
3. What is a sequence?
4. Probability refresher
5. Overview of encodings

Why talk about encodings?

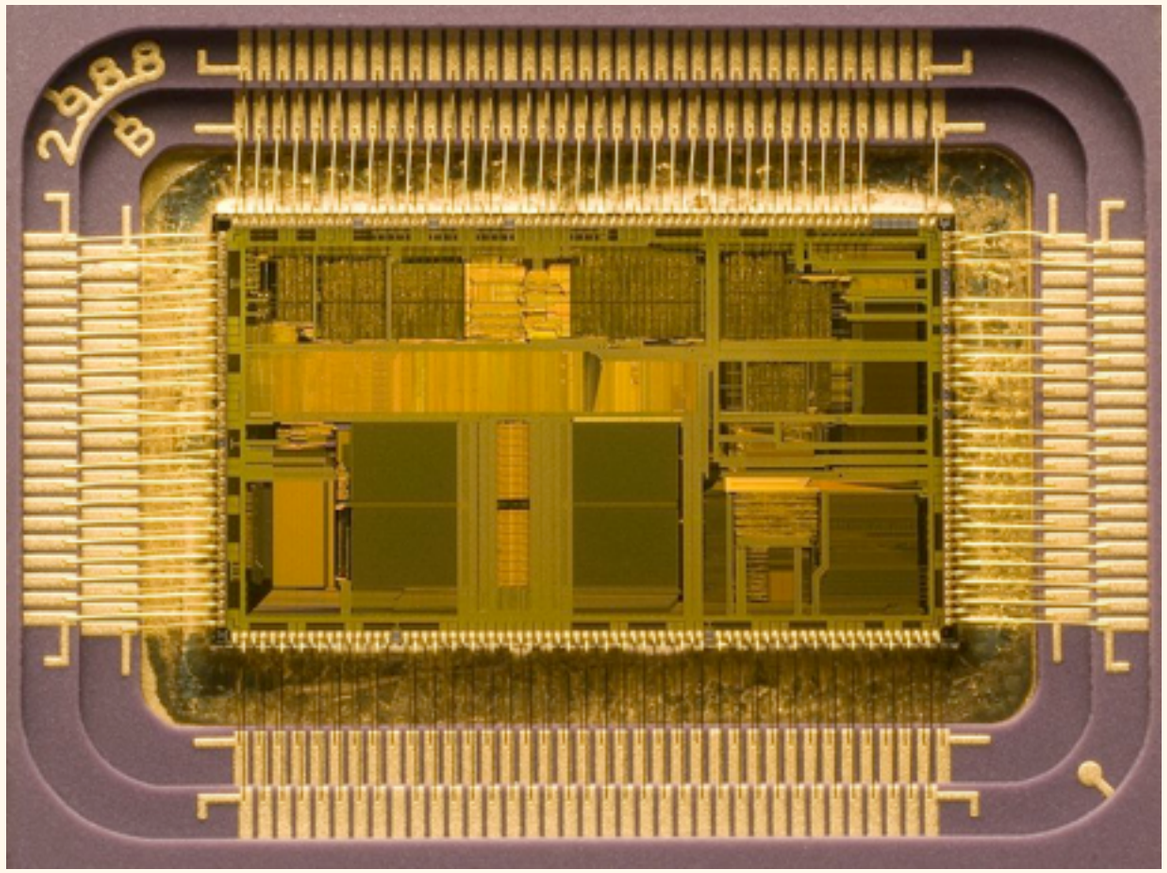
Sequential linguistic data (“text”) has to be represented on disk *somehow*...

... and all kinds of practical problems can be prevented by knowing how it’s done!

Computers only “know” about numbers....



No letters here...



None here, either...

... so how do they do text?

http://en.wikipedia.org/wiki/File:Eniac.jpg

http://en.wikipedia.org/wiki/File:80486dx2-large.jpg

Terminology:

Character

Surprisingly hard to define, but for now: A basic, atomic unit of written text.

Glyph

A specific visual "shape" used to represent a character.

Character Repertoire

A finite collection of abstract characters

Character Set

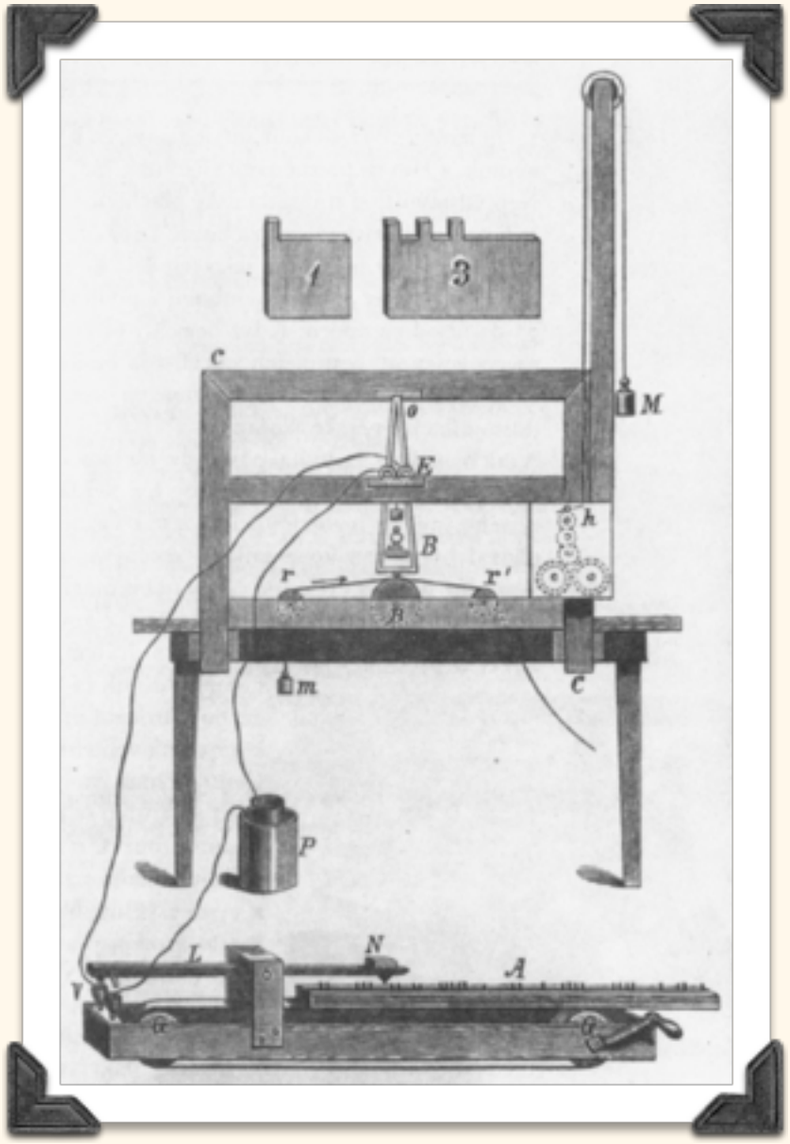
A character repertoire coupled with a systematic internal representation (i.e., some kind of organizational or numbering scheme)

Character Encoding System

An algorithm for representing a character set as a sequence of binary digits.

The story begins with Samuel Morse:*

Samuel Morse
1791–1872



International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A	·−	U	··−
B	−···	V	···−
C	−·−·	W	·−−
D	−··	X	−··−
E	·	Y	−·−−
F	··−·	Z	−−··
G	−−·		
H	····		
I	··		
J	·−−−		
K	−·−	1	·−−−−
L	·−··	2	··−−−
M	−−	3	···−−
N	−·	4	····−
O	−−−	5	·····
P	·−−·	6	−····
Q	−−·−	7	−−···
R	·−·	8	−−−··
S	···	9	−−−−·
T	−	0	−−−−−

It doesn’t, actually, but we have to start somewhere...

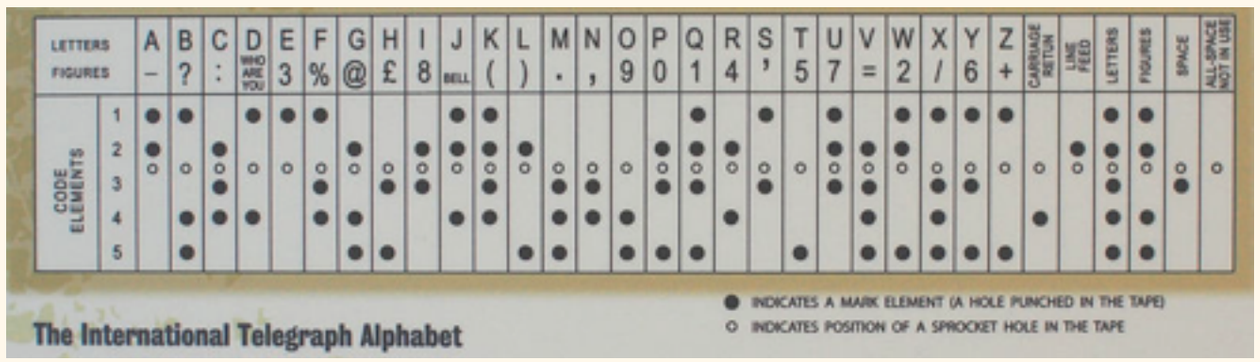
http://en.wikipedia.org/wiki/File:Samuel_Morse_1840.jpg

http://en.wikipedia.org/wiki/File:Morse_telegraph.jpg

http://en.wikipedia.org/wiki/File:International_Morse_Code.svg

The story continues with Émile Baudot, the inventor of a popular printing telegraph.

Émile Baudot
1845–1903



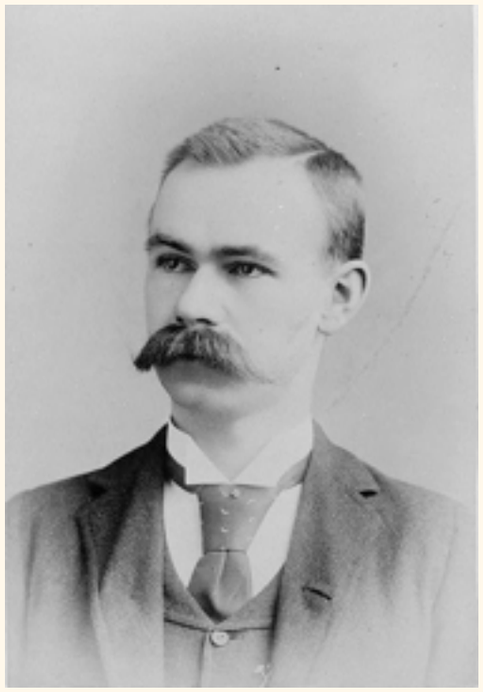
The International Telegraph Alphabet

http://en.wikipedia.org/wiki/Émile_Baudot

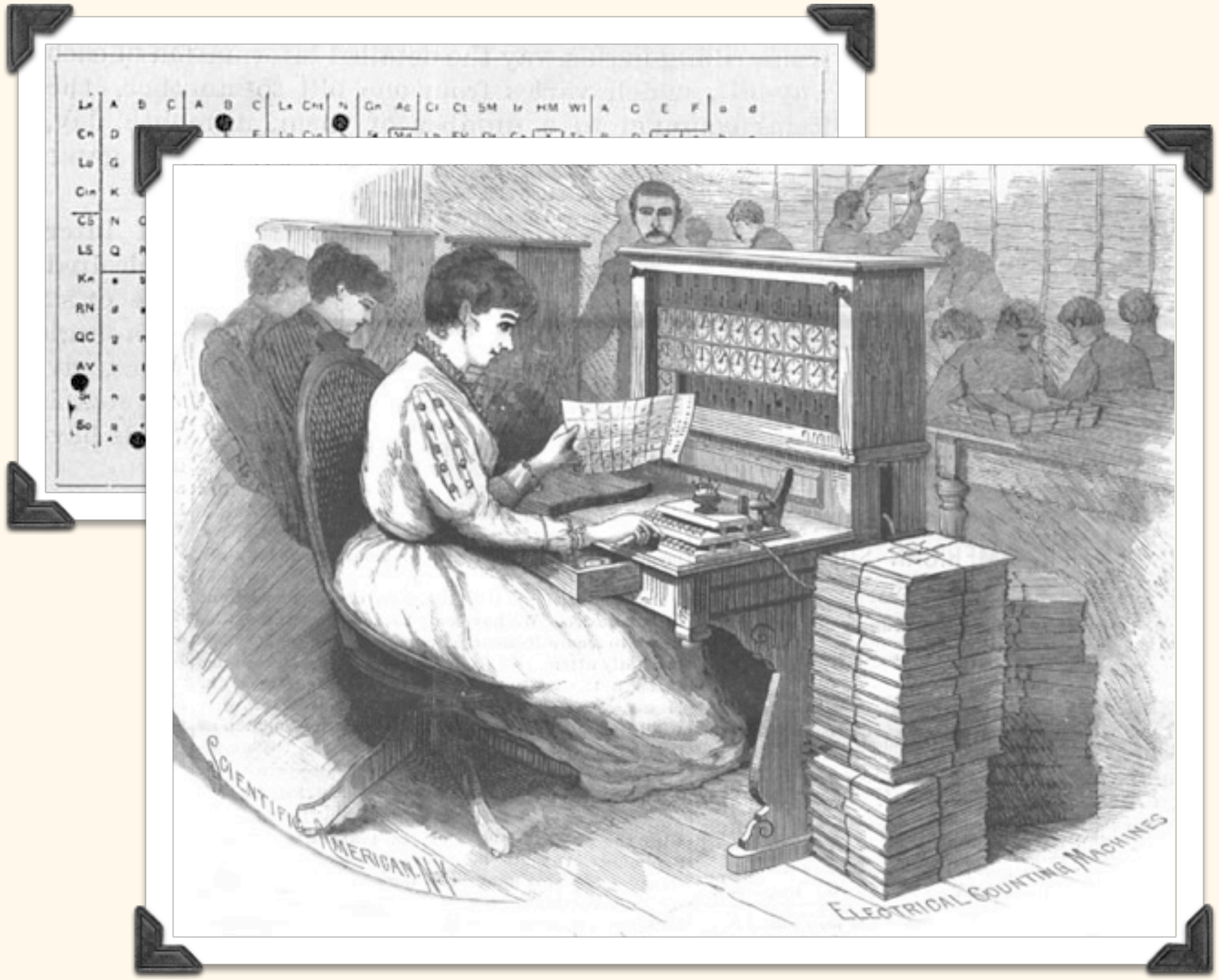
http://en.wikipedia.org/wiki/File:Clavier_Baudot.jpg

http://en.wikipedia.org/wiki/File:International_Telegraph_Alphabet_2.jpg

While the Baudot code is more immediately relevant today, we mustn’t forget Herman Hollerith.



Herman Hollerith
1860–1929



http://en.wikipedia.org/wiki/Herman_Hollerith

http://en.wikipedia.org/wiki/File:Hollerith_punched_card.jpg

http://en.wikipedia.org/wiki/File:1890_Census_Hollerith_Electrical_Counting_Machines_Sci_Amer.jpg

In 1963, the American Standards Association released the first version of ASCII.*

USASCII code chart

Bits b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4 b_3 b_2 b_1 ↓ Row ↓ Column →	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

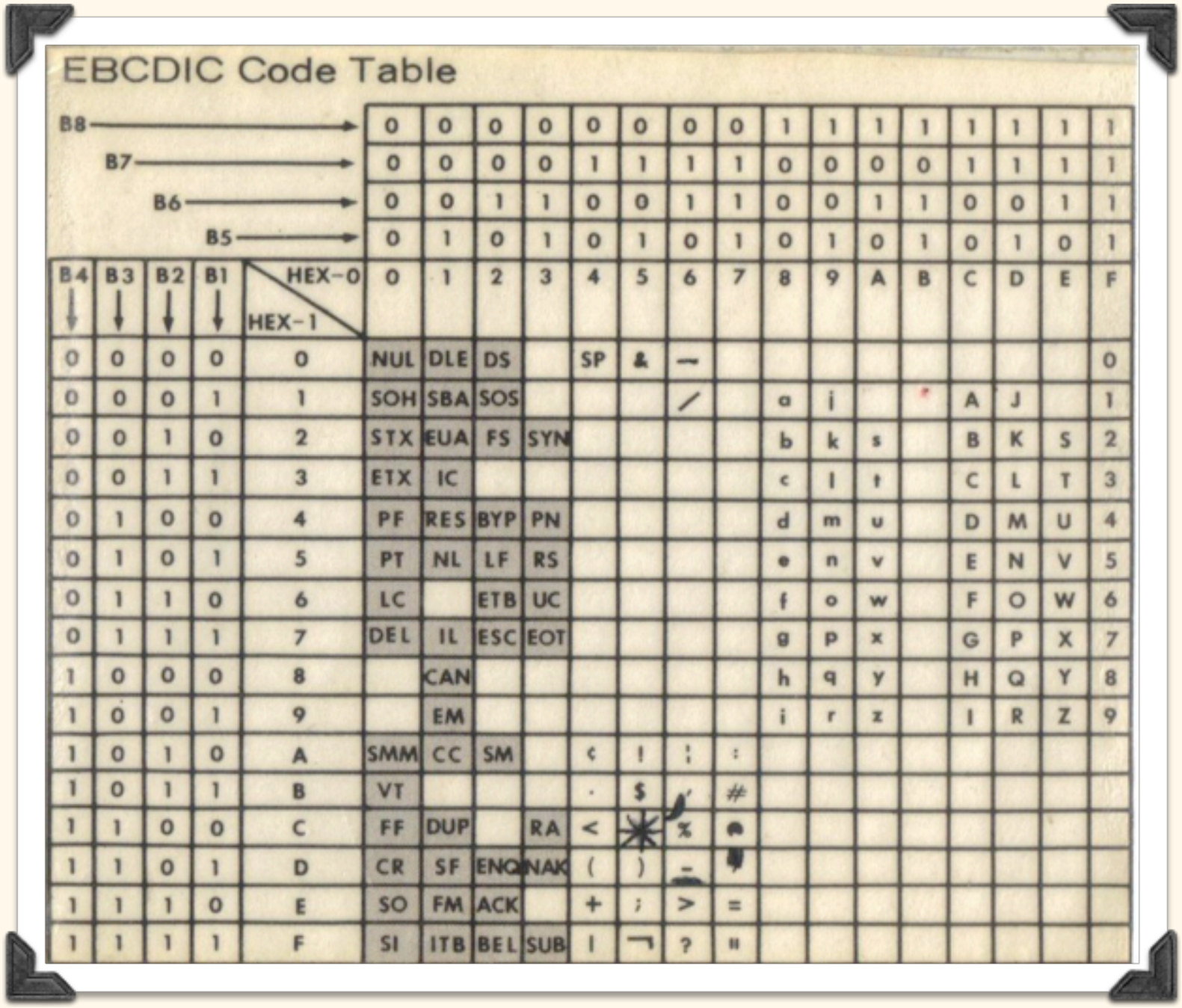
ASCII uses 7 bits for each character, and was strongly inspired by earlier teleprinter codes.

*American Standard Code for Information Interchange

http://en.wikipedia.org/wiki/File:ASCII_Code_Chart-Quick_ref_card.jpg

Not to be outdone, IBM introduced their own standard, EBCDIC* around the same time.

EBCDIC used 8 bits, and was directly descended from the Hollerith punch-card scheme.



EBCDIC Code Table

B8 →	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
B7 →	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
B6 →	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
B5 →	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
B4 ↓	B3 ↓	B2 ↓	B1 ↓	HEX-1 \ HEX-0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	NUL	DLE	DS		SP	&	—									0
0	0	0	1	1	SOH	SBA	SOS				/		a	j			A	J		1
0	0	1	0	2	STX	EUA	FS	SYN					b	k	s		B	K	S	2
0	0	1	1	3	ETX	IC							c	l	t		C	L	T	3
0	1	0	0	4	PF	RES	BYP	PN					d	m	u		D	M	U	4
0	1	0	1	5	PT	NL	LF	RS					e	n	v		E	N	V	5
0	1	1	0	6	LC		ETB	UC					f	o	w		F	O	W	6
0	1	1	1	7	DEL	IL	ESC	EOT					g	p	x		G	P	X	7
1	0	0	0	8		CAN							h	q	y		H	Q	Y	8
1	0	0	1	9		EM							i	r	z		I	R	Z	9
1	0	1	0	A	SMM	CC	SM		¢	!	¦	:								
1	0	1	1	B	VT				.	$	,	#								
1	1	0	0	C	FF	DUP		RA	<	*	%	@								
1	1	0	1	D	CR	SF	ENQ	NAK	(	)	_	'								
1	1	1	0	E	SO	FM	ACK		+	;	>	=								
1	1	1	1	F	SI	ITB	BEL	SUB	|	¬	?	"								

* Extended Binary Coded Decimal Interchange Code

Notice anything in particular about the character repertoire in both schemes?

3	4	5	6	7
0	@	P	`	p
1	A	Q	a	q
2	B	R	b	r
3	C	S	c	s
4	D	T	d	t
5	E	U	e	u
6	F	V	f	v
7	G	W	g	w
8	H	X	h	x
9	I	Y	i	y
:	J	Z	j	z
;	K	[	k	{
<	L	\	l	|
=	M	]	m	}
>	N	^	n	~
?	O	_	o	DEL

								0
	a	j			A	J		1
	b	k	s		B	K	S	2
	c	l	t		C	L	T	3
	d	m	u		D	M	U	4
	e	n	v		E	N	V	5
	f	o	w		F	O	W	6
	g	p	x		G	P	X	7
	h	q	y		H	Q	Y	8
	i	r	z		I	R	Z	9
:								

There were two main strategies for handling non-English characters:

1. Use the spare bit to free up more space; stash more characters in 0x80–0xFF.
2. Use more than one byte per character.

Købmagergade

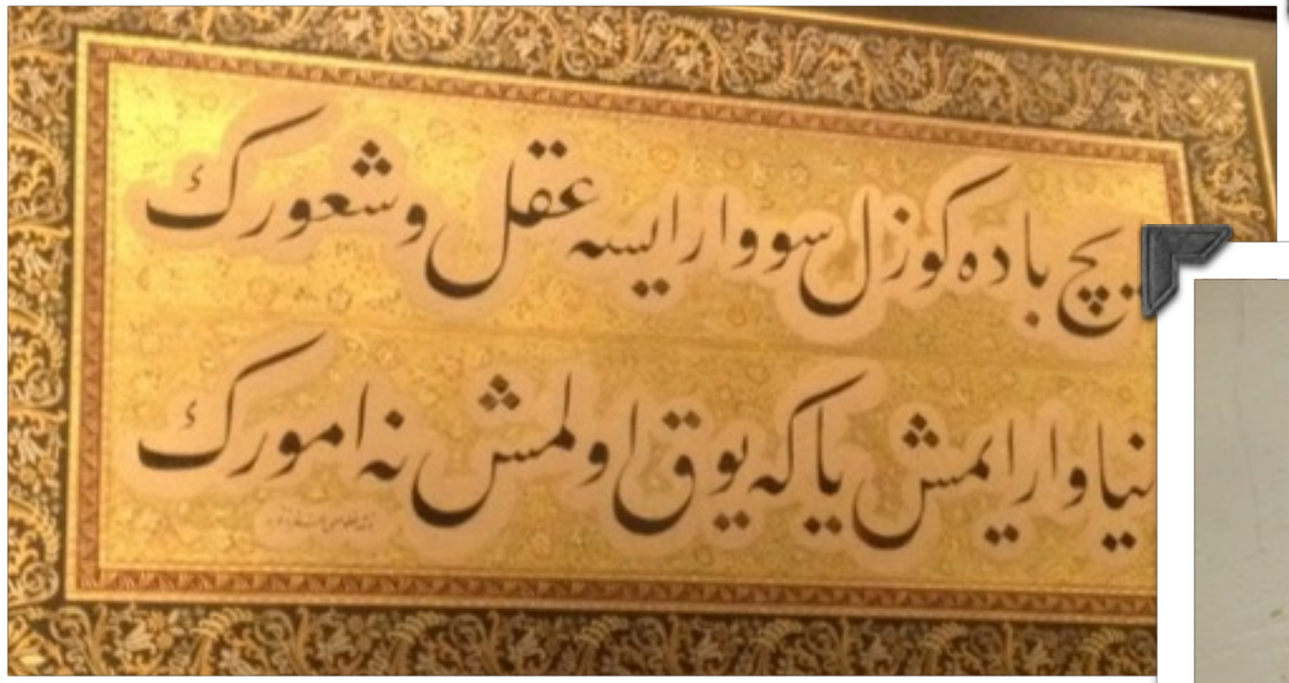
Østerbrogade

Table 2 – Code table of Latin alphabet No. 1

b_4	b_3	b_2	b_1		b_8 0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	
					b_7 0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	
					b_6 0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	
					b_5 0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
					00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	
0	0	0	0	00			SP	0	@	P	`	p			NBSP	°	À	Ð	à	ð	0
0	0	0	1	01			!	1	A	Q	a	q			¡	±	Á	Ñ	á	ñ	1
0	0	1	0	02			"	2	B	R	b	r			¢	²	Â	Ò	â	ò	2
0	0	1	1	03			#	3	C	S	c	s			£	³	Ã	Ó	ã	ó	3
0	1	0	0	04			$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô	4
0	1	0	1	05			%	5	E	U	e	u			¥	µ	Å	Õ	å	õ	5
0	1	1	0	06			&	6	F	V	f	v			¦	¶	Æ	Ö	æ	ö	6
0	1	1	1	07			'	7	G	W	g	w			§	·	Ç	×	ç	÷	7
1	0	0	0	08			(	8	H	X	h	x			¨	¸	È	Ø	è	ø	8
1	0	0	1	09			)	9	I	Y	i	y			©	¹	É	Ù	é	ù	9
1	0	1	0	10			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú	A
1	0	1	1	11			+	;	K	[	k	{			«	»	Ë	Û	ë	û	B
1	1	0	0	12			,	<	L	\	l	|			¬	¼	Ì	Ü	ì	ü	C
1	1	0	1	13			-	=	M	]	m	}			SHY	½	Í	Ý	í	ý	D
1	1	1	0	14			.	>	N	^	n	~			®	¾	Î	Þ	î	þ	E
1	1	1	1	15			/	?	O	_	o				¯	¿	Ï	ß	ï	ÿ	F
					0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	hex

1997-11-11

ISO 8859-1 (aka “ISO Latin-1”)


بچ باده کوزل سووارایسه عقل وشعورک
نیاوارایمش یاکه یوق اولمش نه امورک


ΟΔΟΣ
ΖΩΓΡΑΦΟΥ


A1900
INDIA IS GREAT
DANGER
ট্রাফিক আইন মেনে চলুন

Table 2 - Code table of Latin/Arabic alphabet

b_4	b_3	b_2	b_1		b_8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	
					b_7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	
					b_6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	
					b_5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
						00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	
0	0	0	0	00			SP	0	٠	@	P	`	p			NBSP			ذ	-	◌ِ	0
0	0	0	1	01			!	1	١	A	Q	a	q					ء	ر	ف	◌ّ	1
0	0	1	0	02			"	2	٢	B	R	b	r					آ	ز	ق	◌ْ	2
0	0	1	1	03			#	3	٣	C	S	c	s					أ	س	ئ		3
0	1	0	0	04			$	4	٤	D	T	d	t			¤		ؤ	ش	ل		4
0	1	0	1	05			%	5	٥	E	U	e	u					إ	ص	م		5
0	1	1	0	06			&	6	٦	F	V	f	v					ئ	ض	ن		6
0	1	1	1	07			'	7	٧	G	W	g	w					ا	ط	ه		7
1	0	0	0	08			(	8	٨	H	X	h	x					ب	ظ	و		8
1	0	0	1	09			)	9	٩	I	Y	i	y					ة	ع	ى		9
1	0	1	0	10			*	:		J	Z	j	z					ت	غ	ي		A
1	0	1	1	11			+	;		K	[	k	{				؛	ث		ً		B
1	1	0	0	12			,	<		L	\	l	|			،		ج		ٌ		C
1	1	0	1	13			-	=		M	]	m	}			SHY			ح	ٍ		D
1	1	1	0	14			.	>		N	^	n	~					خ		َ		E
1	1	1	1	15			/	?		O	_	o					؟	د		ُ		F
						0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	hex

ISO 8859-6, for simple Arabic (but not Persian, etc.)

Table 2 - Code table of Latin/Cyrillic alphabet

b_8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	
b_7	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	
b_6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	
b_5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	
b_4 b_3 b_2 b_1																	
0 0 0 0	00			SP	0	@	P	`	p			NBSP	А		а	р	№	0
0 0 0 1	01			!	1	A	Q	a	q			Ё	Б	С	б	с	ё	1
0 0 1 0	02			"	2	B	R	b	r			Ђ	В	Т	в	т	ђ	2
0 0 1 1	03			#	3	C	S	c	s			Ѓ	Г	У	г	у	ѓ	3
0 1 0 0	04			$	4	D	T	d	t			Є	Д	Ф	д	ф	є	4
0 1 0 1	05			%	5	E	U	e	u			S	Е	Х	е	х	s	5
0 1 1 0	06			&	6	F	V	f	v			І	Ж	Ц	ж	ц	і	6
0 1 1 1	07			'	7	G	W	g	w			Ї	З	Ч	з	ч	ї	7
1 0 0 0	08			(	8	H	X	h	x			J	И	Ш	и	ш	j	8
1 0 0 1	09			)	9	I	Y	i	y			Љ	Й	Щ	й	щ	љ	9
1 0 1 0	10			*	:	J	Z	j	z			Њ	К	Ъ	к	ъ	њ	A
1 0 1 1	11			+	;	K	[	k	{			Ћ	Л	Ы	л	ы	ћ	B
1 1 0 0	12			,	<	L	\	l	|			Ќ	М	Ь	м	ь	ќ	C
1 1 0 1	13			-	=	M	]	m	}			SHY	Н	Э	н	э	§	D
1 1 1 0	14			.	>	N	^	n	~			ў	О	Ю	о	ю	ў	E
1 1 1 1	15			/	?	O	_	o				Џ	П	Я	п	я	џ	F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	hex

ISO 8859-5, for languages using the Cyrillic alphabet

This approach had “pros” and “cons”:

Pro:

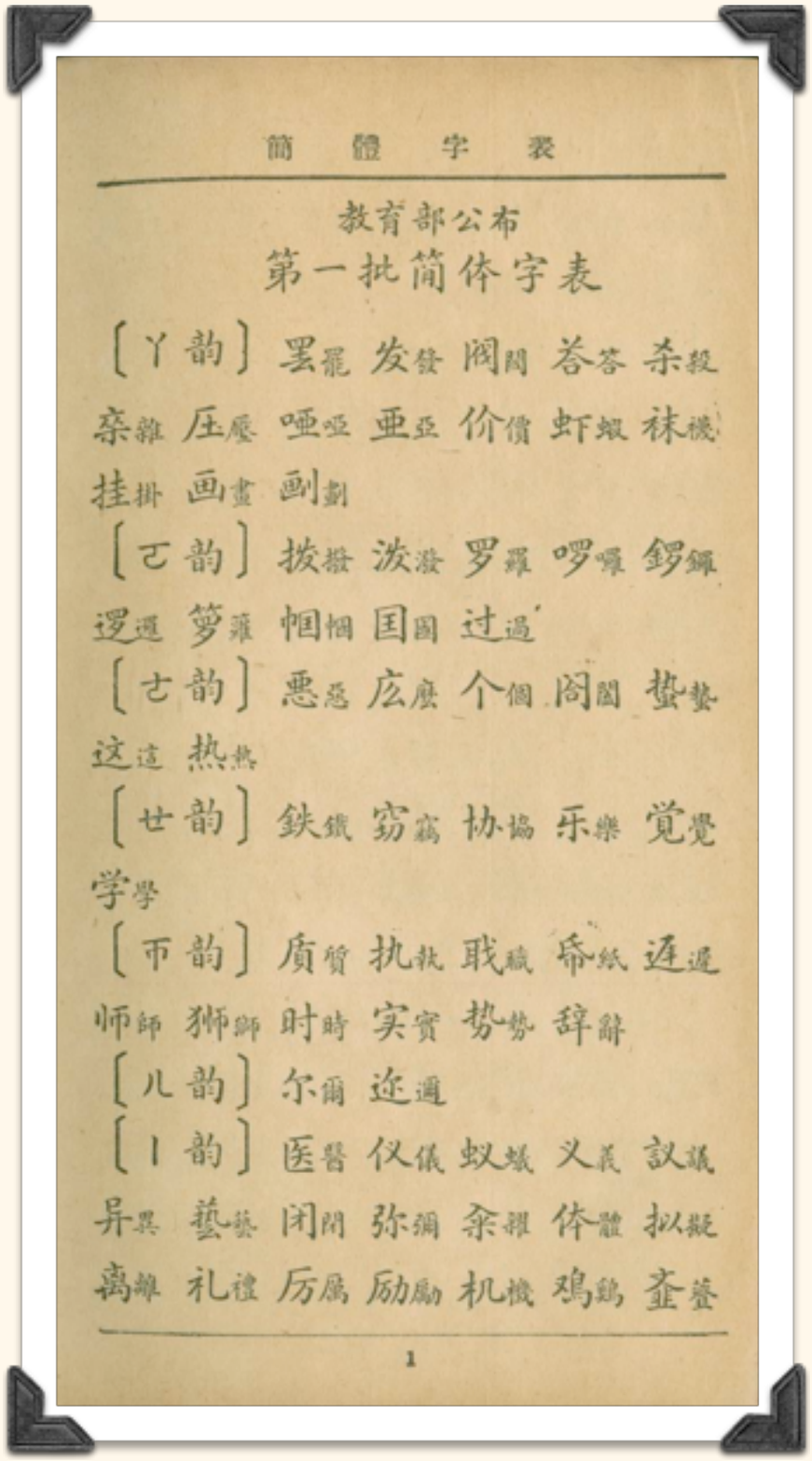
- Simple to implement
- Preserves compatibility with English-language computer systems
- Minimal changes needed to programming techniques

Con:

- No good way for a file to contain content in more than one language
- Difficult to transfer files between computers in different countries
- What if an alphabet needs more than 128 characters?

The most obvious example: East Asian languages such as Chinese which use many thousands of characters.

The first chunk of the “simplified” character set used in the PRC contained ≈300 characters.



简 體 字 表

教育部公布

第一批简体字表

〔丫韵〕罢罷 发發 阀閥 答答 杀殺
杂雜 压壓 哑啞 亚亞 价價 虾蝦 袜襪
挂掛 画畫 剧劇
〔已韵〕拔撥 泼潑 罗羅 啰囉 锣鑼
逻邏 箩籮 帼幗 国國 过過
〔亡韵〕恶惡 厉厲 个個 冈岡 垦墾
这這 热熱
〔廿韵〕铁鐵 窃竊 协協 乐樂 觉覺
学學
〔币韵〕质質 执執 戟戴 帋紙 迳逕
师師 狮獅 时時 实實 势勢 辞辭
〔儿韵〕尔爾 迩邇
〔丨韵〕医醫 仪儀 蚁蟻 义義 议議
异異 艺藝 闭閉 弥彌 杂雜 体體 拟擬
离離 礼禮 厉厲 励勵 机機 鸡鷄 奁簾

1

http://en.wikipedia.org/wiki/File:ROC24_SC1.jpg

The solution: use more than one byte per character.

Two bytes = 2^16 ≈ 65,000 possible characters.

0xB9 0x40　　辟

Of course, there are dozens of different such schemes, each with vendor-specific variations.

By the 1980s, the situation had gotten bad.

Primarily due to wider global adoption of computers, and the resulting increase in electronic information exchange.

Something had to be done...

In 1987, engineers from Xerox and Apple began planning a unified, multilingual, and extensible character set:

In 1991, the Unicode Consortium published Version 1.0 of the Unicode standard; the current version is 6.3.

Unicode defines an address space of more than 1,000,000 possible characters, and currently specifies ≈110,000.

Characters in Unicode are assigned a distinct numerical address, their “code point”.

Code points are grouped by writing system (“Latin”, “Han”, etc.).

Code points U+0000 through U+00FF are basically identical to ISO 8859-1.

Anatomy of a code point:

Letter	A
Code Point	U+0041
Name	Latin Capital Letter A
Script	Latin
Category	Uppercase Letter

Anatomy of a code point:

Letter	ج
Code Point	U+062C
Name	Arabic Letter Jeem
Script	Arabic
Category	“Other Letter”

Unicode specifies other metadata for each code point:

Case-folding equivalents (A → a)

“Compatibility” equivalents (² → 2)

Text direction (right-to-left vs. left-to-right)

Hyphenation and line-breaking rules

Ligature and joining rules (think ﬂ and ﷲ)

Numerical equivalency

Punctuation properties

Etc.

What about diacritic marks?

Unicode allows for “combining” code points:

Letter	´
Code Point	U+00B4
Name	Acute Accent
Script	Common
Category	Modifier Symbol

These allow “composition” of Unicode characters:

U+0065 U+00B4 é

But wait! What about:

Letter	é
Code Point	U+00E9
Name	Latin Small Letter e with Acute
Script	Latin
Category	Lowercase Letter

Unicode allows *both* pre- and post-composed accented characters...

So “café” might not equal “café”!

Luckily, Unicode also specifies rules for normalizing strings for sorting, searching, comparing, etc.

There are four “normalization forms:”

café²

NFD	cafe´²	U+0063	U+0061	U+0066	U+0065	U+0301	U+00B2
NFC	café²	U+0063	U+0061	U+0066	U+00E9	U+00B2	
NFKD	cafe´2	U+0063	U+0061	U+0066	U+0065	U+0301	U+0032
NFKC	café2	U+0063	U+0061	U+0066	U+00E9	U+0032	

Luckily, this doesn’t come up *too* often... and many software systems deal with it transparently.

Thus far, we've talked about Unicode as an abstract *character set*.

How do we *encode* Unicode strings?

There are, as always, many different options.

The first popular Unicode encoding was UCS-2, which simply expresses the code point as a pair of bytes:

A	U+0041	`0x00 0x41`
é	U+00E9	`0x00 0xE9`

This works, but eventually, 16 bits weren't enough...

UTF-16 introduced *surrogate pairs*, which allow non-BMP code points to be encoded. Many popular programs utterly fail to handle this properly.

The first popular Unicode encoding was UCS-2, which simply expresses the code point as a pair of bytes:

Or should that be `0x41 0x00`?

A	U+0041	`0x00 0x41`
é	U+00E9	`0x00 0xE9`

UCS-2/UTF-16 have the problem of *endianness*- given two bytes, which is the “most significant?”

UTF-16 allows for a “byte order marker” at the beginning of a file, which allows decoders to infer the encoder’s endianness.

The most commonly-seen Unicode encoding is UTF-8, which uses a variable number of bytes to encode each code point.

UTF-8 has the lovely property that code points 0x00 through 0x7F encode identically to their ASCII counterparts!

A	U+0041	`0x41`
é	U+00E9	`0xC3 0xA9`

Most of the time, for most people, UTF-8 is the sensible choice...

... however, if your text has many characters from higher-up in the Unicode code space (≥U+0800), UTF-16 can be more efficient.

However! Beware the surrogate pair and byte-order-marker, for therein lie troubles.

Problems arise when there is a mismatch between the scheme used to encode a file and that used to decode it.

Word: “café”

UTF-8 bytes: `0x63 0x61 0x66 0xC3 0xA9`

Latin-1 interpretation: “cafÂ©”

Common characters that cause trouble include:

Diacritics (é, ü)
Typographical (“smart”) quotes (“”)
Trademark & Copyright symbols (™, ®, ©)

Common trouble scenarios:

File written in one program, read in another...

Data sent to database, stored, retrieved, and displayed (each a possible point of encoding-related trouble)

User enters data into a web form, hits “submit”, hilarity ensues...

The solution in all cases is the same:

Be clear about what encoding you're producing and what you're expecting!

This is one nice thing about XML: valid XML documents all specify their encoding...

... but note that some documents *lie*.

That’s it for today...

For Wednesday: Read chapter 2 of J&M

Questions?