Processing Text

22 January 2024
Assignment o: Essentials

Due before the start of class on Wednesday
Where are we?
Our textbook is called *Speech and Language Processing*, but this class will be focused on text.

Why might we do that?
Corpus | 'kɔrpus |
noun (plural corpora | 'kɔrpəra | or corpuses)

1 a collection of written texts, especially the entire works of a particular author or a body of writing on a particular subject: the Darwinian corpus.
   • a collection of written or spoken material in machine-readable form, assembled for the purpose of studying linguistic structures, frequencies, etc.

2 Anatomy the main body or mass of a structure.
   • the central part of the stomach, between the fundus and the antrum.

Origin

late Middle English (denoting a human or animal body): from Latin, literally ‘body’. corpus (sense 1) dates from the early 18th century.
What *isn’t* a text corpus?
Text on computers
Encodings

mapping
ASCII
ASCII

Babar
By Shanaka Dias
telnet towel.blinkenlights.nl
ASCII

telnet towel.blinkenlights.nl
## ASCII

*American Standard Code for Information Interchange*

<table>
<thead>
<tr>
<th>Bits</th>
<th>Code point</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>01000001</td>
<td>65</td>
<td>&quot;A&quot;</td>
</tr>
<tr>
<td>01010011</td>
<td>83</td>
<td>&quot;S&quot;</td>
</tr>
<tr>
<td>01000011</td>
<td>67</td>
<td>&quot;C&quot;</td>
</tr>
<tr>
<td>01001001</td>
<td>73</td>
<td>&quot;I&quot;</td>
</tr>
<tr>
<td>01001001</td>
<td>73</td>
<td>&quot;I&quot;</td>
</tr>
<tr>
<td>Dec</td>
<td>Char</td>
<td>Dec</td>
</tr>
<tr>
<td>-----</td>
<td>--------------------</td>
<td>-----</td>
</tr>
<tr>
<td>0</td>
<td>NUL (null)</td>
<td>32</td>
</tr>
<tr>
<td>1</td>
<td>SOH (start of heading)</td>
<td>33</td>
</tr>
<tr>
<td>2</td>
<td>STX (start of text)</td>
<td>34</td>
</tr>
<tr>
<td>3</td>
<td>ETX (end of text)</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>EOT (end of transmission)</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>ENQ (enquiry)</td>
<td>37</td>
</tr>
<tr>
<td>6</td>
<td>ACK (acknowledge)</td>
<td>38</td>
</tr>
<tr>
<td>7</td>
<td>BEL (bell)</td>
<td>39</td>
</tr>
<tr>
<td>8</td>
<td>BS (backspace)</td>
<td>40</td>
</tr>
<tr>
<td>9</td>
<td>TAB (horizontal tab)</td>
<td>41</td>
</tr>
<tr>
<td>10</td>
<td>LF (NL line feed, new line)</td>
<td>42</td>
</tr>
<tr>
<td>11</td>
<td>VT (vertical tab)</td>
<td>43</td>
</tr>
<tr>
<td>12</td>
<td>FF (NP form feed, new page)</td>
<td>44</td>
</tr>
<tr>
<td>13</td>
<td>CR (carriage return)</td>
<td>45</td>
</tr>
<tr>
<td>14</td>
<td>SO (shift out)</td>
<td>46</td>
</tr>
<tr>
<td>15</td>
<td>SI (shift in)</td>
<td>47</td>
</tr>
<tr>
<td>16</td>
<td>DLE (data link escape)</td>
<td>48</td>
</tr>
<tr>
<td>17</td>
<td>DC1 (device control 1)</td>
<td>49</td>
</tr>
<tr>
<td>18</td>
<td>DC2 (device control 2)</td>
<td>50</td>
</tr>
<tr>
<td>19</td>
<td>DC3 (device control 3)</td>
<td>51</td>
</tr>
<tr>
<td>20</td>
<td>DC4 (device control 4)</td>
<td>52</td>
</tr>
<tr>
<td>21</td>
<td>NAK (negative acknowledge)</td>
<td>53</td>
</tr>
<tr>
<td>22</td>
<td>SYN (synchronous idle)</td>
<td>54</td>
</tr>
<tr>
<td>23</td>
<td>ETB (end of trans. block)</td>
<td>55</td>
</tr>
<tr>
<td>24</td>
<td>CAN (cancel)</td>
<td>56</td>
</tr>
<tr>
<td>25</td>
<td>EM (end of medium)</td>
<td>57</td>
</tr>
<tr>
<td>26</td>
<td>SUB (substitute)</td>
<td>58</td>
</tr>
<tr>
<td>27</td>
<td>ESC (escape)</td>
<td>59</td>
</tr>
<tr>
<td>28</td>
<td>FS (file separator)</td>
<td>60</td>
</tr>
<tr>
<td>29</td>
<td>GS (group separator)</td>
<td>61</td>
</tr>
<tr>
<td>30</td>
<td>RS (record separator)</td>
<td>62</td>
</tr>
<tr>
<td>31</td>
<td>US (unit separator)</td>
<td>63</td>
</tr>
</tbody>
</table>

These are all of the characters in ASCII.
1963  ASCII, a 7-bit standard


Examples: Latin-1, Latin/Cyrillic, Latin/Greek, Latin/Hebrew,

But CJKV languages need bigger character sets, e.g., Big5 (two bytes), GB 18030 and ISO 2022 (more complex).

How do you tell which one a file is using?
Today we mostly use *Unicode*. Unicode isn’t an encoding; it’s a listing of code points.

It can be encoded in different ways:

- UTF-32 is a 32-bit fixed-length encoding
- UTF-8 and UTF-16 are variable-length encodings
<table>
<thead>
<tr>
<th>Character</th>
<th>Code point</th>
<th>UTF-8 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>U+0041</td>
<td>01000001</td>
</tr>
<tr>
<td>ए</td>
<td>U+0A10</td>
<td>11100000 10101000 10010000</td>
</tr>
</tbody>
</table>
Announcing The Unicode® Standard, Version 15.0

Version 15.0 of the Unicode Standard is now available, including the core specification, annexes, and data files. This version adds 4,489 characters, bringing the total to 149,186 characters. These additions include two new scripts, for a total of 161 scripts, along with 20 new emoji characters, and 4,193 CJK (Chinese, Japanese, and Korean) ideographs. The new scripts and characters in Version 15.0 add support for modern language groups including:

- Nag Mundari, a modern script used to write Mundari, a language spoken in the eastern Congo.

WEB ARCHIVE

What is Unicode?
The Unicode Consortium
Archived Announcements
What's New in Unicode 15.0

The latest emoji list drafted by the Unicode Consortium is due for formal approval today, with new emojis including a Goose, a Hyacinth, a Shaking Face, and a plain Pink Heart.

Keith Broni

Sep 13, 2022 - 6 min read
Introduced by Unicode 12.1 in 2019
Computers have gotten better at typography, but they’ve made things more difficult for NLP.
Computers have gotten better at typography, but they’ve made things more difficult for NLP.
Things that can cause trouble:

- Contextual characters ("quotation marks" rather than "quotation marks")
- Newlines (LF, CR, or CRLF)
- Ligatures (scoff)
- Combining characters ( JWZ)
Introduction to Python development
#!/usr/bin/env python3

print("It's your friend, Emacs")
#!/usr/bin/env python3

print("It's your friend, Emacs")
M-x shell

M means the “meta” key, which is what you and I call “Escape”.

```python
#!/usr/bin/env python3
print("It's your friend, Emacs")
```
Other options:

vim
gedit
BBEdit
VSCode*
...

Other options:

- vim
- gedit
- BBEdit
- VSCode*

* I know this is the one you want to choose, but it makes things more complicated, because it uses its own Python runtime rather than what’s installed on the system. If you choose to use it, you’re on your own.
Jupyter notebooks (like Google Colab) are great, and they’re widely used in NLP and data science. Feel free to use them for experimentation or when you’re working on your final project, but – unless otherwise noted – the assignments you turn in should be normal, well-written Python files (.py), not notebook (.ipynb).
Demo: Files and encodings in Python
As an example of a file to read, we will use a relatively small, unannotated document from Project Gutenberg.

Project Gutenberg is an online collection of texts whose copyright has expired. It contains texts in many languages.

Download the First Project Gutenberg Collection of Edgar Allan Poe:

gutenberg.org/cache/epub/1062/pg1062.txt
with open("/Users/jgordon/Downloads/pg1062.txt") as f:
    for line in f:
        print(line)
Because we didn’t specify the encoding, Python will use its default.

What’s the default? 🤷

It depends on the system you’re running on!
with open("/Users/jgordon/Downloads/pg1062.txt", encoding="utf-8-sig") as f:
    for line in f:
        print(line)
Download gutenberg.org/ebooks/67094.txt.utf-8
with open("/Users/jgordon/Downloads/pg67094.txt",
    encoding="utf-8-sig") as f:
    for line in f:
        print(line)

Compare with

with open("/Users/jgordon/Downloads/pg67094.txt",
    encoding="latin1") as f:
    for line in f:
        print(line)
If we want to write a file, change the "r" mode to "w":

```python
with open("myoutfile.txt", "w") as f:
    print("First line of output", file=f)
    print("Second line of output", file=f)
    print("Here's a number:", 5, file=f)
```
Search
LITERARY CHARACTER APB, $400:

His victims include Charity Burbage, Mad Eye Moody & Severus Snape; he’d be easier to catch if you’d just name him!
BEATLES PEOPLE, $200:

“And anytime you feel the pain, hey” this guy
“refrain, don’t carry the world upon your shoulders”
In the 2004 opening ceremonies, a sole member of this team opened the Parade of Nations; the rest of his team closed it.
Some of these questions are really hard for people because you need to know so much trivia, and our meat minds are bad at that.

But computers can store a lot; Watson had 21.6 TB of storage, back in 2011.
LINUS’S LAW: “Given enough eyeballs, all bugs are shallow.”
“Given enough text, all questions are easy.”
Regular expressions in Python
RE functions to know:

re.search
re.match
re.finditer
re.findall
re.compile
re.sub
import re

s = "Hello there"

m = re.search(r"\b(t?here)\b", s)

print(m.group(1))
import re

s = "Hello there"

m = re.search(r"\b(t?here)\b", s)

print(m.group(1))
```python
import re

s = "Hello there"

m = re.match(r"\b(t?here)\b", s)

print(m.group(1))
```
import re

s = "Hello there"

m = re.match(r"\b(t?here)\b", s)

print(m.group(1))

Error! re.match only matches from the beginning of the string. It’s equivalent to starting the RE with ^.
import re

s = "Hello there, hello here, hello everywhere"

for m in re.finditer(r"\b(t?here)\b", s):
    print(m.group(1))
import re

s = "Hello there, hello here, hello everywhere"

for match in re.findall(r"\b(t?here)\b", s):
    print(match)
import re

s = "Hello there"

prog = re.compile(r"(Hello|howdy)"

m = prog.match("Hello there")
print(m.group(1))

m = prog.match("Howdy partner")
print(m.group(1))
import re

$s$ = "Hello there"

$prog$ = re.compile(r"(Hello|howdy)")

$m$ = prog.match("Hello there")
print($m$.group(1))

$m$ = prog.match("Howdy partner")
print($m$.group(1))
import re

s = "Hello there"

t = re.sub("(Hello|Hi) there", r"\1", s)

print(t)
Practice: Information Extraction
Output will be triples (entity1, relation, entity2), e.g.,

("Vassar College", "located in", "Poughkeepsie, NY")

("Grace Hopper", "born in", "1906")
“[Regular expressions] are particularly useful for searching in texts, when we have a pattern to search for and a corpus of texts to search through. A regular expression search function will search through the corpus, returning all texts that match the pattern.”

Jurafsky & Martin, § 2.1
Consider learning when people were born.
Consider learning when people were born.

What do you search for?
Let’s try some of these out!

As our corpus, we’ll look at an old snapshot of English Wikipedia.
born in \[0-9]\{4\}

Don’t want to match places or other descriptions, e.g.,

born in New York

born in poverty
#!/usr/bin/env python3

import fileinput
import re

prog = re.compile(r"((?:(?:[A-Z][a-z]+) )+)\(born .* ([0-9]{4})\)\")

for line in fileinput.input():
    m = prog.search(line)
    name = m.group(1).strip()
    year = m.group(2)
    print(f"("{name}", "born in", "{year}")")
born in ([0-9]{4}|[0-9]+ (AD|BC))
born in ([0-9]{4}|[0-9]+ (AD|BC|CE|BCE))
born on the 8th of May, 1885
Another pattern:

```
born on .+ [0-9]{4}
```
David Robert Jones (8 January 1947 – 10 January 2016), known professionally as David Bowie (/ˈboʊi/ BOH-ee),[1] was an English singer, songwriter, musician, and actor. He is regarded as one of the most influential musicians of the 20th century. Bowie was acclaimed by critics and musicians, particularly for his innovative work during the 1970s. His career was marked by reinvention and visual presentation, and his music and stagecraft had a significant impact on popular music.

Bowie developed an interest in music from an early age. He studied art, music and design before embarking on a professional career as a musician in 1963. He released a string of unsuccessful singles with local bands and a solo album before achieving his first top-five entry on the UK Singles Chart with “Space Oddity” (1969). After a period of experimentation, he re-emerged in 1972 during the glam rock era with the flamboyant and androgynous alter ego Ziggy Stardust. The character was spearheaded by the success of "Starman" and album The Rise and Fall of Ziggy Stardust and the Spiders from Mars, which won him widespread popularity. In 1975, Bowie's style shifted towards a sound he referred to as "plastic soul".
We could keep going!
These kind of searches let us learn lots of information that’s stated in text.

Which companies bought which other companies.
What state is a town in?
Which musicians made which albums?
As we work on this information retrieval program, we’ve been trying to fix two kinds of errors:

1. **False positives**: Matching strings that we shouldn’t have matched (e.g., *born in humble circumstances*)

2. **False negatives**: Not matching things that we should have matched (e.g., *born on the first of January, 1901*)
Error types

Program thinks it says when someone was born

- Actually says when someone was born: True positive
- Doesn’t actually say when someone was born: False positive

Program thinks it doesn’t say when someone was born

- Actually says when someone was born: False negative
- Doesn’t actually say when someone was born: True negative
In NLP, we’re always dealing with these kinds of errors.

Reducing the error rate for an application often involves two antagonistic efforts:

- Increasing accuracy or precision (minimizing false positives)
- Increasing coverage or recall (minimizing false negatives)
Acknowledgments

This class incorporates material from:

Carolyn Anderson, Wellesley College
Katrin Erk, University of Texas at Austin
Katie Keith, Williams College
Xanda Schofield, Harvey Mudd College
Jurafsky & Martin, *Speech and Language Processing*, 3rd ed. draft