Introduction to Corpora

Elliott Ash

Max Planck Summer School 2017

Overview

- These slides describe the process of getting a corpus of written language.
- Input:
  - A set of documents (e.g. text files), D.
- Output:
  - A matrix, X, containing statistics about phrase frequencies in those documents.

Text as Data

- Text data is a sequence of characters called **documents**.
- The set of documents is the **corpus**.
- Text data is **unstructured**:
  - the information we want is mixed together with (lots of) information we don't.
  - How to separate the two?
- All text data approaches will throw away some information:
  - The trick is figuring out how to retain valuable information.

Text as Data

- Text data is a sequence of characters called **documents**.
- The set of documents is the **corpus**.

- Text data is **unstructured**:
  - the information we want is mixed together with (lots of) information we don't.
  - How to separate the two?
- All text data approaches will throw away some information:
  - The trick is figuring out how to retain valuable information.

Text as Data

- Text data is a sequence of characters called **documents**.
- The set of documents is the **corpus**.

- Text data is **unstructured**:
  - the information we want is mixed together with (lots of) information we don't.
  - How to separate the two?
- All text data approaches will throw away some information:
  - The trick is figuring out how to retain valuable information.

Outline

Publicly Available Corpora

- There is already a vast amount of data out there that has already been compiled (e.g. CourtListener, Twitter, New York Times, Reuters, Google, Wikipedia).
- Chris Bail curates a list of these datasets:
  - `https://docs.google.com/spreadsheets/d/1I7cvuCBQxosQK2evTcdL3qtglaEPc0WFEs6rZMx-xiE/edit`
- Many proprietary corpora are becoming available for research:
  - Lexis
  - Web of Science

Publicly Available Corpora

- There is already a vast amount of data out there that has already been compiled (e.g. CourtListener, Twitter, New York Times, Reuters, Google, Wikipedia).
- Chris Bail curates a list of these datasets:
  - `https://docs.google.com/spreadsheets/d/1I7cvuCBQxosQK2evTcdL3qtglaEPc0WFEs6rZMx-xiE/edit`
- Many proprietary corpora are becoming available for research:
  - Lexis
  - Web of Science

Screen Scraping

- A screen scraper is a computer program that:
  - loads/reads in a web page
  - finds some information on it
  - grabs the information
  - stores it in a dataset
- Once upon a time you could collect virtually any piece of information from the internet by screen scraping.
  - But now web sites make it difficult with restrictive terms of use, bot-blockers, javascript, etc.
  - Still, a little creativity goes a long way.

Screen Scraping

- A screen scraper is a computer program that:
  - loads/reads in a web page
  - finds some information on it
  - grabs the information
  - stores it in a dataset
- Once upon a time you could collect virtually any piece of information from the internet by screen scraping.
  - But now web sites make it difficult with restrictive terms of use, bot-blockers, javascript, etc.
  - Still, a little creativity goes a long way.

What a web site looks like to us

What a web site looks like to a computer

```
<!DOCTYPE html>
<html lang="en" dir="ltr" class="client-nojs">
<head>
<meta charset="UTF-8" />
<title>World Health Organization ranking of health systems in 2000 - Wikipedia, the free encyclopedia</title>
<meta name="generator" content="MediaWiki 1.26wmf10" />
<link rel="alternate" href="android-app://org.wikipedia/http/en.m.wikipedia.org/wiki/World_Health_Organization_ranking_of_health_systems_in_2000" />
<link rel="alternate" type="application/x-wiki" title="Edit this page" href="/w/index.php?title=World_Health_Organization_ranking_of_health_systems_in_2000&amp;action=edit" />
<link rel="edit" title="Edit this page" href="/w/index.php?title=World_Health_Organization_ranking_of_health_systems_in_2000&amp;action=edit" />
<link rel="apple-touch-icon" href="/static/apple-touch/wikipedia.png" />
<link rel="shortcut icon" href="/static/favicon/wikipedia.ico" />
<link rel="search" type="application/opensearchdescription+xml" href="/w/opensearch_desc.php" title="Wikipedia (en)" />
<link rel="EditURI" type="application/rsd+xml" href="//en.wikipedia.org/w/api.php?action=rsd" />
<link rel="alternate" hreflang="x-default" href="/wiki/World_Health_Organization_ranking_of_health_systems_in_2000" />
<link rel="copyright" href="//creativecommons.org/licenses/by-sa/3.0/" />
<link rel="alternate" type="application/atom+xml" title="Wikipedia Atom feed" href="/w/index.php?title=Special:RecentChanges&amp;feed=atom" />
<link rel="canonical" href="https://en.wikipedia.org/wiki/World_Health_Organization_ranking_of_health_systems_in_2000" />
<link rel="stylesheet" href="//en.wikipedia.org/w/load.php?debug=false&amp;lang=en&amp;modules=ext.uls.nojs%7Cext.visualEditor.viewPageTarget.noscript%7Cext.wikihiero%7Cmediawiki.legacy.commonPrint%2Cshared%7Cmediawiki.sectionAnchor%7Cmediawiki.skinning.interface%7Cmediawiki.ui.button%7Cskins.vector.styles%7Cwikibase.client.init&amp;only=styles&amp;skin=vector&amp;*" />
<meta name="ResourceLoaderDynamicStyles" content="" />
<link rel="stylesheet" href="//en.wikipedia.org/w/load.php?debug=false&amp;lang=en&amp;modules=site&amp;only=styles&amp;skin=vector&amp;*" />
ttps://en.wikipedia.org/w/index.php?title=World_Health_Organization a:lang(mzn),a:lang(ps),a:lang(ur){text-decoration:none}
```

Let's try it: Screen scraping in Python

```
# load a web page
import urllib
url = 'https://goo.gl/VRF8Xs'
page = urllib.request.urlopen(url)


# read its contents as a string
html = page.read()
print(html[:400])
print(html[-400:])
print(len(html))
```

Let's try it: Screen scraping in Python

```
# load a web page
import urllib
url = 'https://goo.gl/VRF8Xs'
page = urllib.request.urlopen(url)


# read its contents as a string
html = page.read()
print(html[:400])
print(html[-400:])
print(len(html))
```

HTML parsing

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')
print(soup.title)
text = soup.get_text()
lines = text.splitlines()
print(len(lines))
# drop empty lines
lines = [line for line in lines if line != ”]
print(len(lines))
print(lines[:20])
```

HTML parsing

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')
print(soup.title)
text = soup.get_text()
lines = text.splitlines()
print(len(lines))
# drop empty lines
lines = [line for line in lines if line != ”]
print(len(lines))
print(lines[:20])
```

HTML parsing

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')
print(soup.title)
text = soup.get_text()
lines = text.splitlines()
print(len(lines))
# drop empty lines
lines = [line for line in lines if line != ”]
print(len(lines))
print(lines[:20])
```

HTML parsing

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')
print(soup.title)
text = soup.get_text()
lines = text.splitlines()
print(len(lines))
# drop empty lines
lines = [line for line in lines if line != ”]
print(len(lines))
print(lines[:20])
```

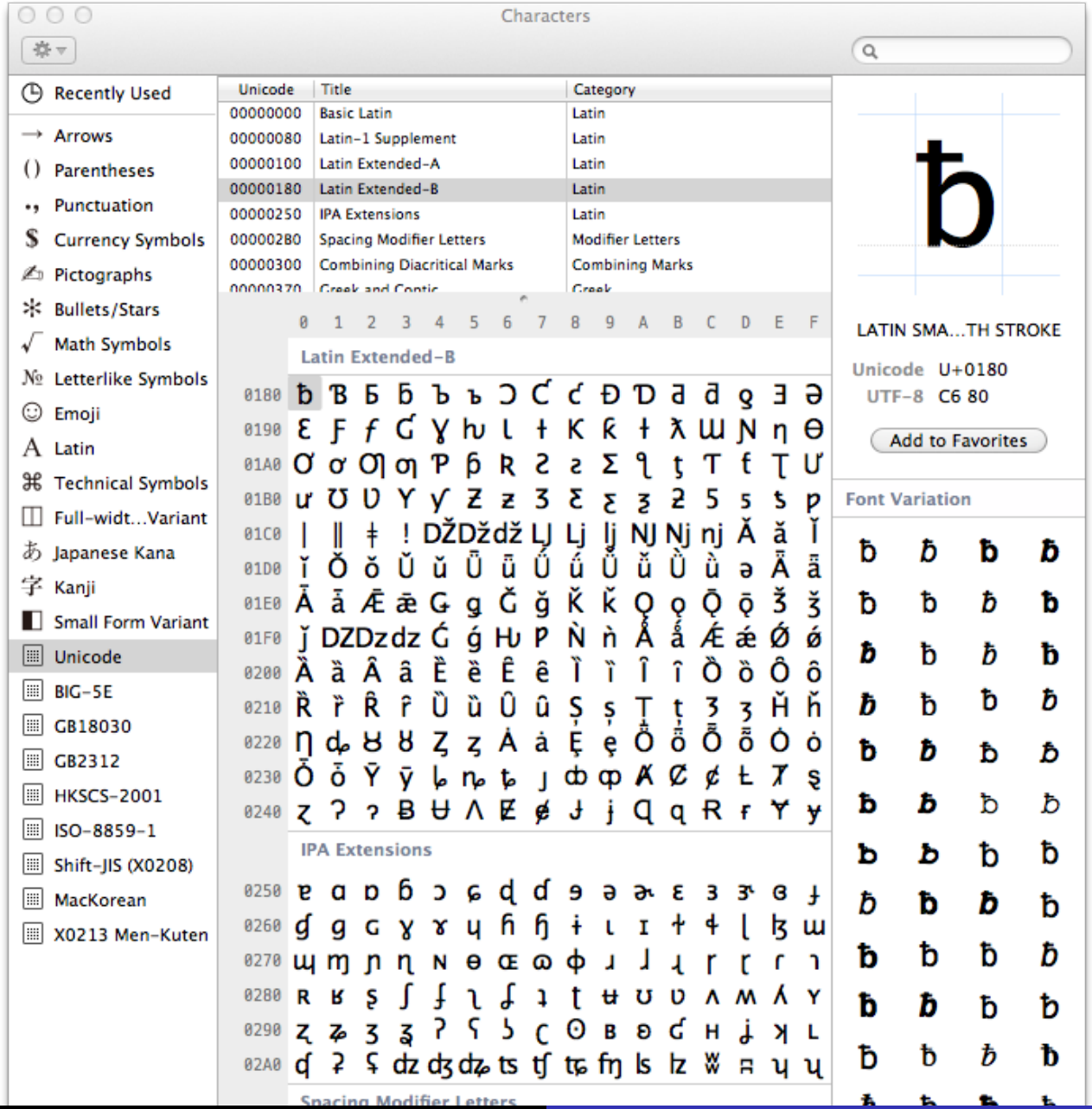
HTML parsing

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')
print(soup.title)
text = soup.get_text()
lines = text.splitlines()
print(len(lines))
# drop empty lines
lines = [line for line in lines if line != ”]
print(len(lines))
print(lines[:20])
```

HTML parsing

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')
print(soup.title)
text = soup.get_text()
lines = text.splitlines()
print(len(lines))
# drop empty lines
lines = [line for line in lines if line != ”]
print(len(lines))
print(lines[:20])
```

Character Encoding



Removing unicode characters

```
from unidecode import unidecode
fixed = unidecode('Visualizations\xa0')
print(fixed)
```

Corpus cleaning

- What we've already done:
  - removed HTML markup, extra white space, and unicode
- But HTML markup is often valuable:
  - HTML markup for section header names.
  - Legal database web sites often have HTML tags for citations to other cases.
- Other cleaning steps:
  - page numbers
  - hyphenations at line breaks
  - table of contents, indexes, etc.
- These are all corpus-specific, so inspect ahead of time.

Corpus cleaning

- What we've already done:
  - removed HTML markup, extra white space, and unicode
- But HTML markup is often valuable:
  - HTML markup for section header names.
  - Legal database web sites often have HTML tags for citations to other cases.
- Other cleaning steps:
  - page numbers
  - hyphenations at line breaks
  - table of contents, indexes, etc.
- These are all corpus-specific, so inspect ahead of time.

Corpus cleaning

- What we've already done:
  - removed HTML markup, extra white space, and unicode
- But HTML markup is often valuable:
  - HTML markup for section header names.
  - Legal database web sites often have HTML tags for citations to other cases.
- Other cleaning steps:
  - page numbers
  - hyphenations at line breaks
  - table of contents, indexes, etc.
- These are all corpus-specific, so inspect ahead of time.

Browser Automation

- Many web sites are designed to be difficult to scrape.
- Python has many solutions for simulating a human browser:
  - robobrowser
  - selenium (chromedriver, phantomjs)
- Other solutions if all else fails:
  - DownThemAll! plug-in for Firefox
  - Hire mechanical turkers to manually download data.

Browser Automation

- Many web sites are designed to be difficult to scrape.
- Python has many solutions for simulating a human browser:
  - robobrowser
  - selenium (chromedriver, phantomjs)
- Other solutions if all else fails:
  - DownThemAll! plug-in for Firefox
  - Hire mechanical turkers to manually download data.

API's

- API = Application Programming Interface
  - These are developer-oriented tools that provide access to cleaner data.
- Chris Bail's list of API's that could be interesting for research:
  - `https://docs.google.com/spreadsheets/d/1ZEr3okdlb0zctmX0MZKo-gZKPsq5WGn1nJOxPV7al-Q/edit`
- The example data set was obtained from the CourtListener API (`courtlistener.com/api`).

Outline

Documents and metadata

- For small corpora, you might have the text and metadata together in a spreadsheet.
- For larger corpora, you might have:
  - A document is a text file (or an item in a relational database).
  - A corpus is a folder of text files.
  - The filenames for the text files should contain an identifier for linking to metadata.

What counts as a document?

- The unit of document analysis will vary depending on your question.
- If you are looking at how judges decide different types of cases, then a case would be a document.
- If you are looking at how judges differ within a court, then you might aggregate all of a judge's cases as a document.
- If you are looking at the impact of court cases on crime in a year, you might aggregate all the cases in a single year as a single document.
- If you are looking at how different topics are discussed within single cases, then a document might be a section or a paragraph.

What counts as a document?

- The unit of document analysis will vary depending on your question.
- If you are looking at how judges decide different types of cases, then a case would be a document.
- If you are looking at how judges differ within a court, then you might aggregate all of a judge's cases as a document.
- If you are looking at the impact of court cases on crime in a year, you might aggregate all the cases in a single year as a single document.
- If you are looking at how different topics are discussed within single cases, then a document might be a section or a paragraph.

What counts as a document?

- The unit of document analysis will vary depending on your question.
- If you are looking at how judges decide different types of cases, then a case would be a document.
- If you are looking at how judges differ within a court, then you might aggregate all of a judge's cases as a document.
- If you are looking at the impact of court cases on crime in a year, you might aggregate all the cases in a single year as a single document.
- If you are looking at how different topics are discussed within single cases, then a document might be a section or a paragraph.

What counts as a document?

- The unit of document analysis will vary depending on your question.
- If you are looking at how judges decide different types of cases, then a case would be a document.
- If you are looking at how judges differ within a court, then you might aggregate all of a judge's cases as a document.
- If you are looking at the impact of court cases on crime in a year, you might aggregate all the cases in a single year as a single document.
- If you are looking at how different topics are discussed within single cases, then a document might be a section or a paragraph.

What counts as a document?

- The unit of document analysis will vary depending on your question.
- If you are looking at how judges decide different types of cases, then a case would be a document.
- If you are looking at how judges differ within a court, then you might aggregate all of a judge's cases as a document.
- If you are looking at the impact of court cases on crime in a year, you might aggregate all the cases in a single year as a single document.
- If you are looking at how different topics are discussed within single cases, then a document might be a section or a paragraph.

Iterating over documents in a dataframe

```
# set your working directory
import os
os.chdir('/home/elliott/Dropbox/text-project')
processed = {}
import pandas as pd
df = pd.read_csv('case-data.csv')
for i, row in df.iterrows():
    docid = row['cluster_id']
    text = row['snippet']
    document = process_document(text)
    processed[docid] = document
```

Iterating over documents in a dataframe

```
# set your working directory
import os
os.chdir('/home/elliott/Dropbox/text-project')
processed = {}
import pandas as pd
df = pd.read_csv('case-data.csv')
for i, row in df.iterrows():
    docid = row['cluster_id']
    text = row['snippet']
    document = process_document(text)
    processed[docid] = document
```

Iterating over documents in a dataframe

```
# set your working directory
import os
os.chdir('/home/elliott/Dropbox/text-project')
processed = {}
import pandas as pd
df = pd.read_csv('case-data.csv')
for i, row in df.iterrows():
    docid = row['cluster_id']
    text = row['snippet']
    document = process_document(text)
    processed[docid] = document
```

Iterating over documents in a dataframe

```
# set your working directory
import os
os.chdir('/home/elliott/Dropbox/text-project')
processed = {}
import pandas as pd
df = pd.read_csv('case-data.csv')
for i, row in df.iterrows():
    docid = row['cluster_id']
    text = row['snippet']
    document = process_document(text)
    processed[docid] = document
```

Iterating over documents in a dataframe

```
# set your working directory
import os
os.chdir('/home/elliott/Dropbox/text-project')
processed = {}
import pandas as pd
df = pd.read_csv('case-data.csv')
for i, row in df.iterrows():
    docid = row['cluster_id']
    text = row['snippet']
    document = process_document(text)
    processed[docid] = document
```

Iterating over documents in a dataframe

```
# set your working directory
import os
os.chdir('/home/elliott/Dropbox/text-project')
processed = {}
import pandas as pd
df = pd.read_csv('case-data.csv')
for i, row in df.iterrows():
    docid = row['cluster_id']
    text = row['snippet']
    document = process_document(text)
    processed[docid] = document
```

Iterating over documents in a dataframe

```
# set your working directory
import os
os.chdir('/home/elliott/Dropbox/text-project')
processed = {}
import pandas as pd
df = pd.read_csv('case-data.csv')
for i, row in df.iterrows():
    docid = row['cluster_id']
    text = row['snippet']
    document = process_document(text)
    processed[docid] = document
```

Iterating over documents in text files

```
from glob import glob
fnames = glob('docs/*txt')
for fname in fnames:
    docid = fname[:-4]
    text = open(fname).read()
    document = process_document(text)
    processed[docid] = document
```

Iterating over documents in text files

```
from glob import glob
fnames = glob('docs/*txt')
for fname in fnames:
    docid = fname[:-4]
    text = open(fname).read()
    document = process_document(text)
    processed[docid] = document
```

Iterating over documents in text files

```
from glob import glob
fnames = glob('docs/*txt')
for fname in fnames:
    docid = fname[:-4]
    text = open(fname).read()
    document = process_document(text)
    processed[docid] = document
```

Iterating over documents in text files

```
from glob import glob
fnames = glob('docs/*txt')
for fname in fnames:
    docid = fname[:-4]
    text = open(fname).read()
    document = process_document(text)
    processed[docid] = document
```

Iterating over documents in text files

```
from glob import glob
fnames = glob('docs/*txt')
for fname in fnames:
    docid = fname[:-4]
    text = open(fname).read()
    document = process_document(text)
    processed[docid] = document
```

Iterating over documents in text files

```
from glob import glob
fnames = glob('docs/*txt')
for fname in fnames:
    docid = fname[:-4]
    text = open(fname).read()
    document = process_document(text)
    processed[docid] = document
```

Saving data in Python

- pandas makes it easy to save files:

```
pd.to_pickle(processed,
'processed_corpus.pkl')
```

- If you have a dataframe `df`, you can save it as a python pickle, CVS, excel spreadsheet, or stata dataset:

```
df.to_pickle('dataset.pkl')
df.to_csv('dataset.csv')
df.to_excel('dataset.xlsx')
df.to_stata('dataset.dta')
```

- relational databases and HDF5 also available
- pandas can read all of these formats as well, of course.