Lecture 11: Social Media and Web Mining, Format Conversion

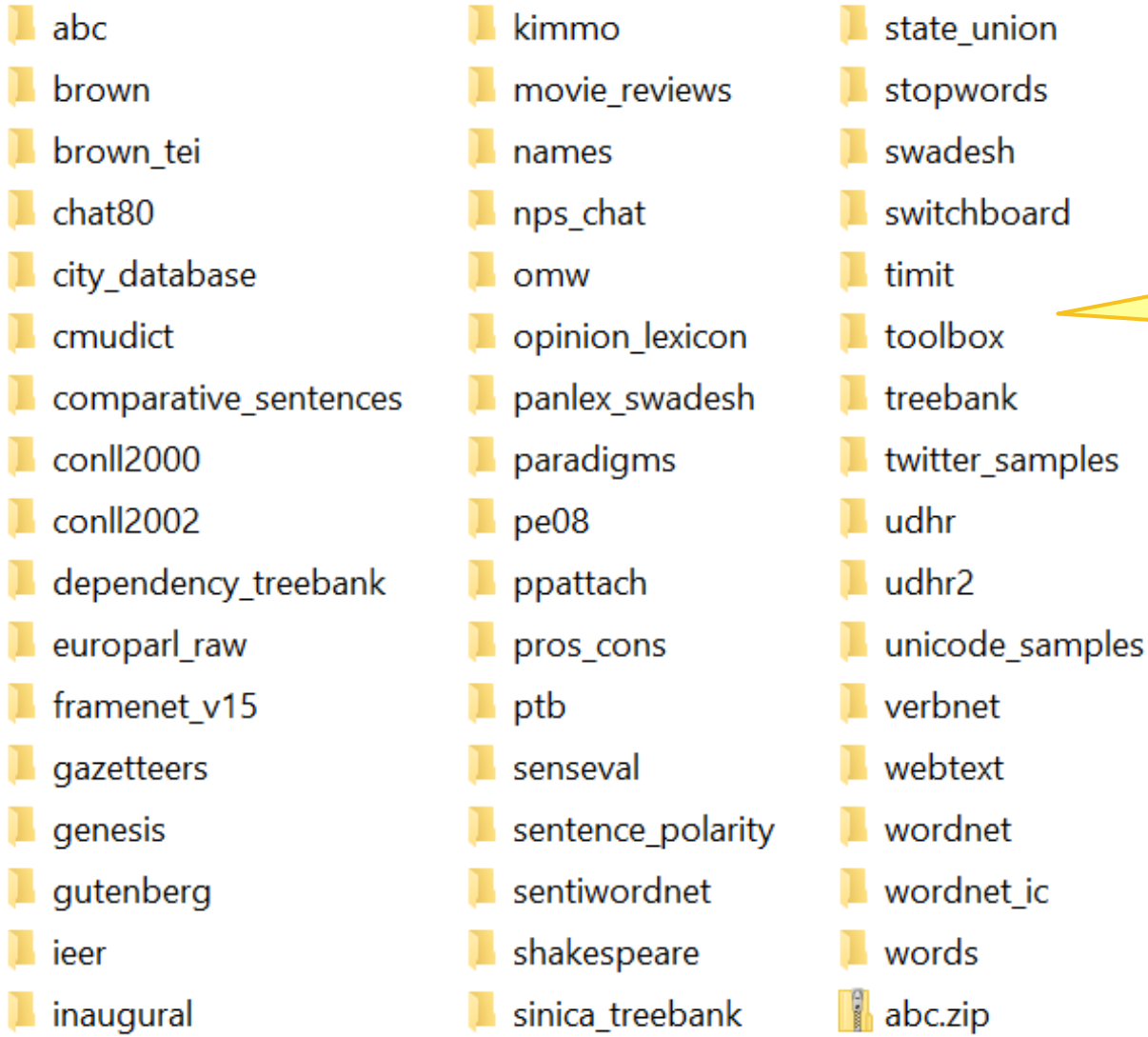
LING 1340/2340: Data Science for Linguists

Na-Rae Han

Objectives

- Corpus data: standard and popular formats
  - Review of common data formats
  - Conversion: iconv (encoding), dos2unix and unix2dos (line ending)
- Web and social media mining
  - Web pages: processing HTML
  - Social media mining: Twitter and JSON

A brief tour of NLTK's many "corpus" data



Many of them are language data, not corpora per se

Diverse genres and data formats represented!

Resource-specific (ad-hoc) formats

- Brown corpus

```
The/at Fulton/np-tl County/nn-tl Grand/jj-tl Jury/nn-tl said/vbd
Friday/nr an/at investigation/nn of/in Atlanta's/np$ recent/jj
primary/nn election/nn produced/vbd ``/`` no/at evidence/nn ''/''
that/cs any/dti irregularities/nns took/vbd place/nn ./.
```

- Korean Treebank corpus:

```
;;05:127: 저는 그 일을 할 수 있는 한 빨리 하겠습니다 .

(S (NP-SBJ 저/NPN+는/PAU)
   (VP (NP-OBJ-LV 그/DAN
                  일/NNC+을/PCA)
       (VP (NP-ADV (S (NP-SBJ (S (NP-SBJ *pro*)
                                 (VP 하/VV+ㄹ/EAN))
                              (NP 수/NNX))
                         (ADJP 있/VJ+는/EAN))
                     (NP 한/NNX))
           (ADVP 빨리/ADV)
           (VP (LV 하/VV+겠/EPF+습니다/EFN))))
   ./SFN)
```

NOT standard
(cf. XML, JSON).
Project-dependent.

It is up to end users to understand the data format, then write code to parse data files.

Refer to documentation!

Also: Python libraries may already exist

Dependency annotation: format

- https://raw.githubusercontent.com/UniversalDependencies/UD_English-EWT/dev/en_ewt-ud-dev.conllu

```
# sent_id = weblog-blogspot.com_nominations_20041117172713_ENG_20041117_172713-0002
# newpar id = weblog-blogspot.com_nominations_20041117172713_ENG_20041117_172713-p0002
# text = President Bush on Tuesday nominated two individuals to replace retiring jurists on federal courts in the Washington area.
1	President	President	PROPN	NNP	Number=Sing	5	nsubj	5:nsubj	_
2	Bush	Bush	PROPN	NNP	Number=Sing	1	flat	1:flat	_
3	on	on	ADP	IN	_	4	case	4:case	_
4	Tuesday	Tuesday	PROPN	NNP	Number=Sing	5	obl	5:obl:on	_
5	nominated	nominate	VERB	VBD	Mood=Ind|Tense=Past|VerbForm=Fin	0	root	0:root	_
6	two	two	NUM	CD	NumType=Card	7	nummod	7:nummod	_
7	individuals	individual	NOUN	NNS	Number=Plur	5	obj	5:obj	_
8	to	to	PART	TO	_	9	mark	9:mark	_
9	replace	replace	VERB	VB	VerbForm=Inf	5	advcl	5:advcl:to	_
10	retiring	retire	VERB	VBG	VerbForm=Ger	11	amod	11:amod	_
11	jurists	jurist	NOUN	NNS	Number=Plur	9	obj	9:obj	_
12	on	on	ADP	IN	_	14	case	14:case	_
13	federal	federal	ADJ	JJ	Degree=Pos	14	amod	14:amod	_
14	courts	court	NOUN	NNS	Number=Plur	11	nmod	11:nmod:on	_
15	in	in	ADP	IN	_	18	case	18:case	_
16	the	the	DET	DT	Definite=Def|PronType=Art	18	det	18:det	_
17	Washington	Washington	PROPN	NNP	Number=Sing	18	compound	18:compound	_
18	area	area	NOUN	NN	Number=Sing	14	nmod	14:nmod:in	SpaceAfter=No
19	.	.	PUNCT	.	_	5	punct	5:punct	_
```

Known as the **CoNLL-U format**
https://universaldependencies.org/format.html

Do not re-invent the wheel.

- If you can, avoid parsing them manually!
- There are Python libraries. Import and use them.
  - CSV & TSV: `pandas`
  - HTML & XML: Beautiful Soup (`bs4`)
  - JSON:
    - `json` library
    - `pandas.read_json`
- NLP-specific formats (Treebank, Universal Dependency, CoNLL):
  - Look at NLTK, see if it has reader
  - If not, chances are there is parser library written by someone somewhere (likely on GitHub)

They are all TEXT files.

- Underneath it all, these files are all TEXT files with **special formatting syntax** and **special characters** designated for formatting purposes.
  - In command line, you can `cat` and `less` through the files. Also: `head`, `tail`
  - You can open them up in a **text editor** (Atom, Notepad++) and edit.
  - Some editors/applications are aware of format-specific syntax and will display **rendered view** (cf. "source view"). Some will apply **syntax highlighting**.
    - Unlike, say, PDF files, style attributes are NOT part of the source data files themselves. (e.g., markdown file)
- As TEXT files, they have two important aspects:
  - **Encoding**: Latin-1 (=ISO-8859-1), ASCII, UTF-8, UTF-16, CP-1252 (Windows-1252), ANSI...
  - **Line endings**: **LF** (`'\n'`: OS X & Linux), **CRLF** (`'\r\n'`: Windows)

Encoding conversion

- **Encoding**: Latin-1 (=ISO-8859-1), ASCII, UTF-8, UTF-16, CP-1252 (Windows-1252), ANSI...
  - Common conversion tool: `iconv`

MINGW64:/c/Users/narae/Desktop/gutenberg

```
narae@T480s MINGW64 ~/Desktop/gutenberg
$ which iconv
/usr/bin/iconv

narae@T480s MINGW64 ~/Desktop/gutenberg
$ iconv -f ASCII -t UTF-16 bible-kjv.txt > bible-kjv.UTF16.txt

narae@T480s MINGW64 ~/Desktop/gutenberg
$ ls -lh bible*
-rw-r--r-- 1 narae 197121 8.3M Feb 15 11:22 bible-kjv.UTF16.txt
-rw-r--r-- 1 narae 197121 4.2M Feb 13 08:54 bible-kjv.txt

narae@T480s MINGW64 ~/Desktop/gutenberg
$ file bible*
bible-kjv.UTF16.txt: Big-endian UTF-16 Unicode text
bible-kjv.txt:       ASCII text
```

`iconv`
to create a new UTF-16 encoded version of the bible file.

UTF-16 means double the file size!

Line-ending conversion

- **Line endings**: **LF** (`'\n'`: OS X & Linux), **CRLF** (`'\r\n'`: Windows)
  - When you read in a CRLF text file in Python, it automatically convert `'\r\n'` to `'\n'`. It reverts (back) to CRLF when writing out.
  - When you git-commit a CRLF text file, git will change `'\r\n'` to `'\n'` by default.
  - Line-ending conversion tool: `dos2unix` and `unix2dos`

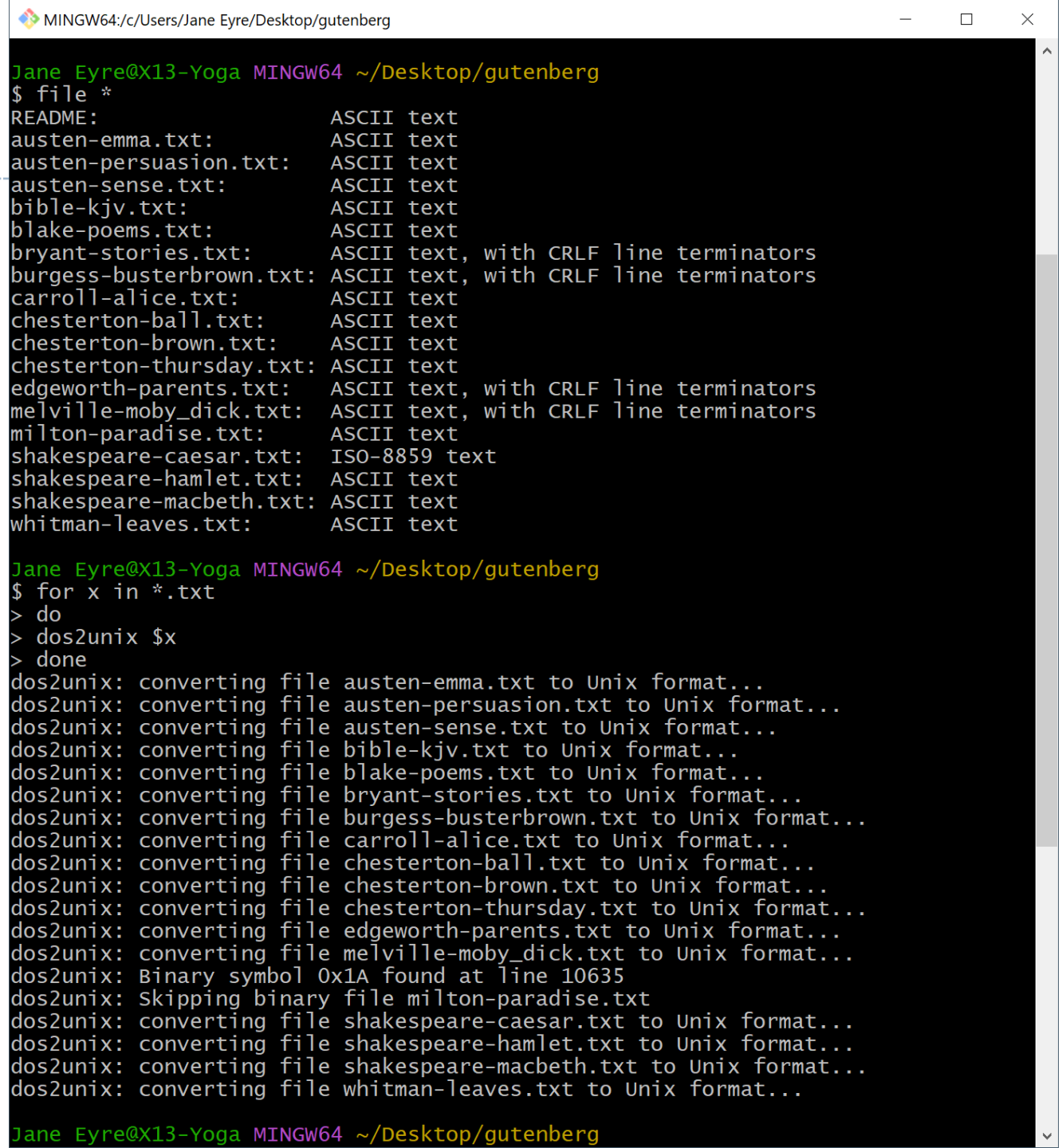
```
narae@X13-Yoga MINGW64 /c/Users/narae/Desktop/gutenberg
$ file melville-moby_dick.txt
melville-moby_dick.txt: ASCII text, with CRLF line terminators

narae@X13-Yoga MINGW64 /c/Users/narae/Desktop/gutenberg
$ wc melville-moby_dick.txt
  22924  212030 1242990 melville-moby_dick.txt

narae@X13-Yoga MINGW64 /c/Users/narae/Desktop/gutenberg
$ dos2unix melville-moby_dick.txt
dos2unix: converting file melville-moby_dick.txt to Unix format...

narae@X13-Yoga MINGW64 /c/Users/narae/Desktop/gutenberg
$ file melville-moby_dick.txt
melville-moby_dick.txt: ASCII text

narae@X13-Yoga MINGW64 /c/Users/narae/Desktop/gutenberg
$ wc melville-moby_dick.txt
  22924  212030 1220066 melville-moby_dick.txt
```

`dos2unix` to covert a file from CRLF to LF

Notice anything different?

- Batch processing in command line
  - Using for-loop in bash:

```
for x in *.txt
do
dos2unix $x
done
```



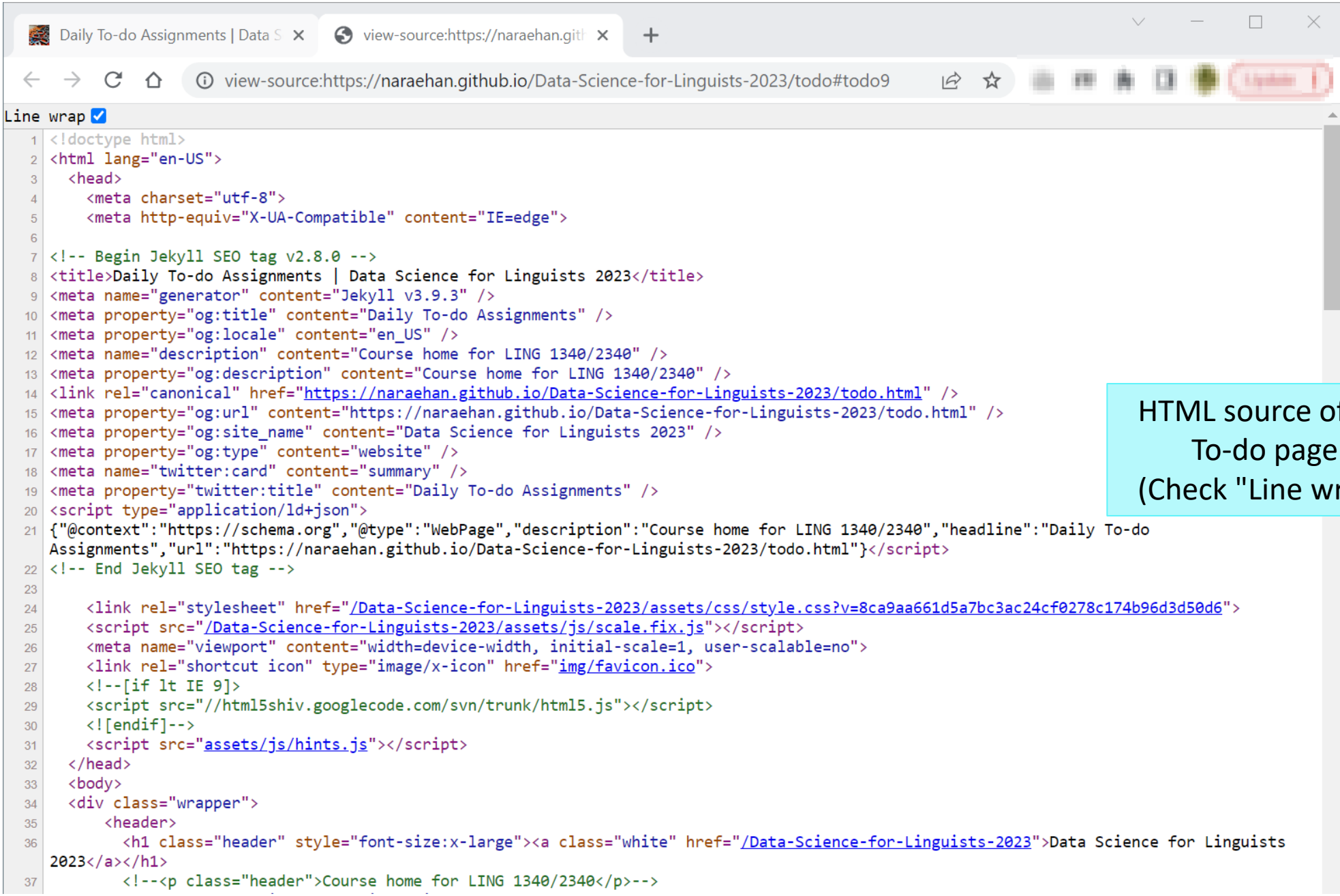
Format conversion: summary

- For some basic aspects (line ending, encoding) your text editor (Notepad++, VS Code, ...) will have built-in menus for converting individual files.
- When dealing with corpora, you may need to convert 100+ files at once.
  - On-line services are too cumbersome.
  - Try batch-processing through command line.
- Automatic tools available on command line.
  - Finding out text file encoding, line ending: `file` command (also `file -i`)
  - Encoding conversion: `iconv` (Linux, OS X, on Git Bash)
  - Line ending conversion: `unix2dos`, `dos2unix`
  - **Pandoc** https://www.pandoc.org/
    - Universal document converter
    - HTML, XML, PDF, LaTeX, Markdown, Epub, MS Doc, ...
    - After installation, you can use it via command line

Web mining

- Involves "web crawling" "web spyder", ...
- `scrapy` is the most popular library.
  - https://scrapy.org/

  ← You will have to install it first.
- You have collected a set of web pages. Now what?
  - A web page typically has tons of non-text, extraneous data such as headers, scripts, etc.
    - Example: https://naraehan.github.io/Data-Science-for-Linguists-2025/todo
  - You will need to parse each page to extract textual data.
  - Beautiful Soup (bs4) is capable of parsing XML and HTML files.
- OK, so you've processed the web pages as data. Now what?
  - Linguistic analysis?



HTML source of our To-do page. (Check "Line wrap")

Data-mining web & social media

- Twitter sample corpus
  - Static corpus: download from the NLTK data page
- How does one data-mine Twitter (= "X")?
  - Answer: through **API** (Application Program Interface)
  - Getting acquainted with JSON format
  - Tutorials on the Learning Resource page
- Libraries used: `tweepy`, `json`

If you can pay for it…
(RIP free Twitter API)

Processing a static Twitter corpus

- "Twitter Samples" corpus can be downloaded from http://www.nltk.org/nltk_data/

In [3]:
```
# One json object per line
jfile = 'D:/Corpora/twitter_samples/positive_tweets.json'
jlines = open(jfile).readlines()
jlines[0]
```

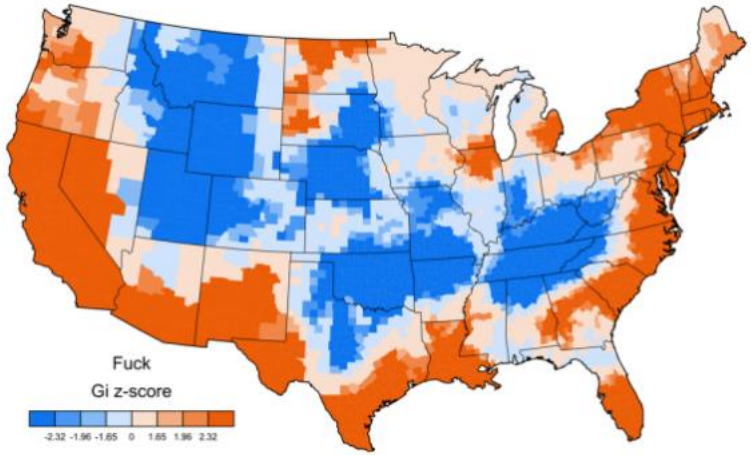
Out[3]:
```
'{"contributors": null, "coordinates": null, "text": "#FollowFriday @France_Int
e @PKuchly57 @Milipol_Paris for being top engaged members in my community this
 week :)", "user": {"time_zone": "Paris", "profile_background_image_url": "htt
```

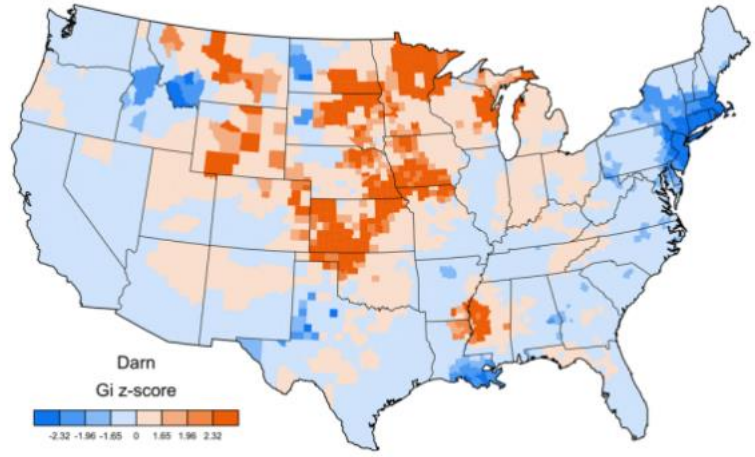
In [5]:
```
# using json library to read line.
import json
json.loads(jlines[0])
```

Out[5]:
```
{'contributors': None,
 'coordinates': None,
 'created_at': 'Fri Jul 24 08:23:36 +0000 2015',
 'entities': {'hashtags': [{'indices': [0, 13], 'text': 'FollowFriday'}],
  'symbols': [],
  'urls': [],
  'user_mentions': [{'id': 3222273608,
    'id_str': '3222273608',
    'indices': [14, 26],
    'name': 'France International',
```

Mining social media for swear words

- https://stronglang.wordpress.com/2015/07/28/mapping-the-united-swears-of-america/
  - Jack Grieve mined Twitter and mapped prominent swear words by geographic regions within the US





Wrapping up

- To-do 12 out
  - Starting on machine learning: `sklearn`
- Next week
  - New topic: machine learning
- Your project: 1st Progress Report
  - Due date moved to next Wednesday
  - Data work typically takes MUCH LONGER! Start NOW.