

Unicode for Linguists

Kevin Scannell
INNET Summer School
Gniezno, Poland
4 September 2013

Encoding basics

- Encoding: a mapping characters \leftrightarrow numbers
- Computers are good at storing numbers!
- Encodings don't care how chars are input
- Also not concerned with how they're displayed
- No fonts, formatting, boldface, etc. – just text!
- “Modularity”; three independent layers
- Confusion in part b/c it was not always so
- (keyboards and fonts bundled together)

Fonts

- Again, correctly-encoded text stands on its own
- Sure, fonts let you to render text on the screen
- But you have good wide-coverage fonts already
- There are some exceptions of course:
- | | | | | | | | | | | | | | | |
- And some fonts are buggy...
- Especially stacked diacritics, ligatures

The Joy of Text

- Unicode encodes “plain text” (and no more)
- Plain text is portable across platforms
- Plain text is long-lived
- Plain text is the ultimate “open” format
- Plain text is indexed by Google, “discoverable”
- No proprietary tools needed to manipulate text
- Plain text is easily viewable, editable
- Allows you to bring powerful Unix tools to bear
- Even marked-up text like XML counts!

Encodings: Some History

- ASCII (1963); A-Za-z, 0-9, etc., map to 0-127
- EBCDIC (1964); mapping to 0-255 (w/ gaps)
- ISO 8859 series (1987-); 0-255, extend ASCII
- Big5 (1984); traditional Chinese
- Shift JIS; Japanese
- GB2312/18030; simplified Chinese
- Literally hundreds of others
- Why was this so terrible?

Alphabet soup

IBM1124, CP424, ISO_10367-BOX, ECMA-128, ISO8859-1, KOI-7, CSIBM1137, EBCDIC-GREEK, ISO-IR-99, GOST_1976874, CSISOLATINCYRILLIC, CP901, IBM933, IBM-1142, CP1026, IBM4909, UTF32BE, CSISO19LATINGREEK, CSIBM4971, MS-GREEK, MAC, ISO_8859-10:1992, TS-5881, WINBALTRIM, IBM-1161, NF_Z_62-010, CSIBM420, CSISO89ASMO449, CP1144, ISO_6937-2, CP-AR, CP281, T.61-8BIT, EUCTW, CSIBM500, ISO-IR-141, INIS, CP10007, IBM16804, PT2, ISO-8859-13, IBM12712, IBM-1162, CSISO86HUNGARIAN, SHIFT_JIS, CUBA, ISO-IR-138, EBCDIC-ES-S, ISO-IR-4, ISO_8859-2:1987, OSF10020359, EBCDIC-CP-FI, CP1251, ISO_8859-4:1988, ISO-IR-103, GB18030, EBCDICATDE, OSF00010020, CSPC862LATINHEBREW, ISO_8859-14, 8859_5, CSISO5427CYRILLIC, CSA_T500-1983, EBCDIC-CP-WT, EBCDIC-IS-FRISS, CSIBM1364, CP274, ISO885914, EBCDIC-CP-AR1, IBM1132, CSIBM863, CSIBM1123, BIG-FIVE, ISO8859-2, 904, IBM1129, ISO-8859-11, SE, CSIBM1026, CP285, UTF16LE, IBM1141, CSIBM297, CP1097, IBM856, IBM277, 1046, NF_Z_62-010_1973, HPTHAI8, CSISO15ITALIAN, CP1141, IBM-1147, CSIBM902, IBM1026, CP819, ISO-8859-9E, CSIBM1129, EBCDICDKNO, CP937, WINDOWS-31J, ARMSCII-8, EBCDIC-AT-DE, ISO_8859-7:1987, IBM9448, CP4909, GB_1988-80, CP278, DECMCS, IBM273, 8859_7, NC_NC00-10:81, EBCDIC-CP-SE, IBM1160, CSISOLATIN2, IBM-1166, OSF10020365, IBM1388, OSF10020111, RK1048, ISO88598, CSISO150GREEKCCITT, CP9448, ISO-IR-69, BIG5, IBM904, ASMO_449, CSISOLATIN4, MACUK, CSISO122CANADIAN2, CSIBM1148, ISO2022CN, IBM866NAV, CSIBM903, ASMO-708, CP1004, IBM1158, CP297, CSISO141JUSIB1002, CP437, MS-EE, CP771, CP1255, IBM1143, CP772, DEC, OSF1002035D, DS2089, MSCP949, ISO-2022-JP-2, TIS-620, ISO88592, CSISO27LATINGREEK1, CP1161, DE, 855, ISO-2022-JP-3, CP1256, CSIBM11621162, CP1390, MS-TURK, NATSDANO, CP500, 1026, IBM1137, IBM284, ISO885916, OSF10020352, CSIBM1390, IBM1163, ES2, ISO_8859-5, CP1160, HPGREEK8, IBM-933, MACINTOSH, UCS4, CP891, LATIN3, CSISO69FRENCH, CP949, IBM-1124, EBCDIC-AT-DE-A, CSISOLATIN5, NAPLPS, IBM868, EUCCN, INIS-8, CSIBM939, ISO885913, CSIBM860, ISO-IR-86, NF_Z_62-010_(1973), ISO8859-7, ANSI_X3.110, ISO-IR-89, CP1364, ISO_8859-9, JIS_C6229-1984-B, ISO_8859-3:1988, CP903, MAC-UK, 437, CSIBM866, WINDOWS-1258, ISO646-CA2, CP939, OSF10020354, KOI8R, CSA7-1, IBM-1122, CP4517, IBM855, ISO_6937-2:1983, GEORGIAN-PS, CSIBM935, UCS-2LE, IBM-16804, CP1081, IBM-1163, CP1124, LATIN10, WINDOWS-1257, CP874, IBM916, ISO_9036, CSIBM803, ISO8859-5, IBM-1046, CSIBM1097, EBCDIC-CP-CA, ISO_69372, CSISO60DANISHNORWEGIAN, OSF10020115, CSEBCDICFISE, EBCDIC-JP-KANA, CP1282, CSIBM1112, IBM874, IBM-1143, BALTIC, CSIBM1025, INIS8, EBCDIC-CP-NO, CP902, BIG-5, CP1254, CSIBM855, EBCDICESA, JP-OCR-B, TCVN5712-1:1993, ISO646-FR1, TSCII, IBM-9066, CSIBM1132, ISO2022JP2, EBCDICDKNOA, IBM1164, IBM4517, UTF-16BE, CP1025, ISO-IR-111, IBM-4971, SJIS-WIN, MAC-CENTRALEUROPE, CP1137, IBM-1008, CSMACINTOSH, EBCDIC-CP-CH, CP905, CP273, VISCII, OSF00010008, ROMAN9, IBM-9448, OSF0001000A, 864, ISO_8859-10, IBM857, LATIN9, OSF10020357, ISO-IR-37, CSIBM273, EBCDIC-CP-HE, LATIN6, ISO88591, CSIBM1164, TIS620.2529-1, CSISO143IECP271, ISO-8859-9, CSIBM1399,

Massive ambiguity

- Character á maps to 225 under ISO-8859-1
- But 225 is ß in CP770 encoding
- And it's Cyrillic es (c) in CP771
- And Arabic Lam (J) in CSIBM9448
- And Thai Sara Ae (๒) in IBM-1161
- And j (lowercase i with ogonek) in ISO-8859-13
- And...

Multilingualism

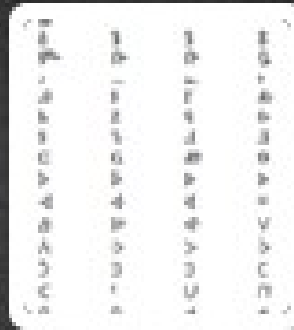
- No single encoding covered many scripts
- Multilingual documents were impossible
- Thumb drive contains `randomsample.txt`
- One random sentence in each of 1500+ langs
- Completely impossible 20 years ago!

Gaps

- Many scripts had no encoding at all
- Individuals invented “ad hoc” encodings
- Occasionally, just a few characters missing
- e.g. Welsh uses ISO 8859-1 + \hat{w} , \hat{y}
- Solution: create a new encoding! (ISO 8859-14)
- Mongolian uses standard Cyrillic + θ , γ
- Solution: “font tricks”; font redraws ϵ , \ddot{i} as θ , γ
- Or fonts that redraw *everything* (e.g. Cherokee)

Unicode!

- First efforts at a universal encoding, late 80's
- Unicode Consortium, ISO, 1991-1992
- First version 1.0.0 in Oct. 1991, 7161 chars
- Current version 6.2 (Sept. 2012), 110182 chars!
- Again, each char is assigned a number
- These numbers are called “code points”



everyunicode

@everyunicode

Twittering every graphical character in the Unicode 6.2 Standard. Task will complete in 2076.

nas.sr/everyunicode

3,527

TWEETS

0

FOLLOWING

506

FOLLOWERS



Ambiguity solved

- Character á still maps to code point 225
- ß maps to code point 223
- Cyrillic es (c) maps to 1089
- Arabic Lam (J) maps to 1604
- Thai Sara Ae (๒) maps to 3649
- j (lowercase i with ogonek) maps to 303

Digression on binary/hex

- So á is 225, but you usually see U+00E1
- ũ is 3649, but usually written U+0E41
- Computers store numbers in binary, “base 2”
- $225 = 128 + 64 + 32 + 1 = 11100001$
- $3649 = 2048 + 1024 + 512 + 64 + 1 = 111001000001$

Hexadecimal vs. binary

- 0 = 0000
- 1 = 0001
- 2 = 0010
- 3 = 0011
- 4 = 0100
- 5 = 0101
- 6 = 0110
- 7 = 0111
- 8 = 1000
- 9 = 1001
- 10 = 1010 = "A"
- 11 = 1011 = "B"
- 12 = 1100 = "C"
- 13 = 1101 = "D"
- 14 = 1110 = "E"
- 15 = 1111 = "F"

Serialization

- “Bits” are binary digits, 0 or 1
- Computer memory divided up into 8-bit chunks
- “Bytes” or “octets”
- 0...255, or 00000000...11111111, or 00...FF
- How do we represent Unicode as bytes?
- Q: Can we store áßcJµj using each code point?
- → 00 E1 00 DF 04 41 06 44 0E 41 01 2F

UTF-16

- Answer: Almost!
- This was essentially the original design
- But, 2 bytes per char allows only 65536 chars
- Code points up to $10FFFF = 1,114,111$
- There's a trick for stuff > 65535 ; more later...
- Also, UTF-16 is inefficient for mostly-ASCII text
- “Kevin” in ASCII: 4B 65 76 69 6E
- In UTF-16: 004B 0065 0076 0069 006E

UTF-8

- Another way to represent Unicode as bytes
- THE way; *de facto* standard on most computers
- ASCII (0...7F) encoded as themselves, 1 byte
- 0080...07FF encoded as 2 bytes (non-trivially!)
- 0800...FFFF encoded as 3 bytes each
- Others as 4 bytes each

A Mystery Explained

- *Co słychać? Co sÅychaÄ? Co sŁychaÄ?*
- ł is U+0142
- Serialized as UTF-8, it's two bytes C5 82
- Interpreting these as bytes in ISO 8859...
- C5 is Å in ISO 8859-1, Ł in ISO 8859-2
- 82 is unassigned in both cases
- Either “doubly-encoded”, or ISO 8859 viewer

Digression: how does UTF-8 work?

- How do I know the bytes C5 82 mean U+0142?
- C = 1100, 5 = 0101, 8 = 1000, 2 = 0010
- So bit stream is 11000101 10000010
- “110” at beginning says character is two bytes
- Throw this out, and the “10” at start of 2nd byte
- Leaves 00101000010.... that's 0142 in hex!!
- Similarly for three, four octet sequences

Ambiguity of another kind

- Sometimes multiple ways to represent one char
- Å = U+212B=ANGSTROM SIGN
- Å = U+00C5=LATIN CAPITAL LETTER A WITH RING ABOVE
- Å can also be encoded as U+0041 followed by U+030A
- LATIN CAPITAL LETTER A, COMBINING RING ABOVE
- U+00C5: “precomposed form”, U+030A: “combining diacritic”
- Can cause problems with search, frequency lists, etc.
- We'll talk about “Normalization Forms” later, given time

Demo

- <http://borel.slu.edu/pub/innet.html>
- That's U+212B, U+00C5, U+0041, U+030A.
- Copy the first char, then search (Ctrl+F) for it
- Repeat, copying 2nd, then 3rd chars.
- Change browsers and try again (!)
- Search merges (often desirable), inconsistently
- More dangerous: sometimes silently *changed*
- e.g. copy and paste, depending on platform/app

What's missing?

- See <http://scriptsource.org/>
- 241 scripts listed in total
- 163 listed in ISO 15924
- 102 scripts in Unicode 6.2
- Info on pending proposals from scriptsource, or:
<http://www.linguistics.berkeley.edu/sei/alpha-script-list.html>
- Bassa Vah, Duployan, Elbasan, Khojki, Khudawadi, ...

What about my language?

- Typical use cases:
- Convert legacy texts to UTF-8
- Fix miscoded files
- Create UTF-8 text from offline notes, images

Inspecting code points

- If you have existing Unicode text in your lang...
- Best approach is to use Unix tools (workshop)
- Can inspect and also modify/correct texts
- Quick online solution for inspecting code points
- <http://rishida.net/tools/conversion/>
- But beware of cutting and pasting!

Unicode font tricks

- Unicode isn't immune from font tricks
- http://www.babel.gwi.uni-muenchen.de/media/downloads/SzOCh_FUT_20110721.pdf
- Khanty language texts, Finno-Ugric Transcript.
- p.43, 2nd line: *atiλnam atə̃m wăłtayə*
- Cut and paste: *ati>nam atõm wâ>taqõ*

Your language in Unicode

- How should I represent my texts in Unicode?
- Is there a writing system? If not, create one!
- Decide on correct Unicode chars
- Unicode charts, or <http://shapecatcher.com/>
- May require use of combining characters
- If not all chars are available, propose new ones
- Sound easy?


Confusables!

- <http://www.unicode.org/Public/security/revision-02/confusables.txt>
- A = U+0041 = LATIN CAPITAL LETTER A
- Α = U+0391 = GREEK CAPITAL LETTER ALPHA
- А = U+0410 = CYRILLIC CAPITAL LETTER A
- ᐃ = U+15C5 = CANADIAN SYLLABICS CARRIER GHO
- Ꭰ = U+13AA = CHEROKEE LETTER GO
- ...
- How to find and fix incorrect characters in existing texts?
- How to make the right choice? ï → ĩ (U+0268 vs. U+0069, U+0335)

Normalization Forms

- NFC vs. NFD
- Indexing for search
- Twitter 140 count uses NFC

Special Characters

- U+FEFF (ZWNBSP a.k.a. BOM)
- U+FFFD  (replacement character)
- U+200C,U+200D (ZWNJ, ZWJ)
- U+200E,U+200F (LTR, RTL)

Keyboard resources

- <http://www.tavultesoft.com/keyman/>
- <http://keymanweb.com/>
- <http://msdn.microsoft.com/en-us/goglobal/bb964665.aspx>
- http://scripts.sil.org/cms/scripts/page.php?item_id=ukelele

Know your bytes!

- “Unix for Poets” – great tutorial by Ken Church
- <http://www.stanford.edu/class/cs124/kwc-unix-for-poets.pdf>
- “Better to do something simple than nothing at all”
- “DIY is more satisfying than begging for help”
- Here's my own tutorial, for linguists:
- <http://borel.slu.edu/pub/innet.html>
- Programming via pipelines
- Filter (grep), map (sed), reduce (wc, etc.)