EECS-317 Data Management and Information Processing

Lecture 14 – Web Scraping & Messy Data

Steve Tarzia

Spring 2019

Northwestern

Announcements

- Final project:
  - Rubric was posted.
  - Part 1 due tomorrow.
- HW5 due Friday.

Last Lecture: A Data Safari

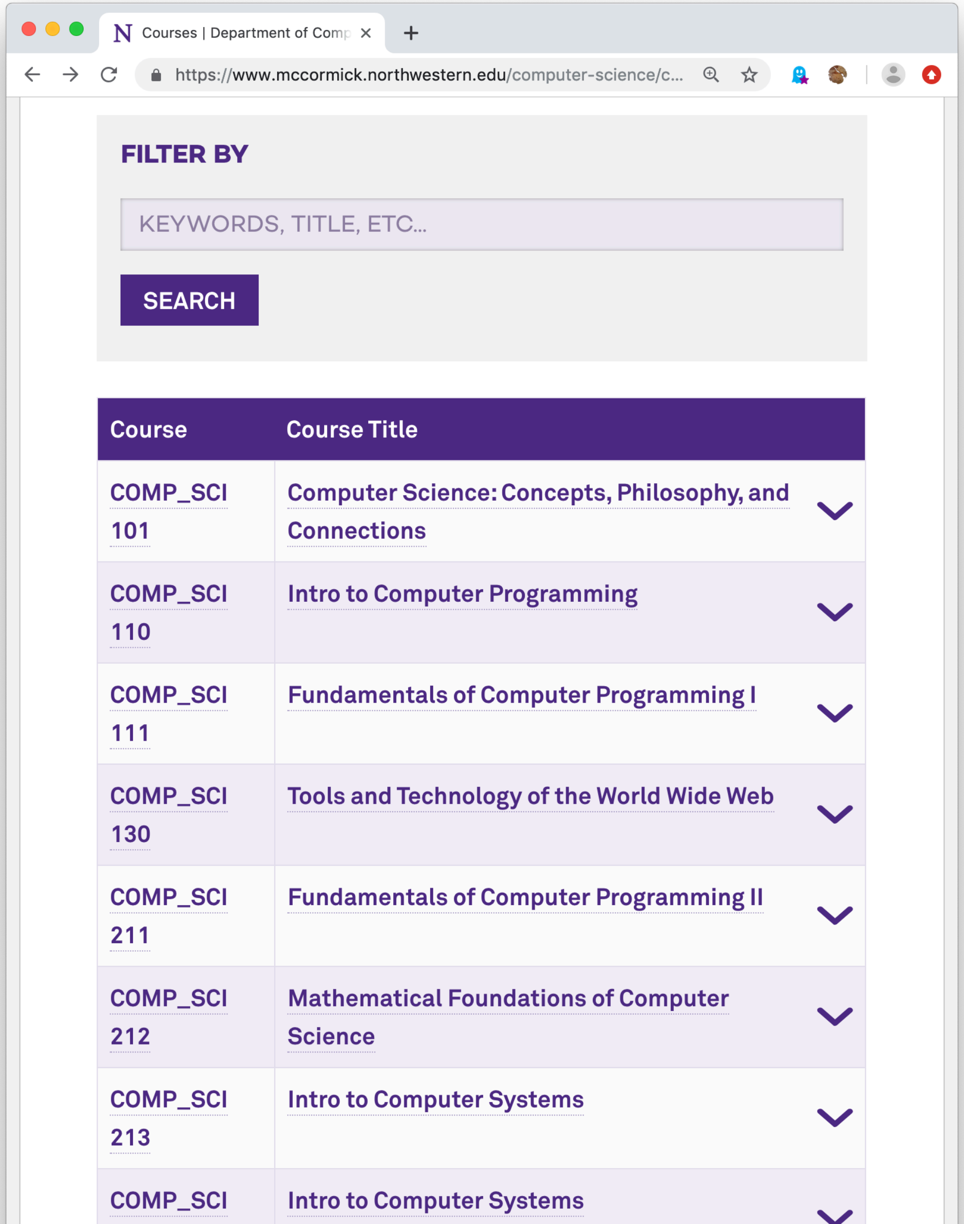
- CSV files are common
- Geographic data uses special file formats ("shape files")
- A data set might include many files (eg., Stanford dogs)
- Multiple tables can be distributed as a single SQLite database file
- REST APIs allow fetching of data by providing query information in the URL (or in a POSTed JSON object).
    - Return value is usually a JSON object.
    - The data provider must provide a specification for the API, to tell users how to construct requests and how to interpret responses.

Web Scraping/Crawling

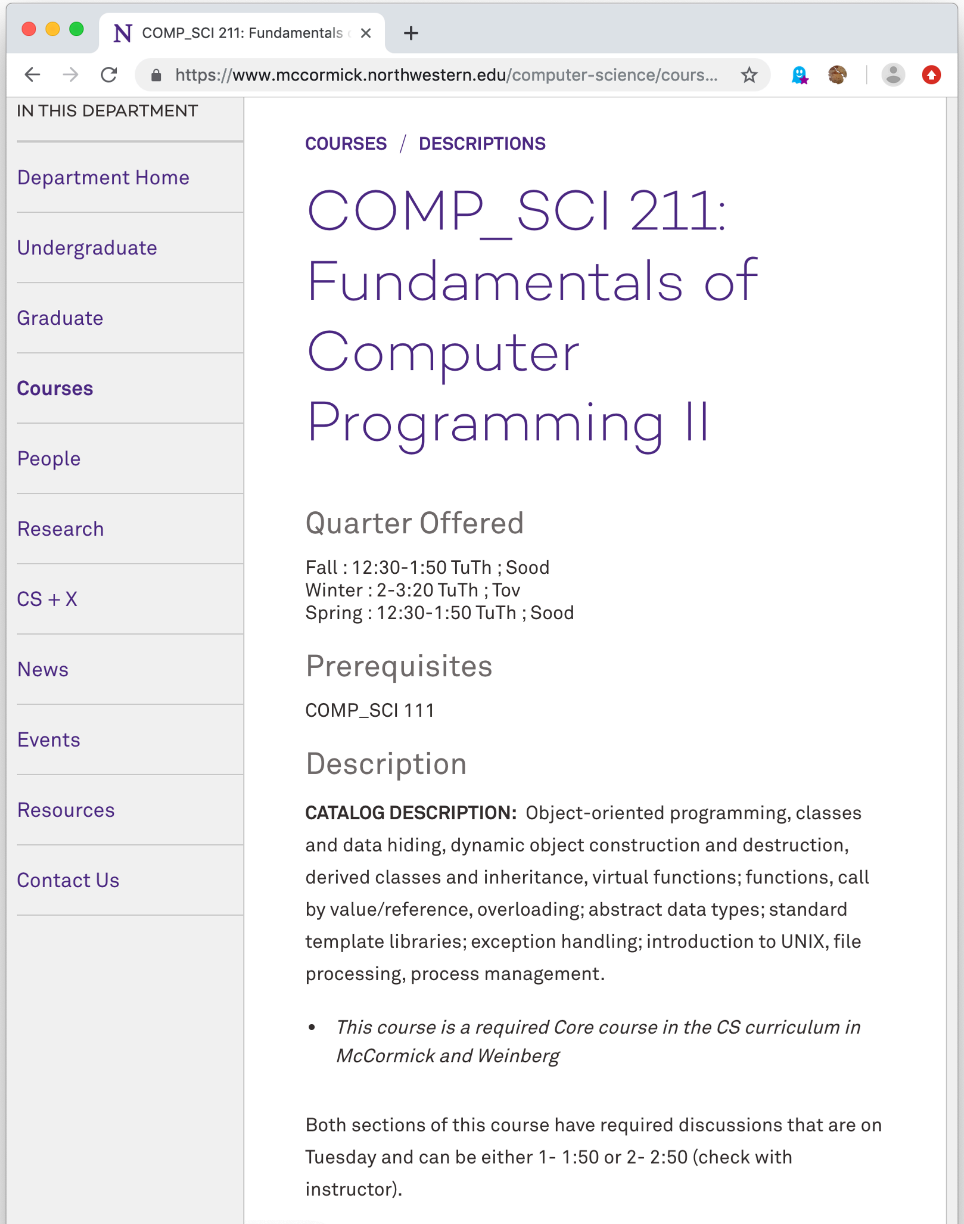
- Some data hosts have websites for humans to browse their data, but no clean way to programmatically access the data (no Data API).
- You could manually *click through* all the pages and copy the data, but this would be tedious.
- **Web scraping** is writing a computer program to "crawl" through a website and get all the data you need.
- **Warning**: don't violate a site's terms of service (more details)
  - For example, Facebook will cancel your account if they think you are scraping content from their site. LinkedIn **sued** ~100 individuals for scraping.
  - Don't steal data from a subscription service.
  - Computer Fraud and Abuse Act (CFAA) may apply even for "public" pages!

Let’s scrape CS course info:

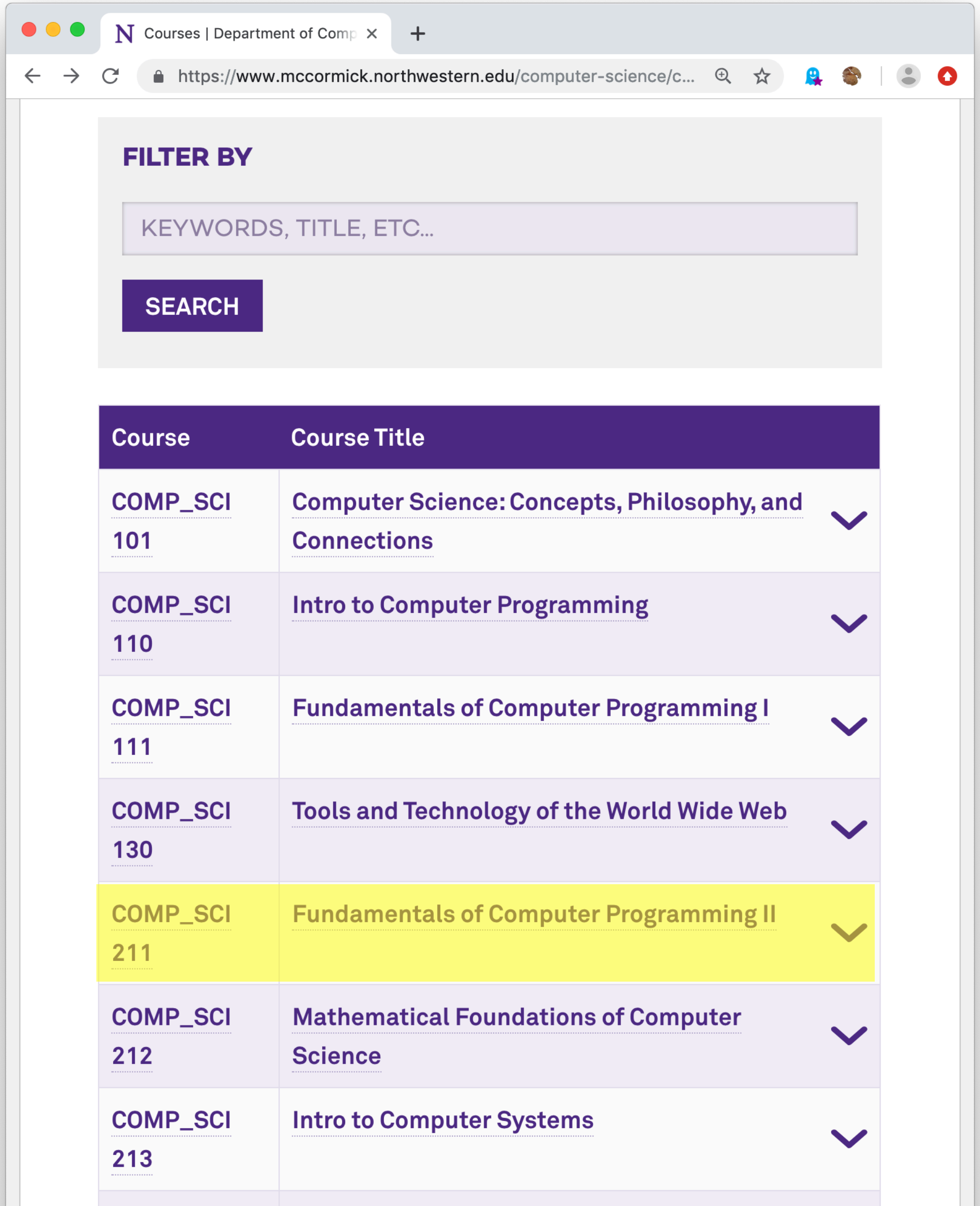
One index page



Many detail pages



Index page

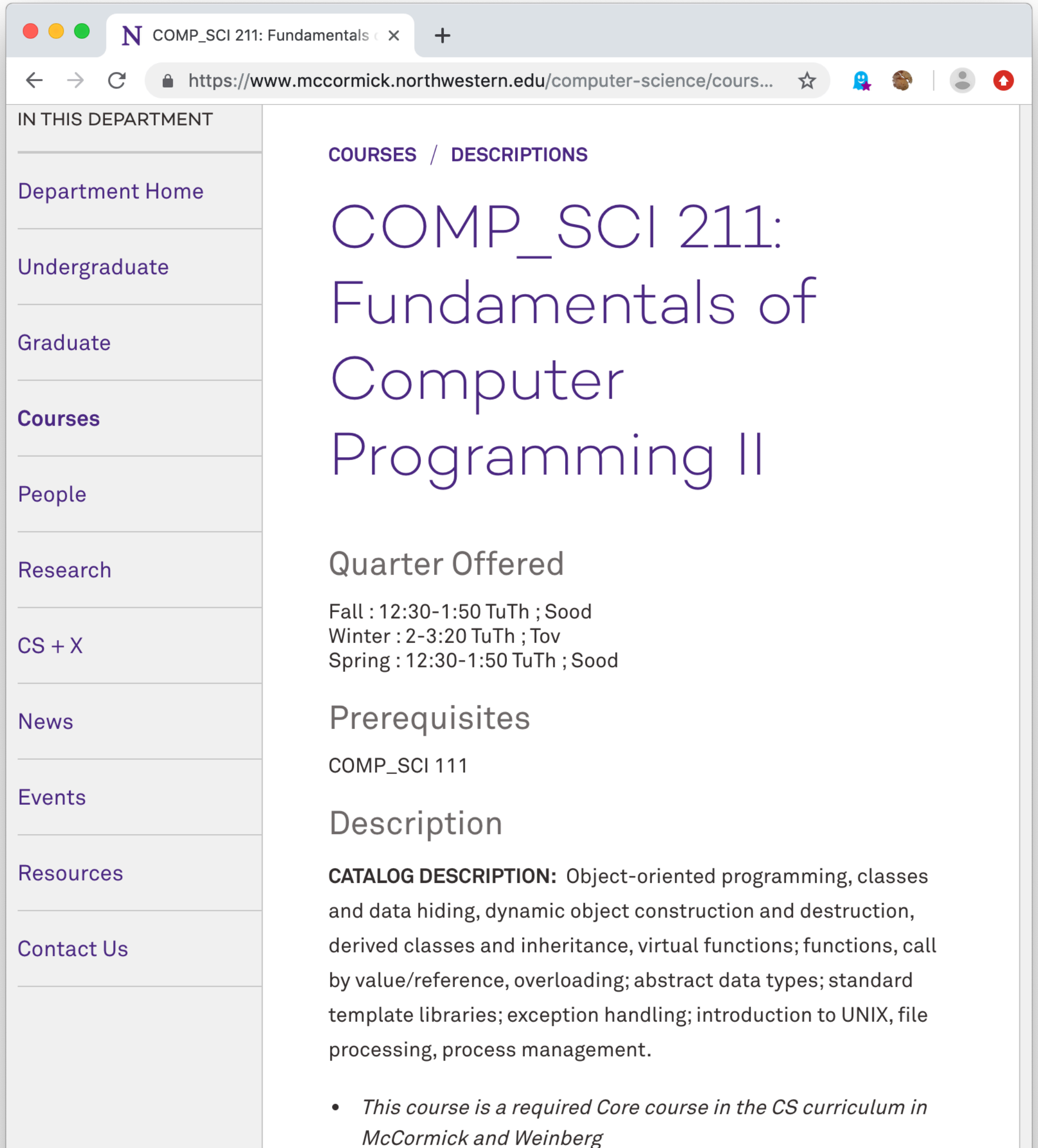


From the index page, we must scrape each course's:

- Course numbers
- Course title
- URL for course details

Then we'll scrape each course's detail page for further info.

Detail page



From the detail page, we wish to scrape:

- Quarter offered, time, instructor
- Prerequisites
- Description

Recommended scraping tools

- Python
- **requests** Python package to fetch web pages using HTTP
  - Recall that we also can use this library to get data from REST data APIs.
- **beautifulsoup4** Python package to parse HTML pages
  - Will be able to pick out elements of the page using *CSS selectors*.

- Code for the McCormick course scraping example is at:
  - https://github.com/starzia/webscraping-examples
  - 51 lines of Python code.

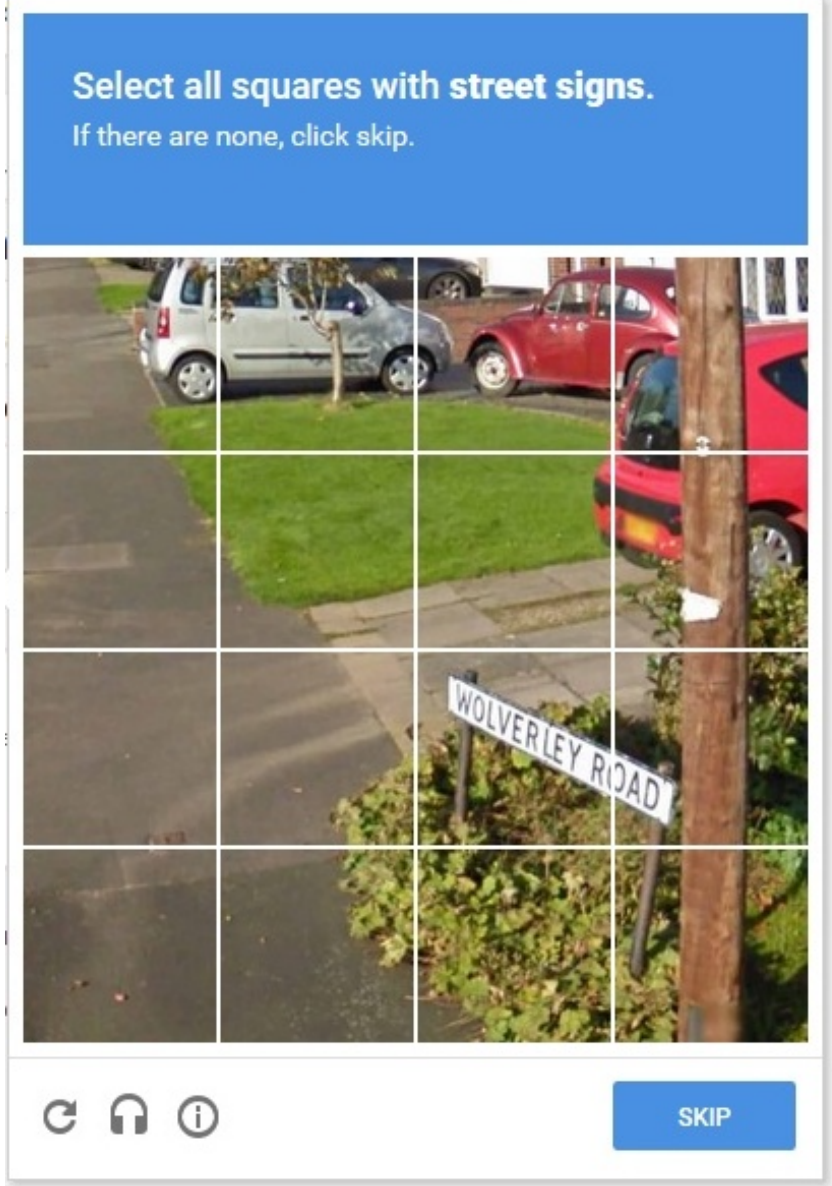
Web scraping example: USMS Swim team stats

- A slightly more complex example requiring 146 lines of Python code.
- https://github.com/starzia/usms-scrape
- Downloads a CSV swim team roster.
- Then scrapes swim meet results for each swimmer.
  - Uses **lxml** package and **XPath** syntax to pick out HTML nodes with relevant data.
- Reorganizes and prints the data as shown →

```
200 IM
   2:09.03 (39) Patrick Lahey
   2:14.23 (23) Daniel Melnick
   2:23.51 (41) David Corr
   2:25.49 (24) William Harris
   2:26.64 (64) Phil Dodson
   2:34.82 (29) Ruby Krueger
   2:41.78 (56) Bill Avery
   2:43.67 (41) Nichelle Pajeau
   2:46.75 (33) Stephen Tarzia
   2:46.94 (57) James Bychowski
   3:02.71 (66) Joe Carroll
   3:15.06 (42) Elizabeth Gjerde
   3:26.16 (61) Kathleen Roderer
   3:27.75 (60) Holly Seguine
   3:31.13 (58) Dana Deane
   3:50.97 (65) Robert Hertel
   4:36.68 (59) Sarah Fodor
```

A very complex web scraping example

- https://github.com/starzia/bibliometrics
- Gathers data for an analysis of the "research impact" of the top ten US business schools.
- Scrapes faculty directory pages for 10 universities, eg: K, H, S, S2
    - Also get lists of publications, like: K, H, S
- Also scrapes Google Scholar search results by using Selenium to control Firefox. Using a full browser (instead of *requests* lib) allows:
    - Scraping of pages that require Javascript
    - A human attendant can "babysit" the program and solve CAPTCHAs when prompted.



Web scraping overview

- Find the pages that hold the data
  - Often you'll start with a hard-coded index page and then programmatically look for links to additional pages.
  - Download the HTML (using Python **requests** package, for example)
- Extract the data from a given page:
  - Web pages are usually generated by a computer program, so the data will always be found within a certain pattern of HTML code.
- Locations in the HTML document can be specified in one of two ways:
  - **CSS selectors** – used be web page designers in Cascading Style Sheets to specify which fonts/colors/etc. *(styles)* apply to which parts of the page.
    - Python **beautifulsoup4** package uses CSS selectors
  - **XPath queries** – used for finding elements in an XML document (remember that HTML is a type of XML).
    - Python **lxml** package used XPath
  - CSS selector and XPath syntax can be tested in the Chrome developer tools.

CSS selectors pick out a set of HTML elements

- Tag type:
  - `'a'` matches `<a href="http://link.com">hello</a>`
- Class name:
  - `'.time'` or `'td.time'` matches `<td class="time">23</td>`
- Id name:
  - `'#best'` or `'td#best'` matches `<td id="best">103</td>`
- Attribute values :
  - `'a[href="http://link.com"]'`
    matches `<a href="http://link.com">hello</a>` but not `<a>this</a>`

Combining CSS Selectors

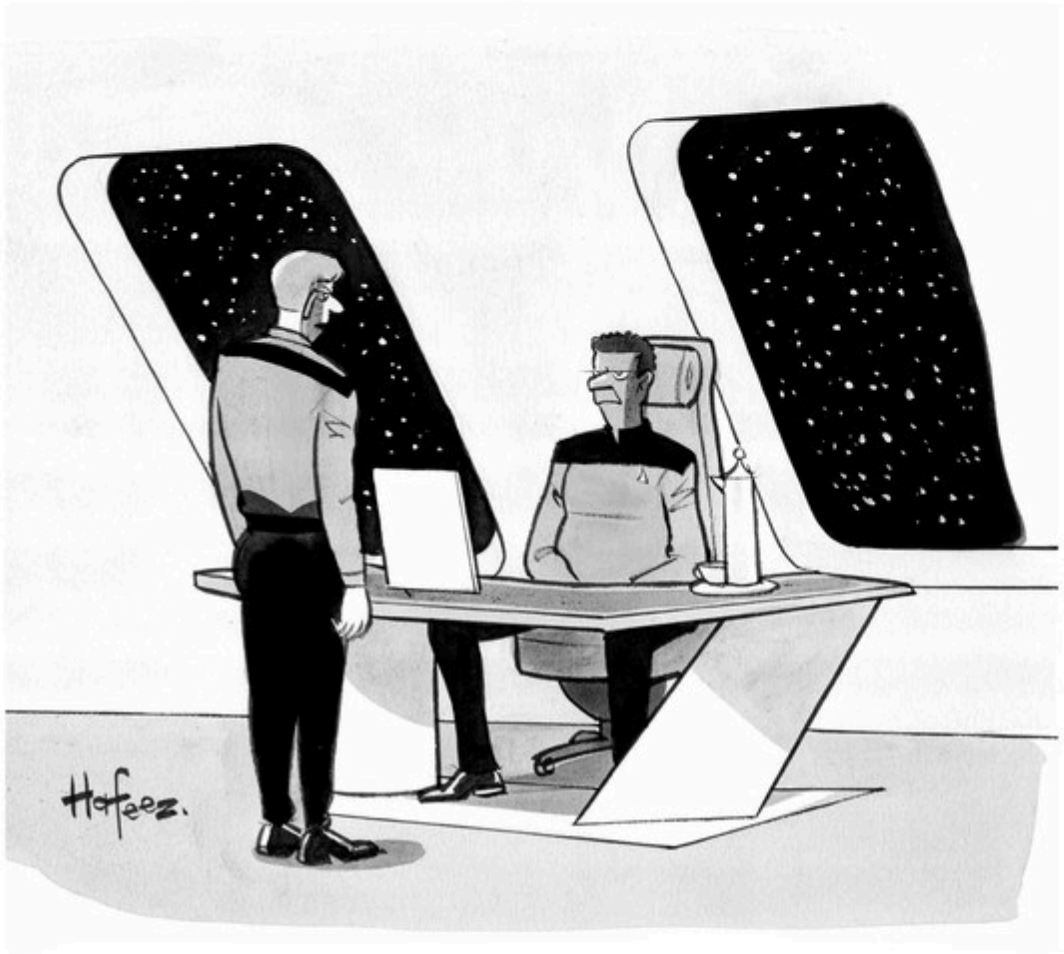
- Descendant: *[space]*
  - **`'table td'`** matches
    `<table><tr>`**`<td>a</td>`**`this`**`<td>b</td>`**`</tr><table>`
- Direct child: >
  - **`'tr > td'`** also matches above
- General siblings: ~
  - **`'td ~ td'`** also matches above
- Adjacent siblings: +
  - **`'p.heading + div.sec'`** matches
    `<div><p class="heading">hello</p>`
    `<div class="sec">target</div>``</div>`

References: https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Selectors https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors

Recap (part 1): Web Scraping

- Data can be **scraped** from web pages by writing code that:
  - Downloads HTML pages
  - Picks out data elements using **CSS selectors** (or XPath)
  - Also pick out links to pages with additional data
  - Repeat!

Intermission



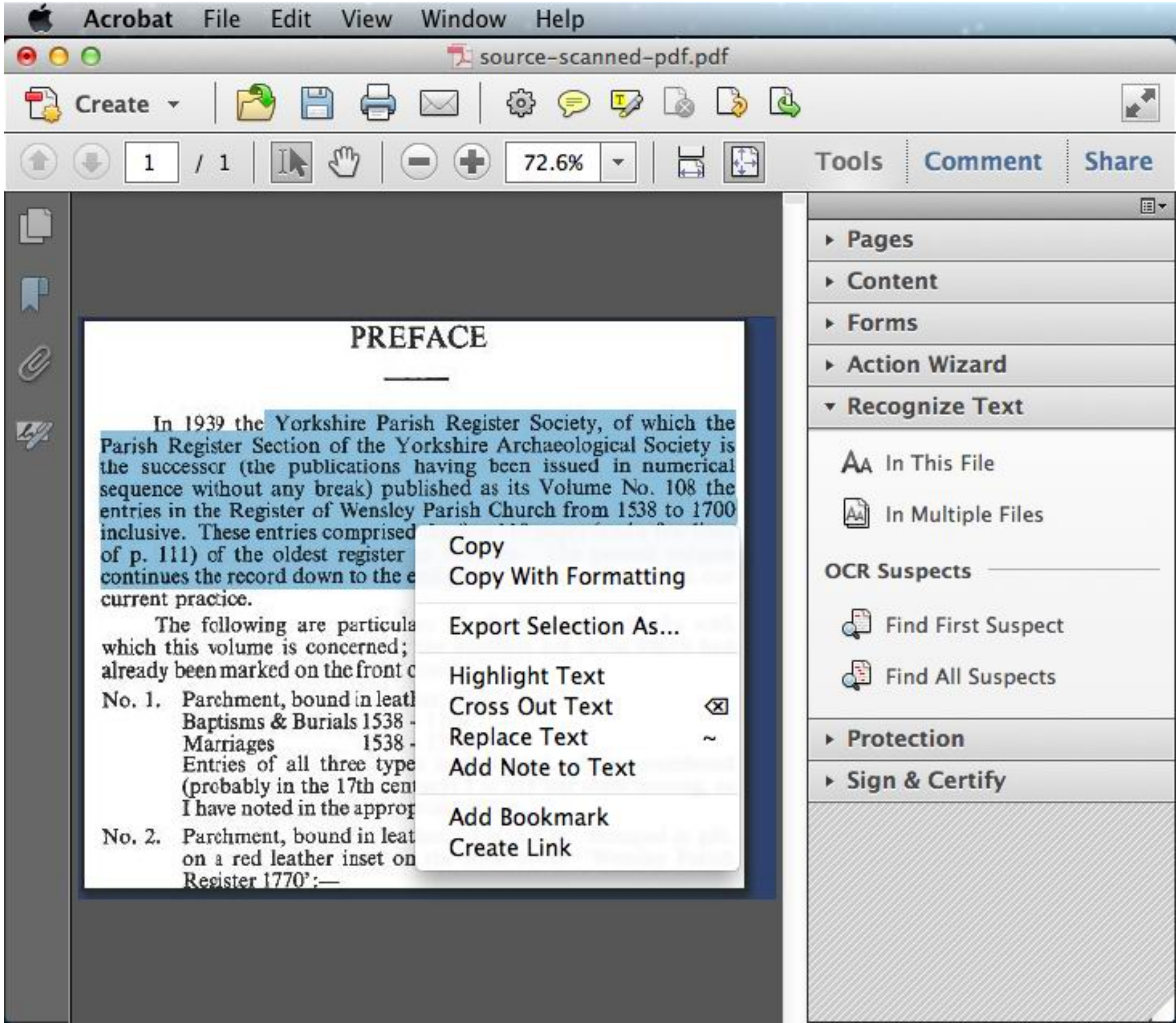
"Bad news, captain. The ship's computer has been sharing all our personal data with the Romulans."

Real world data is often **messy**

- Data entered manually can have typographic errors and use inconsistent naming conventions.
- Different data sources can us different naming conventions.
- Buggy computer programs can produce bad results.
  - Program may not behave well when data is missing.
  - Data overflow and type conversion can cause problems.
- Temporary sensor malfunction can lead to a bogus measurement.
- Numbers may have different units
  - dollars *vs* millions of dollars
  - fraction *vs* percent
- Data can be missing due to an interrupted data import.
- Data may be scanned from paper forms (leading to OCR errors).

Optical Character Recognition (OCR)

- OCR extracts text from scanned **images** of text.
- Adobe Acrobat Pro has OCR.
- Many scanners and all-in-one printers come with OCR software.
- [Python Tesseract]() is open-source, state-of-the-art OCR.
- Text is easy to capture accurately, but punctuation and formatting is difficult.
- Handwriting can also be recognized, with less accuracy.



Handwriting OCR

- Obviously, more difficult and error-prone than typed text.
- “Letter boxes” can help both human and OCR legibility.
- Checkboxes can be difficult to scan due to stray marks
  - A check mark or X mark may “spill over” into another box.
  - This may lead to multiple or zero selections instead of one.


1. Personal Details — Proposed Insured

Purpose of Insurance: SAVINGS & INSURANCE

Title Mr./Mrs./Ms./Dr. Name — First: SARIFUL; Middle: ; Last: ISLAM

Date of Birth: 23101983 — Sex: ☑ Male ☐ Female

Nationality: INDIAN

Country of Residence: INDIA

Age: 33 — Place of Birth: MURSHIDABAD

Age Proof: ☐ Birth Certificate ☐ Passport ☐ Per. Driving License ☐ School Certificate ☐ Service Record ☐ Others PAN CARD

Id Proof: ☑ Aadhar Card ☐ Driving License ☐ PAN card ☐ Passport ☐ Voter ID card ☐ Others

PERMANENT ADDRESS

Door No.: — Building Name: S/O: KAMALU

Plot No./Street Name: DD IN MONDAL

Landmark/: GUDHIA ROAD

Area: P/O:-GUDHIA

Place: MURSHIDABAD

City/District: MURSHIDABAD

State: WEST BENGAL

PIN: 742302

Address Proof: ☐ Passport ☐ Telephone Bill ☐ Electricity Bill ☐ Driving License ☐ Ration Card ☐ Current Bank Passbook ☐ Others AADHAR

Extract Transform Load (ETL)

- ETL programs move data between different storage media.
- For example:
  - Import data from data files into a database.
  - Move from one database to another.
- ETL script can help deal with messy data by including:
  - List of **validation rules** to identify problematic data
  - List of behaviors to **correct or discard** problematic data
  - **Transformations** to apply to the input data before inserting into DB
- Specialized ETL tools exist, like *MS SQL Server Integration Services*
- ETL also be done with general-purpose data processing tools:
  - Plain Python, Pandas, PySpark

How to recognize bad data?

No simple or easy answer.

- Start with good documentation. Know what each column means.
- Define a very strict schema and look for warnings when importing
  - Define columns as **`NOT NULL`**, when appropriate, to prevent incomplete data.
  - Define columns with numeric types rather than text if you expect numbers.
  - Define **foreign keys** if you expect columns to match between tables.
- Look at summary statistics after data is imported:

  `SELECT MIN(col), MAX(col), AVG(col)…`

  - If min and max values are unexpected, then look for outliers by sorting according to that column.
  - In R, use the **`summary(…)`** command on a data frame.
  - In Pandas (Python), use the **`describe()`** method on a data frame.

Debugging a data import

- If data fails to import completely, try loading it into a *temporary text table*
  - Drop keys and use large text types for every column
- Query the text table to look for unexpected values in the source data

This table has strict constraints on what kind of data can be inserted:

```
CREATE TABLE person (
  SSN int NOT NULL,
  firstName varchar(30) NOT NULL,
  lastName varchar(30) NOT NULL,
  birthDate char(10) NOT NULL,
  PRIMARY KEY (SSN)
);
```

This temporary table relaxes those constraints:

```
CREATE TABLE _import_person (
  SSN varchar(1000) NOT NULL,
  firstName varchar(1000) NOT NULL,
  lastName varchar(1000) NOT NULL,
  birthDate varchar(1000) NOT NULL,
);
```

Named Entity Matching

- In real-world data, people, companies, products, etc., all can be represented with variations of their name:
    - Eleanor Roosevelt
    - E. Roosevelt
    - Roosevelt, Eleanor
    - Mrs. Roosevelt
    - Northwestern Univ.
    - NWU
    - Northwestern
    - Northwestern University
    - Apple iPhone 6S
    - iPhone 6 S 32 GB Space Gray
    - A1633
- When combining data from multiple sources, we need **fuzzy matching** to join according to text fields.
    - Look for *approximate* text matches.
    - Humans are good at this, but it's difficult to automate.

SQL synonym table

- A simple solution is to create a ***synonym table*** to list all variations of names.
- Use the synonym table as a linking table in a four-way join.
- For example, if *product* and *product_details* use different variations of the product name:

```
SELECT * FROM product
 INNER JOIN product_synonym AS n1
   ON product.name=n1.name
 INNER JOIN product_synonym AS n2
   ON n1.id=n2.id
 INNER JOIN product_details
   ON n2.name=product_details.name;
```

product_synonym	
product_id	***name***
1	Apple iPhone 6s
1	iPhone 6 S
1	iPhone 6S 32 GB
1	iPhone 6S Space Gray
1	iPhone 6S Gold
2	Google Nexus 6P
2	Nexus 6P
2	Nexus 6-P

Shortcomings of synonym table

- Creating the synonym table manually is slow
  - Cannot be scaled to many thousands of rows
- Synonym table must be updated every time new data arrives.
- However, we may try to apply Machine Learning to automatically generate synonym tables for named entity matching…

Data cleaning tools

- You supply a CSV file, and the tool lets you quickly match synonyms
- http://dedupe.io https://youtu.be/9wEA90Fz-lU?t=109
  - Uses machine learning.
- http://openrefine.org/ https://youtu.be/B70J_H_zAWM
  - Lets you quickly define matching rules.
- Or, develop your own tools (described next)

Text similarity metrics

- An alternative to ML is a graph partitioning approach
- Use text similarity metrics to build a name similarity graph.
- For example, the **edit distance** (or Levenshtein distance) is the minimum number of single-character changes needed to make one phrase equal to another.
  - Edit distance between "school" and "college" is 7 because you have to delete an *s*, *h*, *o*, and add "*lege*"
  - Edit distance between "iPhone 6S" and "iPhone 6-S" is just one (delete the hyphen)
  - Edit distance between "iPhone 5" and "iPhone 6" is also just one, but these are different phone models.
  - Edit distance is useful, but cannot be used blindly.

Amazon Mechanical Turk

- A ***crowdsourcing marketplace***.
- Allows you to pay a few cents for a human to answer a short question.
- Useful for small, repetitive problems requiring **human intelligence**, where simple rules or even Machine Learning would not work.
- Example: pick out the year of graduation from professors' CVs:
  - Each CV is a PDF document (an academic resumé).
  - These documents all have different formats.
  - It's difficult for a computer to reliably parse them, but easy for a person.

JILLIAN CHOWN

Kellogg School of Management, Northwestern University
2211 Campus Drive, Rm 5165, Evanston, IL 60208
jillian.chown@kellogg.northwestern.edu
+1 (224) 714-8277

ACADEMIC APPOINTMENTS

Kellogg School of Management, Northwestern University Evanston, IL

Assistant Professor of Management and Organizations (2016-present)

Donald P. Jacobs Scholar (2016/17)

EDUCATION

Rotman School of Management, University of Toronto Toronto, ON

PhD, Strategic Management (2016)

- Dissertation: Professions, Organizations and the Challenges of Change: A Multi-Method Exploration in the Context of Healthcare Delivery
- Committee: Sarah Kaplan (chair), Anita McGahan, Brian Golden, Nicola Lacetera

Masters of Business Administration (2006)

- Top academic standing for Full Time MBA Core

University of Toronto Toronto, ON

Bachelor of Applied Science - Engineering Science (2004)

CURRICULUM VITAE **March 2018**

Torben Gustav Andersen

Department of Finance
Kellogg School of Management
Northwestern University
2211 Campus Drive
Evanston, Illinois 60208, USA
Tel: (847) 467-1285
E-Mail: t-andersen@kellogg.northwestern.edu

Personal Information

Marital status: Married; three children.
Citizenship: Denmark; U.S. Permanent Resident.

Education

Ph. D. in Economics, **Yale University,** New Haven, 1992.
Master's Degree, Science, Economics and Mathematics (Cand Scient Oecon).
University of Aarhus, Denmark, 1985.

Academic Positions:

2015-2017 Chair, Department of Finance, Kellogg School of Management, Northwestern University.

2000-present: *Nathan S. and Mary P. Sharp Distinguished Professor of Finance*, Department of Finance, Kellogg School of Management, Northwestern University.

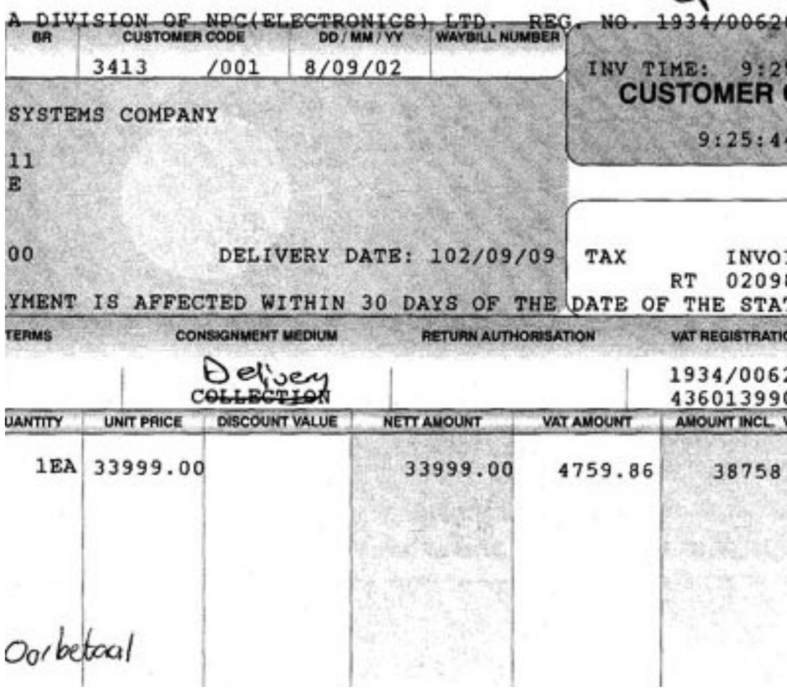
2011-2017 Member of Board, *Foundation for Advancement of Research in Financial Economics*

2013-present: Fellow of the ***Society for Financial Econometrics, SoFiE***.

2008-present: Fellow of the ***Econometric Society***.

2008-present: Research Affiliate of ***The Volatility Institute,*** Stern School of Business, New York University.

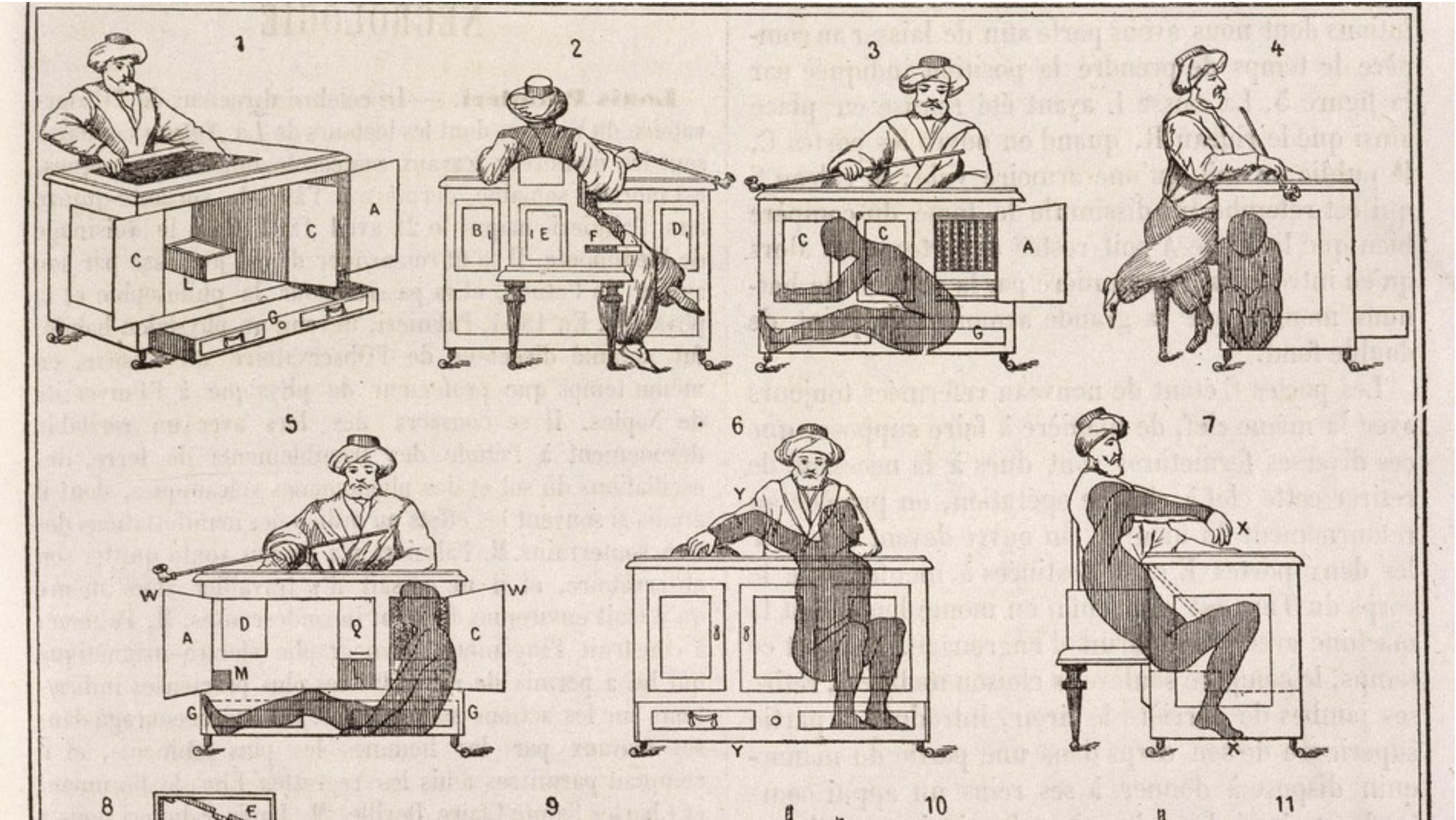
MTurk use cases


A DIVISION OF NPC(ELECTRONICS) LTD. REG. NO. 1934/0062
BR CUSTOMER CODE DD / MM / YY WAYBILL NUMBER
3413 /001 8/09/02
INV TIME: 9:2
CUSTOMER
SYSTEMS COMPANY
9:25:4
11
E
00 DELIVERY DATE: 102/09/09 TAX INVO
RT 0209
YMENT IS AFFECTED WITHIN 30 DAYS OF THE DATE OF THE STA
TERMS CONSIGNMENT MEDIUM RETURN AUTHORISATION VAT REGISTRATIO
Delivery
COLLECTION
1934/0062
43601399
UANTITY UNIT PRICE DISCOUNT VALUE NETT AMOUNT VAT AMOUNT AMOUNT INCL.
1EA 33999.00 33999.00 4759.86 38758

- Parsing data in unstructured forms
- Poorly-scanned documents
- Transcribing audio
- Photo identification.
- Generating *training data* for Machine Learning.

- MTurk tasks can be created by non-programmers,
  but there is also an advanced API to setup complex tasks.

Named after a fake chess-playing **automaton** (1770)



When to trust human input

Humans are unrealiable, so how can we make MTurk results more trustworthy?

- Use only experienced and highly-rated MTurk workers.
- Use majority voting:
  - Give the same task to three different workers
  - If at least two of the three give the same answer, then trust it.
- Manually or programmatically check the results, if possible
  - Sometimes it's easier to check the answer than to generate it.

Crowdsourced data gathering & processing

- *Crowdsourcing* is using the power of online crowds to do some work.
- MTurk is kind of an example, but usually "crowdsourcing" refers to unpaid work.

Examples:

- ProPublica: Free the Files (2012) https://youtu.be/tTlA_TJHq5o?t=198
- iNaturalist
- National Gun Violence Memorial https://youtu.be/UWzWwT546EY

Additional resources on Data Cleaning

- https://github.com/Quartz/bad-data-guide
- https://www.coursera.org/learn/data-cleaning

Recap (part 2): Messy Data

- Data can have missing, incorrect, or inconsistent values for many reasons:
  - Pulled from different sources with different naming or unit conventions
  - Paper scanning (OCR) errors
  - Human input errors
- Variety of tools are needed to deal with messy data:
  - Review summary statistics
  - Synonym tables
  - Named entity matching with ML (dedupe.io and Open Refine)
  - Crowdsourcing: MTurk, home-grown solutions
- Above all, don't blindly trust data you are given!