API-231 / GIS-PubPol
Meeting 10 (Geocoding)

Yuri M. Zhukov
Visiting Associate Professor of Public Policy
Harvard Kennedy School

February 27, 2024

What is **geocoding**?

- assignment of geographic code to descriptive locational data

Example:

- input: "Ann Arbor"
- output: (42.281, -83.748)

Figure 1: Find your location!

Geocoder components

- input query (e.g. address)

  ↓
- pre-processing algorithm (tokenization, standardization)

  ↓
- matching algorithm (exact vs. fuzzy, tie-break rule)

  ↓
- reference data (e.g. gazetteer)

  ↓
- output feature (e.g. point, code)

Figure 2: Input address, output data

Input

Input queries

What can be geocoded?

Descriptive locational data:

1. Postal addresses
   ("1201 South Main Street, Ann Arbor, MI 48104-3722")
2. Street intersections
   ("South Main and Stadium, Ann Arbor, MI 48104-3722")
3. Partial addresses
   ("S. Main St., Ann Arbor, MI")
4. Postal codes ("48104-3722")
5. Named buildings, landmarks
   ("Michigan Stadium")
6. General place names ("Ann Arbor")
7. Free-form queries ("The Big House")

Figure 3: Hail to the victors

Sources of error in input data

1. Imprecise queries → imprecise output
   (street address vs. county name)
2. Ambiguous queries → multiple matches
   (Springfield, Portland, Alexandria)
3. Too much precision → fewer matches
   (regimental command post at Hill 55)
4. Alt. spellings, typos → false matches
   (Granada, Spain vs. state of Grenada)
5. Place name changes → non-matches
   (Aleksandrovka/Yuzovka/Stalino/Donets'k)
6. Slang, nicknames → non-matches
   ("Paris of the Midwest", "Motown")

How to avoid some of these problems?

- pre-process the text of the input query

Figure 4: Wrong number

Pre-processing algorithm

What is **pre-processing**?

- standardization and normalization of input into a format and syntax compatible with the reference dataset

Why pre-process?

- prevent avoidable geocoding errors
- becomes more important where text is more unstructured, ambiguous
    - easy: "Ann Arbor, MI"
    - hard: "the Michigan city of Ann Arbor"
    - harder: "I met my friend Dallas when we were both college students, living in A2"

Figure 5: Undeliverable address

Common pre-processing tasks

1. Remove HTML tags, control characters
2. Remove non-alphanumeric characters
3. Remove capitalization
4. Remove punctuation
5. Parts-of-speech tagging
6. Lemmatization

Figure 6: Lost in translation

Filtering unnecessary words, text

Why strip capitalization, punctuation, etc?

1. Reconcile address formats
   (Cambridge, MA $\neq$ Cambridge MA)
2. Raise probability of match
   (Middlesex county $\rightarrow$ middlesex county)
   (Middlesex County $\rightarrow$ middlesex county)
3. Avoid computational errors
   ('#', '%' are special characters in many programming languages)

MLB Cincinnati Reds T Shirt Size XL
['mlb', 'cincinnati', 'red', 'shirt', 'size']

Razer BlackWidow Chroma Keyboard
['razer', 'blackwidow', 'chroma', 'keyboard']

AVA-VIV Blouse
['ava', 'viv', 'blous']

Leather Horse Statues
['leather', 'hors', 'statu']

24K GOLD plated rose
['gold', 'plate', 'rose']

Figure 7: Sentences $\rightarrow$ Tokens

Parts of speech tagging

Do we care if a word is a noun or a verb?

It depends on the application:

- well-formatted addresses:
  POS unimportant ("Ann Arbor, MI")
- unstructured queries:
  POS more important ("I met my friend Dallas when we were students in A2")
- various POS tagging software available online (`nlp.stanford.edu`)
- some APIs do this automatically

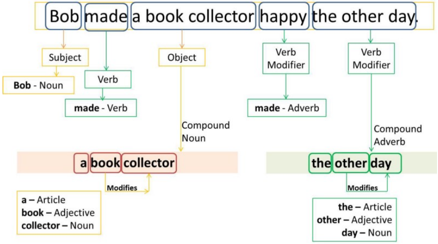

Figure 8: Sentence → POS tags

Lemmatization

relating multiple versions of same word to common, standard term

1. Many-to-one mappings
   - (Ann Arbor, A2, A-squared, the Deuce, Tree Town) → Ann Arbor
   - useful to associate nicknames, historical names with single location
2. One-to-many mappings
   - Dallas → Dallas (TX)
   - Dallas → Dallas (my friend)
   - Jackson → Jackson (MS)
   - Jackson → (Janet) Jackson
   - useful to distinguish places from people
   - requires info about word order, context

Procedure:

- create lookup table for relevant terms
- query table for each occurrence of word
- trade-off: speed vs. accuracy

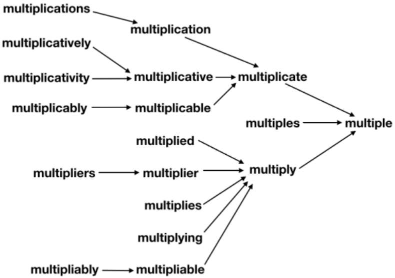

Figure 9: Many-to-one example

Output

Matching algorithm

How to find the best output candidate?

1. Exact vs. fuzzy matching
   - exact: Ann Arbor $\neq$ ann arbor
   - fuzzy: Ann Arbor $\sim$ ann arbor
2. Non-match rule (if zero matches)
   - return N/A?
   - geocode at lower resolution?
   - query additional datasets?
3. Tie-breaking rule (if multiple matches)
   - first match?
   - random match?
   - most precise match?
   - most popular match?

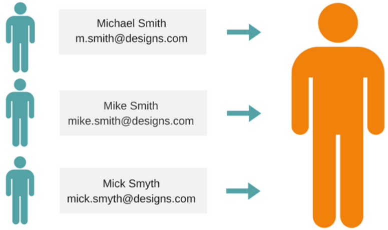

Figure 10: Match-making

Sources of error in matching

1. False positive matches:
   "my friend Dallas" → Dallas, TX
2. False negative matches: "A2" → N/A
3. Multiple matches:
   "Memphis" → Memphis, TN; Memphis, Egypt

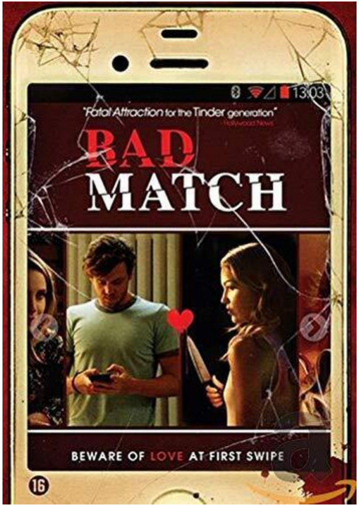

Figure 11: Bad film (probably)

Reference data

What are **reference data**?

Geographically-coded information used to match input to output

1. Gazetteers
2. TIGER/Lines
3. Crowd-sourced

Figure 12: Like this, but electronic

Gazetteer data

- dictionary of standard and alternate spellings of place names, and their geographic locations
  (e.g. NGA GEOnet Names)

Name	Postcode Sector	County	X	Y	Longitude	Latitude
River Ray	SN5 5	Swindon	412577	186443	-1.8199169	51.5766647
Galleygrove	GU31 5	West Sussex	480460	124901	-0.8542991	51.0178031
Sparcells	SN5 5	Swindon	412055	186724	-1.8274401	51.5792026
Monkton Down	SN8 1	Wiltshire	412049	172007	-1.8280256	51.4468736
Marden Copse	SN10 3	Wiltshire	408158	155164	-1.8843980	51.2954929
CAMBERLEY	GU15 4	Surrey	487155	161017	-0.7501234	51.3415119
Eastheath	RG41 2	Wokingham	480781	167319	-0.8401831	51.3991075
Downend	PO16 8	Hampshire	459894	106152	-1.1505390	50.8517277
Clatford	SN8 4	Wiltshire	416226	168557	-1.7680773	51.4157487
Walkers Hill	SN8 4	Wiltshire	411270	163426	-1.8395055	51.3697312
Ratfyn	SP4 7	Wiltshire	416135	142403	-1.7705612	51.1805774
Home Farm	GU8 6	Surrey	494007	145703	-0.6558010	51.2027557
Gunters	GU28 9	West Sussex	491654	125939	-0.6944889	51.0254613
Brewerslees	RG7 4	West Berkshire	462113	166619	-1.1086169	51.3951622
North Hayling	PO11 0	Hampshire	473097	102906	-0.9636544	50.8210266
Hyde	SO23 7	Hampshire	448392	130235	-1.3107250	51.0693529
Stoke Row	RG9 5	Oxfordshire	467805	184064	-1.0234712	51.5513487
Woodbarn	PO18 9	West Sussex	479007	111990	-0.8778097	50.9019247
Halfway	RG20 8	West Berkshire	440966	168474	-1.4123406	51.4137587
Hook	SO31 9	Hampshire	450740	105305	-1.2806854	50.8449856
Elston	SP3 4	Wiltshire	406361	144871	-1.9103513	51.2029606
Wickham Heath	RG20 8	West Berkshire	442046	169861	-1.3966476	51.4261507
Forton	PO12 3	Hampshire	460700	100068	-1.1400959	50.7969385
Six Acres	SN9 6	Wiltshire	410822	155296	-1.8461861	51.2966359

Figure 13: Example gazetteer data

Topologically Integrated Geographic Encoding and Referencing (TIGER/Line)

- U.S. Census Bureau's digital database for finding locations along roads

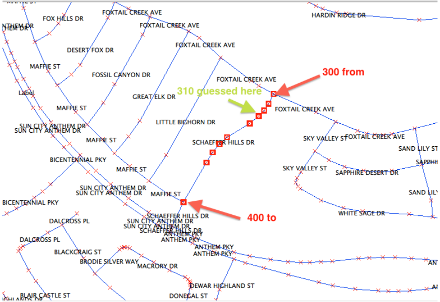

Figure 14: Example TIGER/Line

Crowd-sourced data

- user-generated location data from surveys, GPS devices, free sources
  (e.g. OpenStreetMap Nominatum)

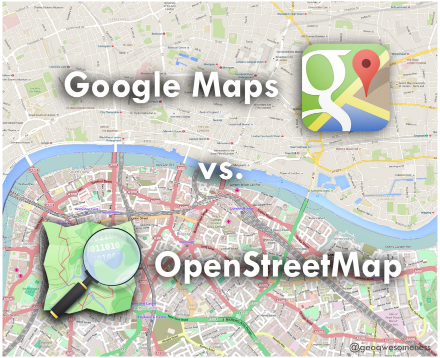

Figure 15: OSM is free, Google isn't

Sources of error in reference data

- data quality can be region-specific
  (e.g. Google vs. Yandex)
- less precise, sparser data in rural areas and developing countries
- some datasets not frequently updated
- different datasets use different standard name spellings

Figure 16: Re-routing

What is the **output**?

Any geographically-referenced information:

1. Point coordinates
   (longitude, latitude)
2. Line features
   (TIGER line segment)
3. Polygon features
   (parcel of land, census block, census tract, municipality, district, region, country)

Figure 17: Location found!

Sources of error in output

1. Point locations for areal references
   - geographic centroid?
   - capital city?
   - population-weighted centroid?
2. Linear interpolation on TIGER/Line shapefiles

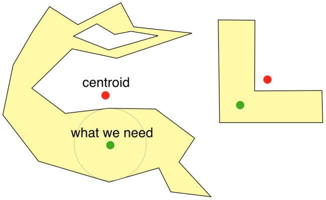

Figure 18: Wrong centroid

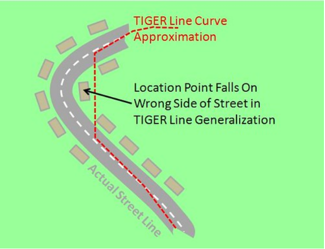

Figure 19: Wrong line