COMP10001 Foundations of Computing
Digital Representation; Ethics and Bias

Semester 1, 2019
Tim Baldwin, Nic Geard, Farah Khan, and Marion Zalk

THE UNIVERSITY OF MELBOURNE

— VERSION: 1579, DATE: MAY 23, 2019 —

Reminders/Announcements

- Project 3 due Thursday 30/5
  no extensions will be possible, but if there are documented medical/personal reasons that have impacted on your ability to complete the project, contact us ASAP to we can come up with a stripped-down version of the project
- Academic honesty meetings to take place shortly for Projects 1 and 2 ... reminder that your submitted code should be your own, but also that you shouldn't be sourcing the code of others during the course of your code development ...
- SES open for subject feedback

Lecture Agenda

- Last lecture:
  - Properties and families of algorithms (cont.)
  - Computational counting
- This lecture:
  - Text representation
  - Ethics and fairness in computing

Lecture Outline

Take 1: Set the "Top Bit"

- **ISO-8859**
  - single-byte encoding built on top of ASCII (7-bit), to include an extra 128 characters, which is enough to represent many orthographies (Latin ± diacritics, Cyrillic, Thai, Hebrew, ...)
  - as 128 characters isn't enough to cover all these orthographies at once, the standard occurs in 16 variants (ISO-8859-1 to ISO-8859-16) representing different orthographic combinations (e.g. ISO-8859-7 for Latin/Greek)

The Limitations of Single-Byte Encodings

- Single-byte encodings such as ISO-8859-* are great if the alphabet is small, but they aren't able to support "big" orthographies such as Chinese, Japanese and Korean (CJK)
- Also, what if we want to have Greek, Hebrew and Arabic in the same document?

Take 2: Variable-width Encodings

- Encode the code points using a variable number of "code units" of fixed size
- The most popular variable-width Unicode encodings are:
  - **UTF-8**: code unit = 8 bits (1 byte)
  - **UTF-16**: code unit = 16 bits (2 bytes)

UTF-8

- UTF-8 is an 8-bit, variable-width encoding, which is compatible with ASCII

Code range (hex)	UTF-8 (bin)
0x000000–0x00007F	0xxxxxxx
0x000080–0x0007FF	110xxxxx 10xxxxxx
0x000800–0x00FFFF	1110xxxx 10xxxxxx 10xxxxxx
0x010000–0x10FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

(i.e. a UTF-8 document containing all code points in the range 0x00–0x7F is identical to an ASCII document)

Features of UTF-8

- Superset of ASCII
- Standard encoding within XML and for HTML5 (and modern operating systems)
- The byte sequence for a given code point never occurs as part of a longer sequence
- Character boundaries are easily locatable

UTF-16

- Similar to UTF-8, except that the code unit is 16 bits rather than 8 bits
- Variable width, as can't fit all 1m+ code points into 16 bits; similar variable-width trick to UTF-8 for higher code points (with the maximum code point length being two code units = 32 bits)
- Not used very widely, but can lead to more compact representations for languages using code points in the 0x000800–0x00FFFF range (which map onto 3 bytes in UTF-8, but 2 bytes in UTF-16)

Sorting in Different Languages

- Sorting is a solved problem under Unicode, right? ... well, no, not for all languages
- Case of Korean:
  - individual Hangul characters are made up of (up to) three "jamo", in the form of the initial consonant (one of 16), vowel (one of 21), and final consonant (one of 28)
  - there is a sort order within each of these jamo sets, with the initial consonant being the primary key, the vowel the secondary key, and the final consonant the tertiary key
  - in Unicode, the first Hangul character is at position 44032_{10}
  - as such, the Unicode code point/sort order is defined by:

```
(((initial * 21 + vowel) * 28) + final) + 44032
```

Sorting in Different Languages

- Case of Japanese:
  - actual sort order in standard dictionaries = pronunciation, then radical/stroke count ... with the complication that most kanji characters have at least two pronunciations, so you have to know how to pronounce the word to know how to sort it(!)
  - in Unicode, the code point order is a hodge-podge in practice, in part because of being backwards compatible with pre-existing standards, and partly because of "unihan" (a unified set of ideographs from Japanese, Korean, and Chinese)
- Case of Chinese:
  - actual sort order in standard dictionaries can be based on pronunciation (e.g. pinyin) or stroke order
  - in Unicode, once again, the actual code point order is a mess

Declaring Character Encodings

- As we have seen, it is not necessarily immediately evident which encoding a given document is encoded in (e.g. consider the ISO-8859-* encodings)
- The primary ways of dealing with this are:
  1. manually specify the "character encoding" in the document, e.g. in the case of HTML:

```
<meta http-equiv="Content-Type"
               content="text/html; charset=iso8859-8" />
```

  2. automatically detect the character encoding, wrt compatibility with encoding standards, user preferences, statistical models, ...

Python and Unicode

- Mostly in Python3, you don't have to worry about character encodings, as strings are natively in Unicode (unlike Python2, which where the `str` type was ASCII only, and there was a separate `unicode` type for Unicode strings
- The only place to be a little bit careful is with file I/O, where if the file you are opening/writing to is not in utf-8, you need to specific the encoding, e.g.

```
>>> fp = open("greek.txt", encoding="iso-8859-7")
>>> doc = fp.read()
```

Python and Unicode

- You can enter Unicode characters either via direct entry, via their code point, or via the Unicode character name:

```
>>> "\U0001F415" == "\N{DOG}"
True
```

- You can encode/decode a string to/from a given encoding using the `encode()`/`decode()` methods:

```
>>> dog_utf8 = "\N{DOG}".encode('utf-8')
>>> print(dog_utf8)
b'\xf0\x9f\x90\x95'
>>> dog_utf8 == \N{DOG}
False
>>> dog_utf8.decode("utf-8") == \N{DOG}
True
```

Lecture Outline

1. Text representation
2. Ethics and Fairness

Ethics of Computing

- Why ethics all of a sudden?
  - computing can trivially be deployed at a global scale, and has real potential to change the world
  - computing can trivially be applied in all sorts of contexts over all sorts of data ... not everything that *can* be computed *should* be computed
- Examples of unethical applications that you could relatively easily implement based on your COMP10001 knowledge (with a bit of upskilling with the right libraries):
  - spam generators
  - DDoS attack systems to bring down web services
  - web crawlers to indiscriminately crawl the web

ACM Code of Ethics and Professional Conduct

1. Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing
2. Avoid harm
3. Be honest and trustworthy
4. Be fair and take action not to discriminate
5. Respect the work required to produce new ideas, inventions, creative works, and computing artefacts
6. Respect privacy
7. Honor confidentiality

Source(s): https://www.acm.org/code-of-ethics

Ethical Data Usage: Cambridge Analytica

- We briefly discussed Cambridge Analytica in the context of Project 1, but let's recap what happened:
  - Facebook app released, which when used, crawled all data of the individual, and all of their friends
  - 270k users used the app, generating a dataset of 87m Facebook users
  - that data was then used to place targeted ads, with the explicit intent of manipulating voter behaviour in the US presidential elections
- What terms of the ACM Code of Ethics did it violate?

Ethical Data Use

- Issues to consider in using *any* data:
  - **privacy**: who owns the data, what access do others have to it, and what does it expose about users both directly and indirectly?
  - **governance**: management of the availability, integrity, security, etc. of the data
  - **fairness**: respect the people behind the data, and ensure that all benefit equally
  - **shared benefit**: those who produce the data should benefit from it
  - **transparency**: be clear as to how the data is used, by whom, and for what

Lecture Summary

- What are Unicode code points and code units?
- What is the "ISO-8859" character encoding family, and how does it extend ASCII?
- What is the relationship between encodings such as `utf-8` and Unicode?
- What is the difference between utf-8 and utf-16?
- What are the general principles of the ACM Code of Ethics?
- What is dual use in the context of computing?
- What are the terms of ethical data use?

Dual Use

- In the context of computing, "dual use" refers to technologies that can equally be used for good or malicious purposes
- Examples from AI:
  - autonomous cars
  - face recognition
  - machine translation
  - video generation ("deep fake")

Bias and Fairness

- Computing and AI is heavily data driven, meaning it is highly susceptible to data bias
- AI systems often not only capture data bias, but tend to amplify it:
  - image labelling
  - gender assignment in image captioning systems
  - speech recognition
  - medical diagnosis
- Important to both quantify and mitigate bias to achieve fairness
- So what? critically important that AI doesn't favour a particular demographic in society, exacerbate stereotypes/reinforce stigmas, for which we need to develop more sophisticated understanding of our datasets, training algorithms, better