# Informatics 225
# Computer Science 221

# Information Retrieval

Lecture 10



*These course materials borrow, with permission, from those of Prof. Cristina Videira Lopes, Addison Wesley 2008, Chris Manning, Pandu Nayak, Hinrich Schütze, Heike Adel, Sascha Rothe, Jerome H. Friedman, Robert Tibshirani, and Trevor Hastie. Powerpoint theme by Prof. André van der Hoek.*

# Text pre-processing

Information Retrieval

# Conversion

- Text is stored in hundreds of incompatible file formats
  - e.g., raw text, RTF, HTML, XML, Microsoft Word, ODF, PDF

# Conversion

- Text is stored in hundreds of incompatible file formats
  - e.g., raw text, RTF, HTML, XML, Microsoft Word, ODF, PDF
- Other types of files, not dedicated to text, are also important
  - e.g., PowerPoint, Excel, Keynote, Pages

# Conversion

- Text is stored in hundreds of incompatible file formats
  - e.g., raw text, RTF, HTML, XML, Microsoft Word, ODF, PDF
- Other types of files, not dedicated to text, are also important
  - e.g., PowerPoint, Excel, Keynote, Pages
- Typically use a conversion tool or library
  - converts the document content into a tagged text format such as HTML or XML
  - retains some of the important formatting information
  - Some converters will not retain formatting (e.g. *nix's pdftotxt, pdf2ps+pdf2ascii, etc.)

# Character Encoding

- Glyphs (γλυφή, carving): the symbols that you see, that encode letters or meanings

# Character Encoding

- Glyphs (γλυφή, carving): the symbols that you see, that encode letters or meanings

- A **character encoding** is a **mapping** between **bits** and **glyphs**
  - i.e., getting from bits in a file to characters on a screen
  - Can be a major source of incompatibility

- ASCII is basic character encoding scheme for English (since 1963)
  - Encodes 128 letters, numbers, special characters, and control characters in 7 bits, extended with an extra bit for storage in bytes

# Character Encoding

- Other languages can have many more glyphs
  - e.g., Chinese > 40,000 characters, with over 3,000 in common use

- Many languages have multiple encoding schemes
  - e.g., CJK (Chinese-Japanese-Korean) family of East Asian languages, Hindi, Arabic
  - must specify encoding in the file : the file is not self-describing
  - can't have multiple languages in one file

# Character Encoding

- Other languages can have many more glyphs
  - e.g., Chinese > 40,000 characters, with over 3,000 in common use
- Many languages have multiple encoding schemes
  - e.g., CJK (Chinese-Japanese-Korean) family of East Asian languages, Hindi, Arabic
  - must specify encoding in the file : the file is not self-describing
  - can't have multiple languages in one file
- Unicode was developed in the late 80's to address encoding problems

# Unicode

- Single mapping from numbers to glyphs that attempts to include all glyphs in common use in all known languages
  - Using reserved area of Unicode, you can even write 𓀀𓀁𓀂𓀃𓀄 𓀅𓀆𓀇

Wikipedia

# Unicode

- Single mapping from numbers to glyphs that attempts to include all glyphs in common use in all known languages
  - Using reserved area of Unicode, you can even write 𓀀𓀁𓀂𓀃𓀄 𓀅𓀆𓀇

- Unicode is a mapping between numbers and glyphs
  - solves the problem of mapping from number to glyphs
  - does not uniquely specify bits to glyph mapping
  - e.g., UTF-8, UTF-16, UTF-32

Wikipedia

# Unicode

- Proliferation of encodings comes from a need for compatibility and to save space
  - UTF-8 uses one byte for English (=ASCII), but as many as 4 bytes for some traditional Chinese characters
    - variable length encoding, more difficult to do string operations
  - UTF-32 uses 4 bytes for every character
    - more memory, no backward compatibility with ASCII

- You can mix both in your design
  - Many applications use UTF-32 for internal text encoding (fast random lookup) and UTF-8 for disk storage (less space)

# Unicode : encoding table

| Decimal | Hexadecimal | Encoding | | | |
|---|---|---|---|---|---|
| 0–127 | 0–7F | 0xxxxxxx | | | |
| 128–2047 | 80–7FF | 110xxxxx | 10xxxxxx | | |
| 2048–55295 | 800–D7FF | 1110xxxx | 10xxxxxx | 10xxxxxx | |
| 55296–57343 | D800–DFFF | Undefined | | | |
| 57344–65535 | E000–FFFF | 1110xxxx | 10xxxxxx | 10xxxxxx | |
| 65536–1114111 | 10000–10FFFF | 11110xxx | 10xxxxxx | 10xxxxxx | 10xxxxxx |

- e.g., Greek letter pi ($\pi$) is Unicode symbol number 960
- In binary, 00000011 11000000 (3C0 in hexadecimal)
- 960 at the second line: the encoding will follow **110**xxxxx **10**xxxxxx
- The high 5 bits go into the first byte and the next 6 bits in the second byte
- Final encoding is **110**01111 **10**000000 (CF80 in hexadecimal)