TEXT ENCODING

CS 562/662: Natural Language Processing

2015-01-08

SEND ME YOUR RESUMÃ© ??

ODE TO A SHIPPING LABEL

Once there was a little **o**
with an accent on top like só.

It started out as UTF-8
(Universal since ’98),
but the program only knew Latin-1,
and changed little **ó** to **Ã³** for fun.

A second program saw the **Ã³**
and said “I know HTML entity!”
So **Ã³** was smartened up to **&ATILDE;&SUP3;**
and passed on through happily.

Another program saw the tangle
(more precisely, ampersands to mangle)
and thus the humble **&ATILDE;&SUP3;** became
&&ATILDE;&&SUP3;.

[Source: Traditional]

Text was something of an afterthought for the creators of digital computing. They were interested in one thing—numbers—and that's fortunate: *computers only know about numbers*.

GLOSSARY

Character: whatever humans think of as the smallest atomic unit of written text

Glyph: the shape of a character

Character set: a finite, organized (i.e., numbered) catalog of characters

Character encoding: an algorithm for mapping elements of a character set to binary digits

Character decoding: the inverse algorithm thereof

ASCII (1963)

Hex	Dec	Char		Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
0x00	0	NULL	null	0x20	32	Space	0x40	64	@	0x60	96	`
0x01	1	SOH	Start of heading	0x21	33	!	0x41	65	A	0x61	97	a
0x02	2	STX	Start of text	0x22	34	"	0x42	66	B	0x62	98	b
0x03	3	ETX	End of text	0x23	35	#	0x43	67	C	0x63	99	c
0x04	4	EOT	End of transmission	0x24	36	$	0x44	68	D	0x64	100	d
0x05	5	ENQ	Enquiry	0x25	37	%	0x45	69	E	0x65	101	e
0x06	6	ACK	Acknowledge	0x26	38	&	0x46	70	F	0x66	102	f
0x07	7	BELL	Bell	0x27	39	'	0x47	71	G	0x67	103	g
0x08	8	BS	Backspace	0x28	40	(	0x48	72	H	0x68	104	h
0x09	9	TAB	Horizontal tab	0x29	41	)	0x49	73	I	0x69	105	i
0x0A	10	LF	New line	0x2A	42	*	0x4A	74	J	0x6A	106	j
0x0B	11	VT	Vertical tab	0x2B	43	+	0x4B	75	K	0x6B	107	k
0x0C	12	FF	Form Feed	0x2C	44	,	0x4C	76	L	0x6C	108	l
0x0D	13	CR	Carriage return	0x2D	45	-	0x4D	77	M	0x6D	109	m
0x0E	14	SO	Shift out	0x2E	46	.	0x4E	78	N	0x6E	110	n
0x0F	15	SI	Shift in	0x2F	47	/	0x4F	79	O	0x6F	111	o
0x10	16	DLE	Data link escape	0x30	48	0	0x50	80	P	0x70	112	p
0x11	17	DC1	Device control 1	0x31	49	1	0x51	81	Q	0x71	113	q
0x12	18	DC2	Device control 2	0x32	50	2	0x52	82	R	0x72	114	r
0x13	19	DC3	Device control 3	0x33	51	3	0x53	83	S	0x73	115	s
0x14	20	DC4	Device control 4	0x34	52	4	0x54	84	T	0x74	116	t
0x15	21	NAK	Negative ack	0x35	53	5	0x55	85	U	0x75	117	u
0x16	22	SYN	Synchronous idle	0x36	54	6	0x56	86	V	0x76	118	v
0x17	23	ETB	End transmission block	0x37	55	7	0x57	87	W	0x77	119	w
0x18	24	CAN	Cancel	0x38	56	8	0x58	88	X	0x78	120	x
0x19	25	EM	End of medium	0x39	57	9	0x59	89	Y	0x79	121	y
0x1A	26	SUB	Substitute	0x3A	58	:	0x5A	90	Z	0x7A	122	z
0x1B	27	FSC	Escape	0x3B	59	;	0x5B	91	[	0x7B	123	{
0x1C	28	FS	File separator	0x3C	60	<	0x5C	92	\	0x7C	124	|
0x1D	29	GS	Group separator	0x3D	61	=	0x5D	93	]	0x7D	125	}
0x1E	30	RS	Record separator	0x3E	62	>	0x5E	94	^	0x7E	126	~
0x1F	31	US	Unit separator	0x3F	63	?	0x5F	95	_	0x7F	127	DEL

ARCHAIC CONTROL CHARACTERS

- `'\a'` (“alert”): ring a bell on the teletype machine

- `'\n'` (“line feed”): advance the paper ream one line

- `'\r'` (“carriage return”): return the printer head carriage to the leftmost edge of the page

COOPERATION?

TWO STRATEGIES

- Exploit the extra bit for another 128 characters

- Use more than one byte per character

ISO/IEC 8859

- Part 1: “Latin-1” (Western European languages): Danish*, Faroese, Finnish*, French*, German, Icelandic, Irish, Italian, Norwegian, Portuguese, Rhaeto-Romance, Scottish Gaelic, Spanish, Catalan, Swedish

- Part 2: “Latin-2” (Central European languages with Roman alphabets): Bosnian, Polish Croatian, Czech, Slovak, Slovene, Serbian, Hungarian

- …

- Part 5: Cyrillic: …

- ...

- Part 8: Hebrew

[*: incomplete support]

UNICODE (1991)

Letter: e
Code point: U+0065
Name: Latin Small Letter e
Script: Latin
Category: Lowercase Letter

Letter: ج
Code point: U+062C
Name: Arabic Letter Jeem
Script: Arabic
Category: Other Letter

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

[U+0065 U+00B4: é]

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

UNICODE METADATA

- Case-folding equivalences (*A* ~ *a*)
- Text direction (left-to-right vs. right-to-left)
- Line-breaking and hyphenation rules
- Ligaturing rules
- etc.

WHEN IS A CAFÉ NOT A CAFÉ?

café²

NFD	U+0063	U+0061	U+0066	U+0065	U+0301	U+00B2
NFC	U+0063	U+0061	U+0066	U+00E9	U+00B2	
NFKD	U+0063	U+0061	U+0066	U+0065	U+0301	U+0032
NFKC	U+0063	U+0061	U+0066	U+00E9	U+0032	

[h/t: Steven Bedrick]

Fast NFKD normalization Perl script:

```
#!/usr/bin/perl
use strict;
use warnings;
use Unicode::Normalize;
use open ":encoding(utf8)";
binmode STDIN, ":encoding(utf8)";
binmode STDOUT, ":encoding(utf8)";
binmode STDERR, ":encoding(utf8)";
while (<>) {
     $_ = NFKD($_);
     print;
}
```

BUT HOW DO WE ENCODE UNICODE?

PROBLEMS WITH UTF-16

- Each code point is expressed as a pair of bytes: very inefficient on average (more on that shortly)

- Problems with the *byte order mark* (BOM) mean that there are really three UTF-16s:

- Proper UTF-16, with a BOM

- big-endian UTF-16 with a missing BOM

- little-endian UTF-16 with a missing BOM

WHY UTF-8 IS BETTER

- Variable byte size is maximally efficient
- And, the first (and shortest) 128 code points are reserved for ASCII
- So, ASCII is valid UTF-8!

ICONV DEMO

PYTHON (2/3)

ENCODING DEMO

REMAINING PROBLEMS

- Most text files do not include a specification of their encoding, and there is **no** infallible way to determine the intended encoding
- And those that do specify their encoding may lie!
- And then there's line endings…

LINE ENDINGS

THE 3 MAJOR STYLES

- DOS and Windows: carriage return and line feed (CR+LF): `\r\n`
- UNIX and spiritual descendants (including Mac OS X): line feed (LF): `\n`
- pre-OS X Mac OS: carriage return (CR): `\r`

Fast, *in-place* newline converter Perl scripts:

```
alias mac2unix="perl -p -i -e 'tr/\r/
\n/'"
alias dos2unix="perl -p -i -e 'tr/
\r//d'"
```

TOKENIZATION

CS 562/662: Natural Language Processing

2015-01-08

Tokenization refers to methods applied to streams of (properly encoded) text to separate it into meaningful chunks.

Pierre Vinken, 61 years old, will join the board as a non executive director Nov. 29. Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group. (WSJ)

Two important units: *token* and *sentence*.

Pierre Vinken , 61 years old , will join the board as a nonexecutive director Nov. 29 .

Mr. Vinken is chairman of Elsevier N.V. , the Dutch publishing group .

Rolls-Royce Motor Cars Inc. said it expects its U.S. sales to remain steady at about 1,200 cars in 1990.

WORD TOKENIZATION

For English: separate words and punctuation characters that glom onto their left (but not, e.g., periods marking abbreviations) and call the resulting units *tokens*

PENN TOKENIZER

The segmentation of known words can also be ambiguous. For example, 这里面 should be 这里 (here) 面 (flour) in the sentence 这里面和米很贵 (flour and rice are expensive here) or 这 (here) 里面 (inside) in the sentence 这里面很冷 (it's cold inside here). The ambiguity can be resolved with information about the neighboring words. In comparison, for the sentence 洽谈会很成功, possible segmentations include 洽谈 (the discussion) 会 (will) 很 (very) 成功 (be successful) and 洽谈会 (the discussion meeting) 很 (very) 成功 (be successful). The ambiguity can only be resolved with contextual information outside the sentence. Human readers often use semantics, contextual information about the document, and world knowledge to resolve segmentation ambiguities.

[Source: Zhang & Clark 2011]

SENTENCE TOKENIZATION

For English: identify which sentential punctuation characters *really* introduce sentence boundaries

FEATURES FOR SENTENCE BOUNDARY DETECTION

```
/(\S+)\s*((\.+)|([!?]))(['`"]}\]]*)(\s+)\s*(\S+)/
```

1. Identity of the candidate punctuation mark
2. Identity of the two (left and right) adjacent tokens
3. The *joint* identity of the adjacent tokens
4. The casing of the two adjacent tokens
5. Does the left token contain a vowel?
6. Does the left token contain an internal period?
7. How many characters long is the left token?

[Source: http://sonny.cslu.ohsu.edu/~gormanky/blog/simpler-sentence-boundary-detection/]