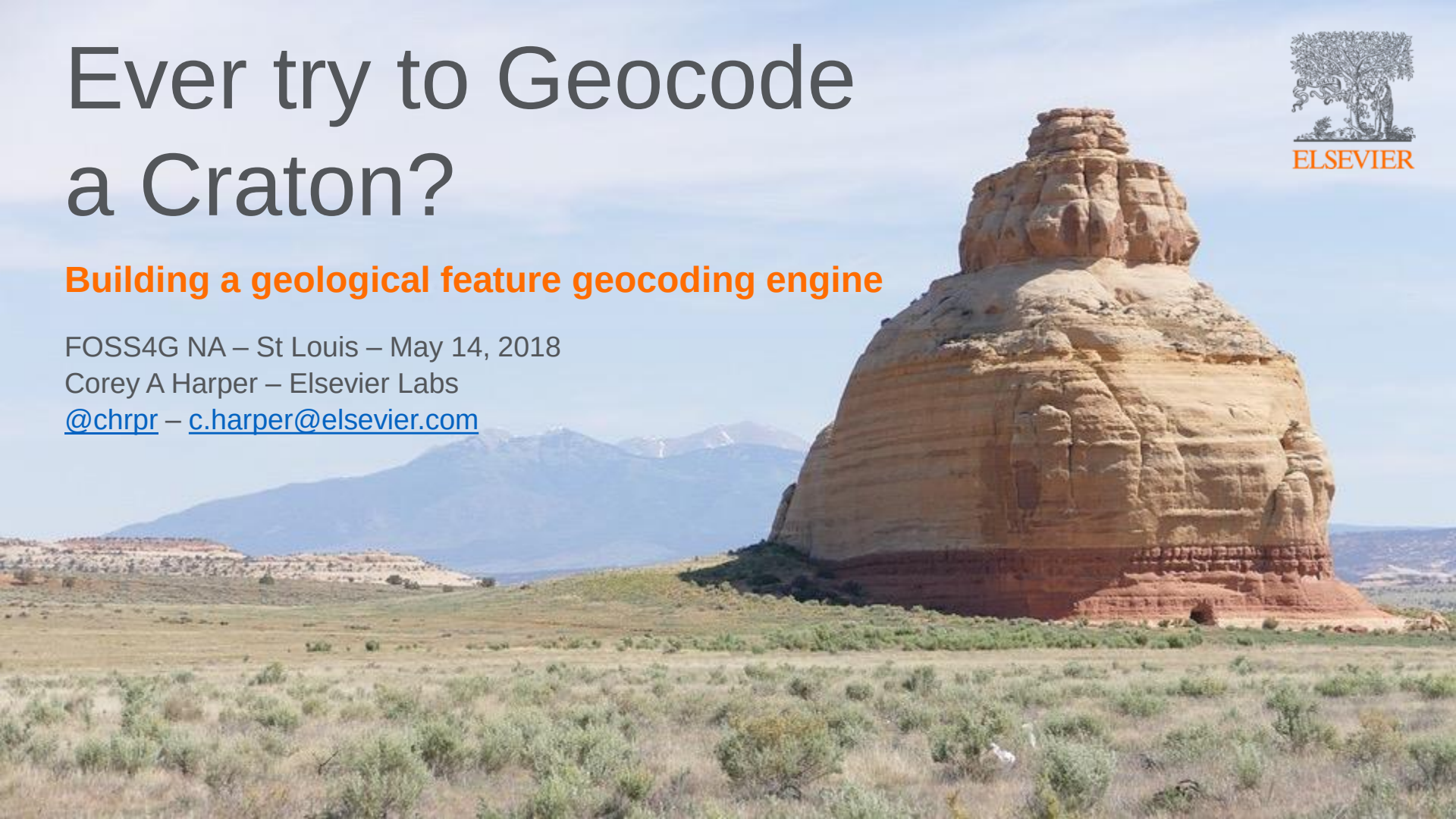
Ever try to Geocode a Craton?

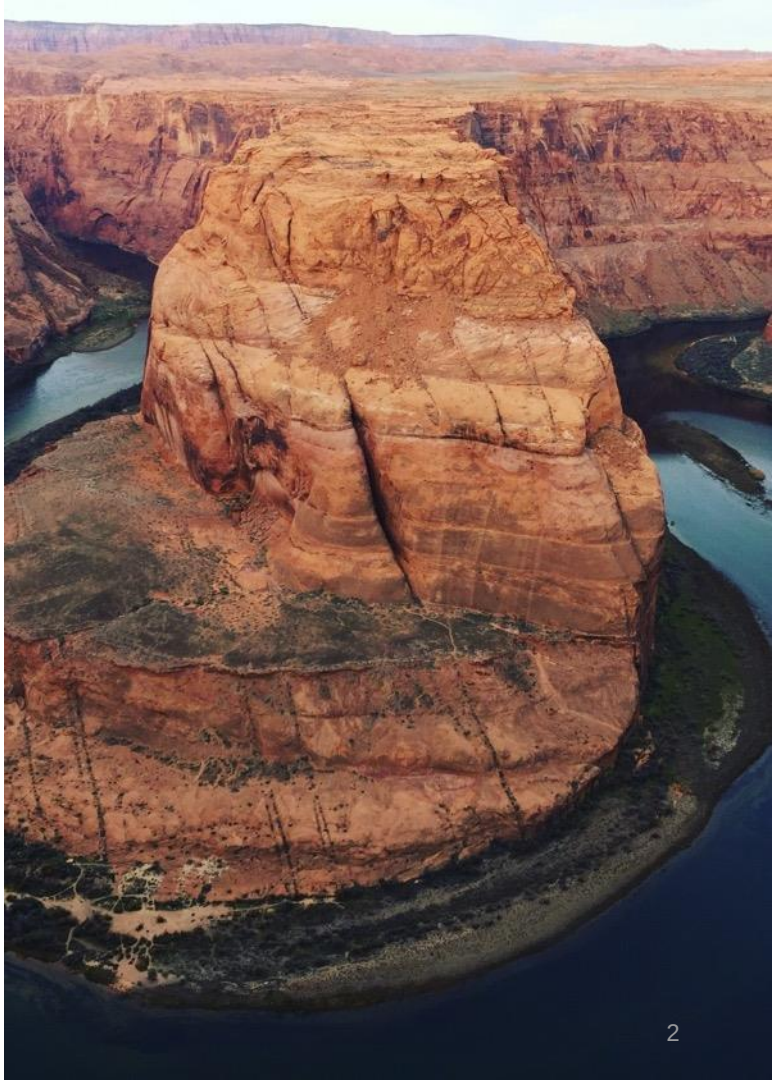
Building a geological feature geocoding engine

FOSS4G NA – St Louis – May 14, 2018
Corey A Harper – Elsevier Labs
@chrpr – c.harper@elsevier.com



Agenda

- Geofacets Background
- Problem Statement
- Previous work
- Data and proposed solution
- Tooling and code
- Results
- Future work



Geofacets™

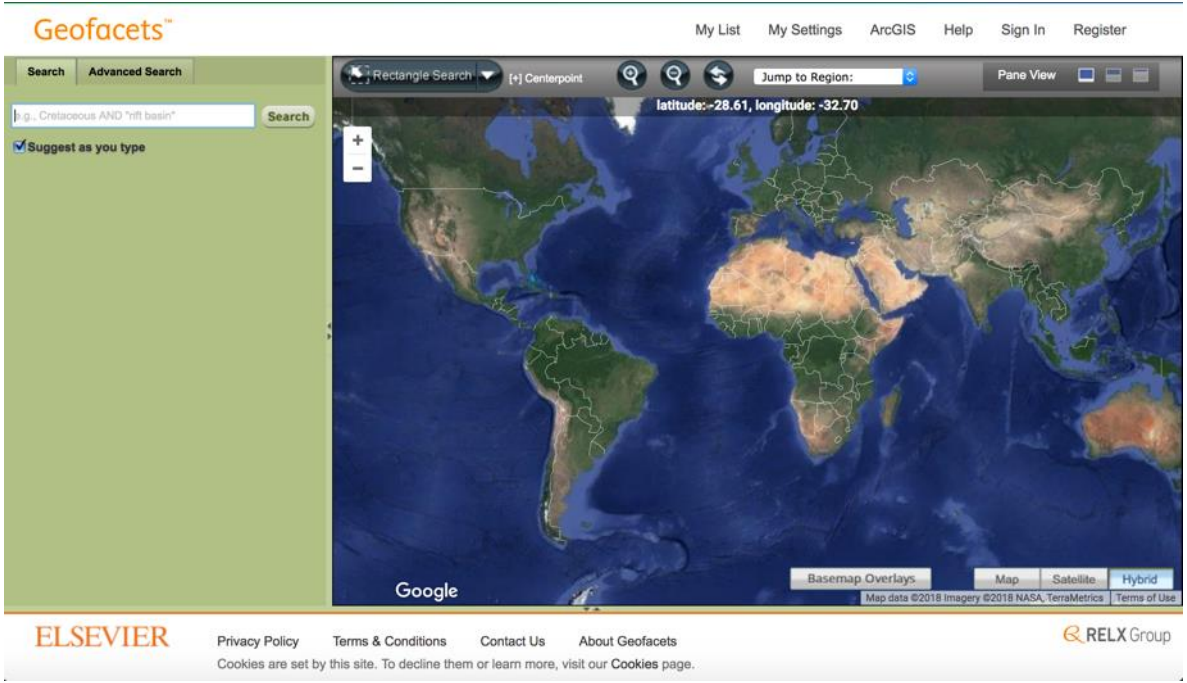
My List | My Settings | ArcGIS | Help | Sign In | Register

Search | Advanced Search

e.g., Cretaceous AND "rift basin" Search

Suggest as you type

Geofacets Demo



How does this work?

- Currently manual processing
- Map inset geolocation at “Area of Interest” level
- Tables and figures geolocated with maps from same article

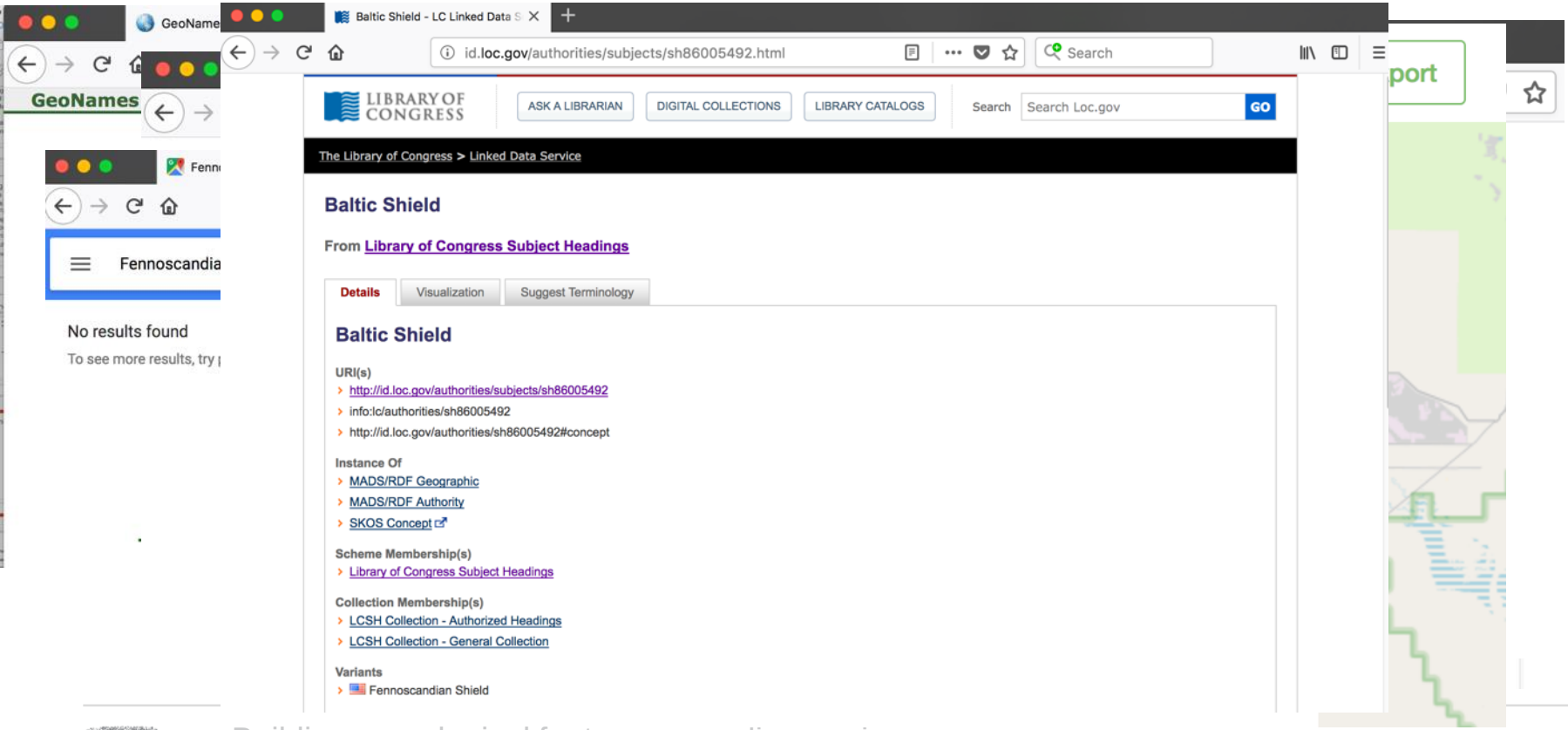
- Hard to scale this to public data sources
- Can’t geo-reference non-cartographic resources

Geocoding (vs Georeferencing)

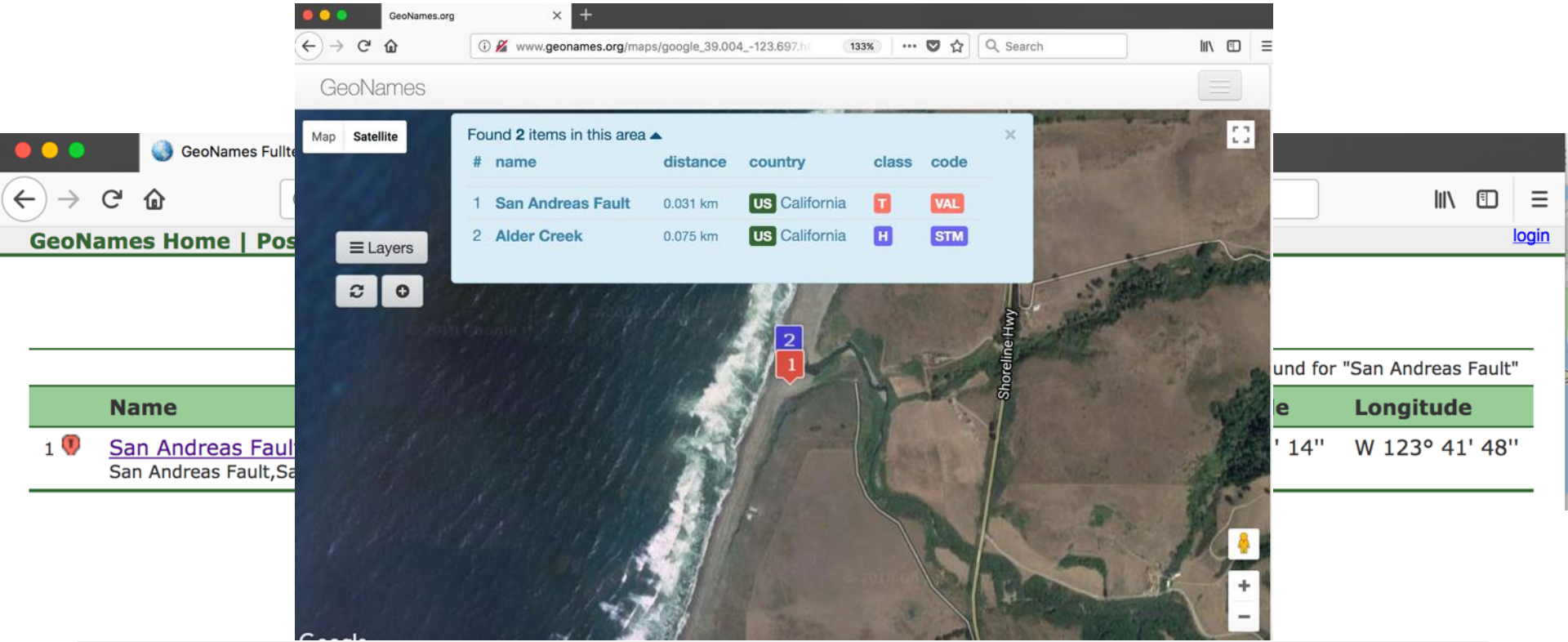
- Geonames
- Two Fishes (no more demo?)
- Who's on First "Spelu
- Google Maps API
- Yahoo Maps API
- Other sources of Geo Data
  - Library of Congress Subject Headings
  - Getty Thesaurus of Geographic Names

Cratons and Shields

Fennoscandian Shield



San Andreas Fault



Details

< Previous | Result 3 of 4864 | Next >

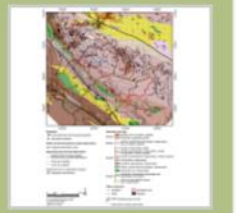
Brittle tectonothermal evolution in the Forsmark area, central Fennoscandian Shield, recorded by paragenesis, orientation and $^{40}Ar/^{39}Ar$ geochronology of fracture minerals

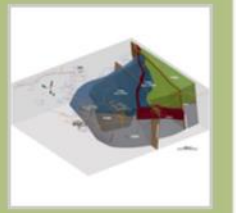
Tectonophysics, Volume 478, 3-4, December 2009, pages 158-174

Sandström, Björn; Tullborg, Eva-Lena; Larson, Sven Åke; Page, Laurence

Results | Article Info + Abstract | Permissions

Results within Article | Map | Metadata | Image Citations | Original Image | 1 of 9 in article Prev Next





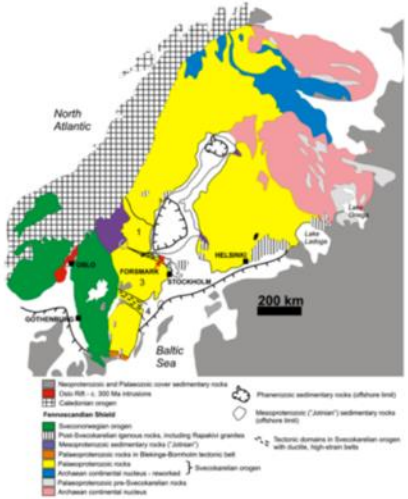
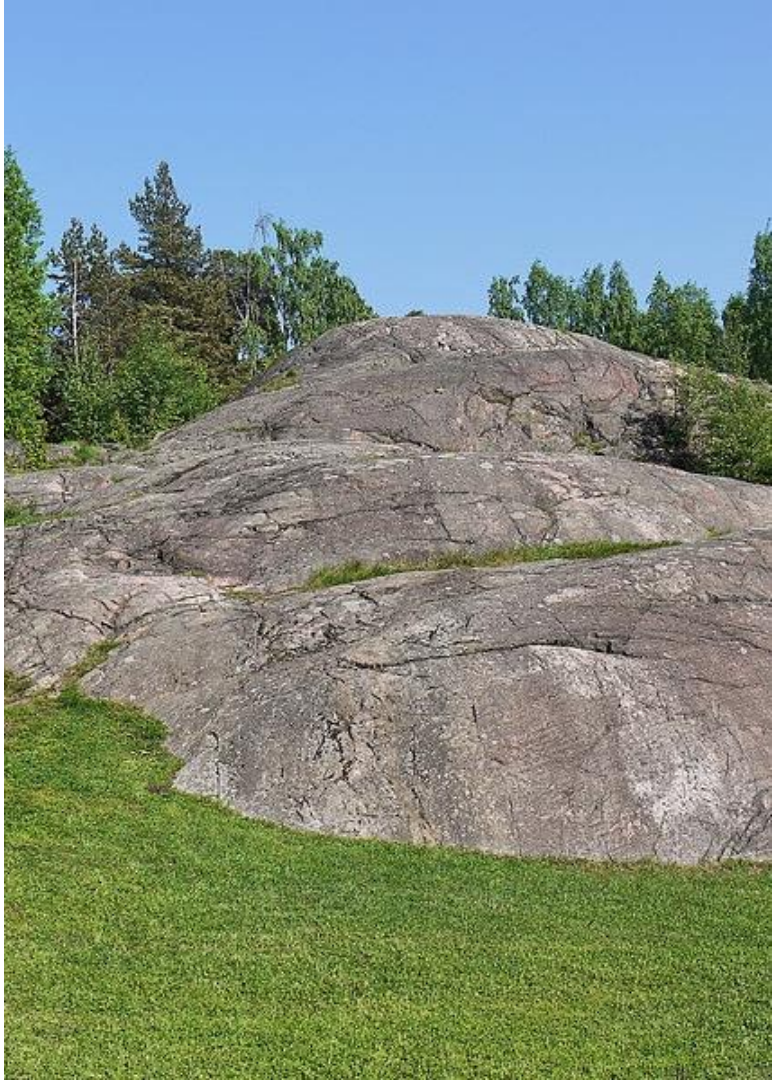


Fig. 1. Major tectonic units in the **Fennoscandian Shield.** The numbers (1-4) represent the subdivision by Hermansson et al. (2007) of the central **Fennoscandian Shield** into four tectonic domains (see text for details). The location of the Forsmark area is indicated by the arrow. The map has been modified after Winterhalter et al. (1981) and Koistinen et al. (2001).

Download as: GeoTIFF+Metadata | Download | ScienceDirect | Read-Only PDF

A Research Proposal

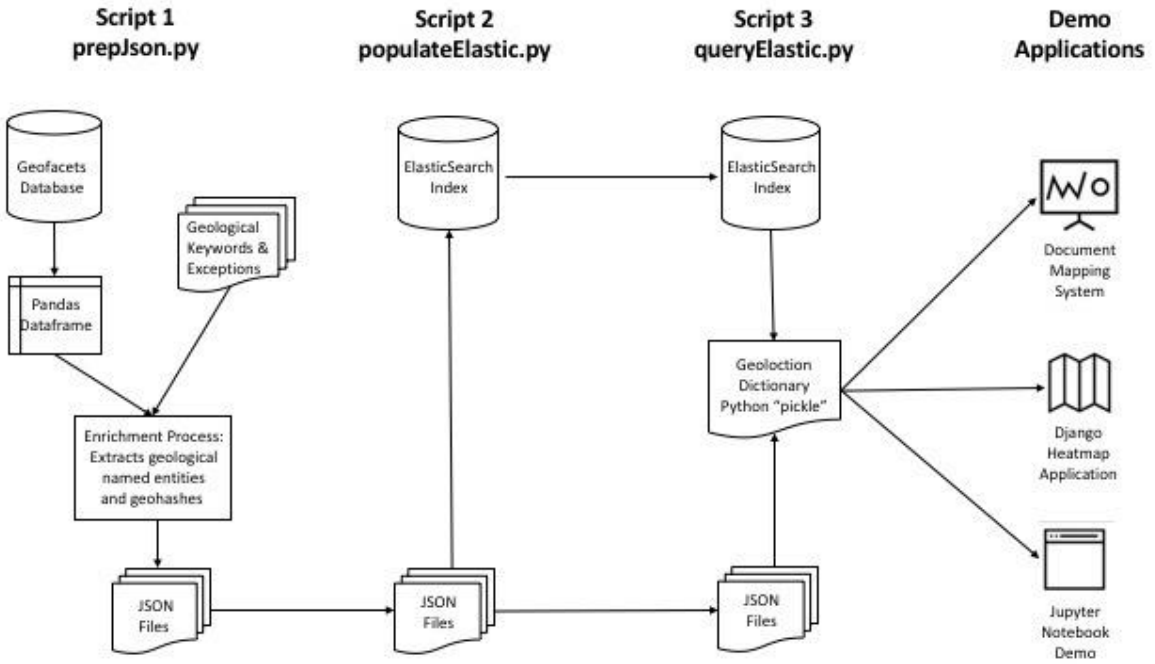
- Roll our own geolocation engine
- Based on Geofacets data:
  - Polygons for Areas of Interest
  - Captions of map insets
  - Additional article metadata?



Open Source Tools

- Geohash Algorithm
- MySQL
- Elastic Search
- Google Maps API
- Python
  - Pandas
  - Shapely
  - Geohash
  - SpaCy
  - GeoPy
  - Django

System Architecture



Starting Data

- Lat / Long Bounding Boxes (Coordinate System Normalized)
- Lat / Long Center Points
- Caption Text
- ~~Other article metadata~~
- ~~Basin Names~~

Code Dive!!

- Panda's Lambdas
  - For Named Entities
  - For GeoHashes
  - For GeoNeighbors
- Some Data Cleanup
- Load to Elastic
- Query Elastic
- Dump to Django

SpaCy Lambdas for Noun Phrases

```
201
202         # Lambda to get those noun_phrases
203         df['noun_phrases'] = df.apply (lambda row: getNounPhrases (row) if(pd.notnull(row['CAPTION'])) else row, axis=1)
204
205         print("Getting geo noun phrases")
206         logger.write("Getting geo noun phrases\n")
207         now = time.strftime("%a, %d %b %Y %H:%M:%S +0000", time.localtime())
208         print(now)
209         logger.write(now+"\n")
210         # Load up your geotypes
211         # These get passed to getGeoNounPhrases
212         with open('geoFeatures.lst') as file:
213             geotypes = file.read().splitlines()
214
215         with open('exceptions.lst') as file:
216             exceptions = file.read().splitlines()
217
218         # Now pull geo specific noun phrases from our set of noun_phrases
219         df['geo_phrases'] = df.apply (lambda row: 1, axis=1)
220         df['geo_phrases'] = df.apply (lambda row: getGeoNounPhrases (row, geotypes, exceptions), axis=1)
221
```

Using SpaCy's Noun Chunker

```
# Function to get noun phrases from captions
def getNounPhrases (row):
    nps = []
    pasttoken = None
    doc = nlp(row['CAPTION'])
    for np in doc.noun_chunks:
        nps.append(" " + html.unescape(np.text) + " ")
        if pasttoken and pasttoken.text + " of " + np.text in doc.text:
            doublenp = pasttoken.text + " of " +  np.text
            nps.append(" " + html.unescape(doublenp) + " ")
        if any(x in " " + np.text.lower() + " " for x in [" gulf ", " sea ", " valley "]):
            pasttoken = np
        else: pasttoken = None
    return "|||".join(nps)
```

Proper Noun Hack, and in Geo Trigger List

```
61  # Get Geo specific noun phrases from noun phrases
62  def getGeoNounPhrases (row, gts, exceptions):
63      try:
64          geonps = []
65          if isinstance(row['noun_phrases'], str):
66              for np in row['noun_phrases'].split("|||"):
67                  nothe = np.replace("The", "")
68                  if (np.strip() not in exceptions
69                     and any(letter.isupper() for letter in nothe)):
70                      for geotype in gts:
71                          if (" " + geotype + " ") in np.lower():
72                              geonps.append(np.strip())
73          return [list(set(geonps))]
74      except ValueError:
75          return None
```

500+ Geological Keywords

```
111  crater
112  crater lake
113  crater lakes
114  craters
115  craton
116  creek
117  crest
118  crevasse
119  cuesta
120  cuestas
121  current
122  cutbank
```

```
167  fjord
168  fjords
169  flat
170  flats
171  floodplain
172  flow
173  fluvial landform
174  fluvial landforms
175  fold
176  fold belt
177  footwall
178  ford
```

```
483  shutter ridge
484  sill
485  sinkhole
486  site
487  slab
488  slide
489  slope
490  slopes
491  snowfield
492  sound
493  spit
494  spring
```

GeoHash

- Alphanumeric strings
- 32 Cell Grids
- Represent Lat / Longs
- Longer strings == More Precision
- Up to 12 characters (≤ 37.2mm × 18.6mm)

- Image generated at: http://geohash.gofreerange.com

Calculate Approx. Area of Map

```
87  # Estimate area of each area of interest (in km)
88  def tryGetArea (row):
89      try:
90          bl = row[['SW_COORD_LAT','SW_COORD_LNG']]
91          br = row[['SE_COORD_LAT','SE_COORD_LNG']]
92          tr = row[['NE_COORD_LAT','NE_COORD_LNG']]
93          tl = row[['NW_COORD_LAT','NW_COORD_LNG']]
94
95          width = vincenty(bl, br).kilometers
96          height = vincenty(bl, tl).kilometers
97          area = width*height
98          return area
99      except ValueError:
100         return None
101
```

Get Geohash For Center

- At a precision appropriate to the Map Area

```
255         df['geohash'] = df.apply (
256             lambda x: geohash.encode(x.CENTER_COORD_LAT, x.CENTER_COORD_LNG, precision=3) if x.polar == True else
257                     (geohash.encode(x.CENTER_COORD_LAT, x.CENTER_COORD_LNG, precision=2)if x.area > 1500000 else
258                     (geohash.encode(x.CENTER_COORD_LAT, x.CENTER_COORD_LNG, precision=3)if x.area > 100000 else
259                     (geohash.encode(x.CENTER_COORD_LAT, x.CENTER_COORD_LNG, precision=4)if x.area > 5000 else
260                      geohash.encode(x.CENTER_COORD_LAT, x.CENTER_COORD_LNG, precision=5)))), axis=1)
```

Recursively fill out bounding box

```
# Recursive bit for getNeighbors
def processHash (gh, counter, error, polygon, outhashes):
    point = Point(geohash.decode(gh))
    if polygon.contains(point):
        outhashes.append(gh)
        newhashes = geohash.neighbors(gh)
        for h in newhashes:
            if h not in outhashes:
                processHash(h, counter, error, polygon, outhashes)
    return outhashes
```

Some rather large regions


```
{
  "M_ID": "S0301-9268(83)80003-5_2",
  "CAPTION": "Fig. 2. The Rio de la Plata Province in southern Uruguay.",
  "geo_phrases": [
    "The Rio de la Plata Province"
  ],
  "neighbors": [
    "69y",
    "6dp",
    "6f1",
    "6f0",
    "6cc",
    "6cb",
    "6dn",
    "69z"
  ],
  "area": 138996.6025274768,
  "NW_COORD_LAT": -32.571869,
  "NW_COORD_LNG": -58.391544,
  "NE_COORD_LAT": -32.571869,
  "NE_COORD_LNG": -53.211308,
  "SW_COORD_LAT": -35.2295,
  "SW_COORD_LNG": -58.391544,
  "SE_COORD_LAT": -35.2295,
  "SE_COORD_LNG": -53.211308
},
```

And some very small regions


```
  {
    "M_ID": "S0025-3227(07)00180-6_1",
    "CAPTION": "Fig. 1. Coastal back-barrier sites in south-east Devon showing p
ositions of electrical resistivity survey lines and boreholes. Slapton Sands (A)
; Blackpool Sands (B). &#169; Crown Copyright/Database right 2006. An Ordnance S
urvey/EDINA supplied service. Coordinates are Ordnance Survey of Great Britain N
ational Grid System and Latitude and Longitude.",
    "geo_phrases": [
      "Slapton Sands",
      "Blackpool Sands"
    ],
    "neighbors": [
      "gbvqh"
    ],
    "area": 7.5509389056,
    "NW_COORD_LAT": 50.294758,
    "NW_COORD_LNG": -3.666299,
    "NE_COORD_LAT": 50.294758,
    "NE_COORD_LNG": -3.635169,
    "SW_COORD_LAT": 50.264175,
    "SW_COORD_LNG": -3.666299,
    "SE_COORD_LAT": 50.264175,
    "SE_COORD_LNG": -3.635169
  },
```

Elastic Search

- Stuff all this JSON in
  - Map the geo_phrases as keywords
  - Map the geohashes as geo_points
- Get a list of *all* geohashes
- Query every hash, get keyword list and count as aggregations

Reduce and invert

- Reformat this data such that:
  - 1 Row per geological named entity
  - A list of associated hashes / counts:
    the Little Goose Creek Complex,c2j2=1|c2j2t=1|9rtmk=1|9rtmh=1|9rtmm=1|9rtm=1|9rtmj=1|9rtmt=1|9rtms=1,
  - Dump to a really simple Django model

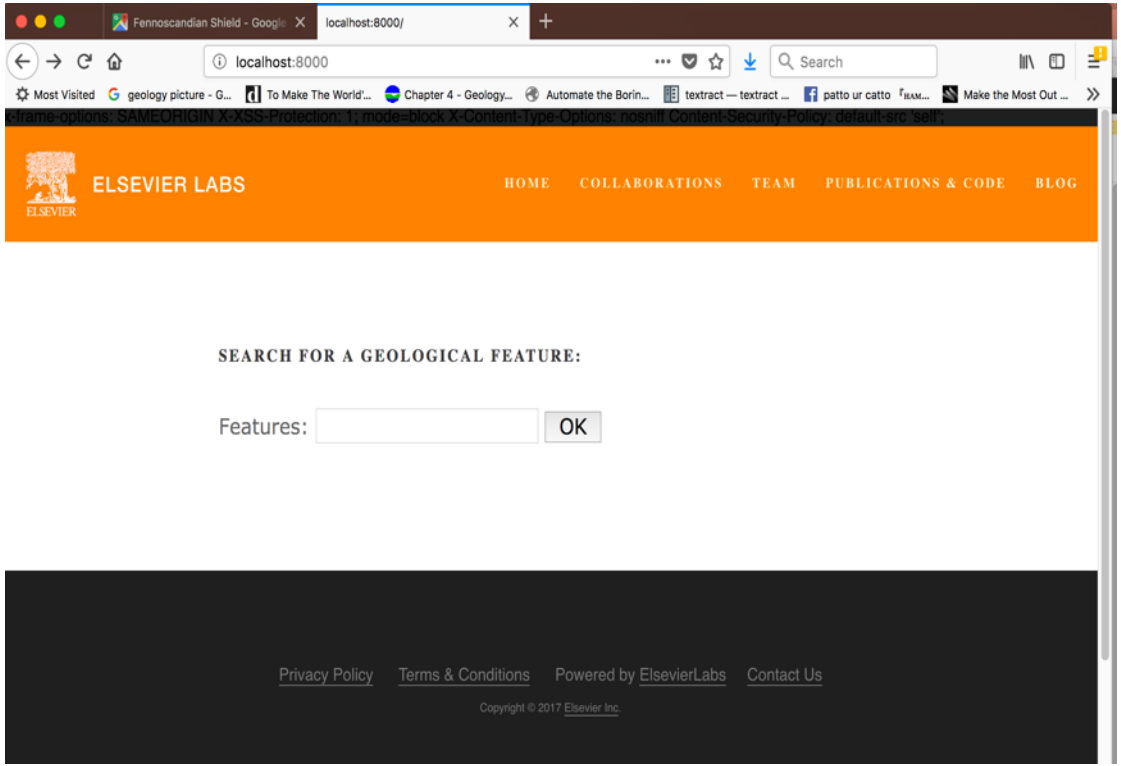
With function to format for Google Maps

```
7  class Feature(models.Model):
8      name = models.CharField(max_length=500)
9      geos = models.TextField(max_length=2000000)
10
11     def geos_as_list(self):
12         hash_list = self.geos.split('|')
13         latlng_list = []
14         for gh in hash_list:
15             #location: new google.maps.LatLng(37.782, -122.447), weight: 10
16             latlng_list.append("{location: new google.maps.LatLng"
17                                +str(geohash.decode(gh.split('=')[0]))
18                                +", weight: " + gh.split('=')[1] + "},"
19                                )
20         latlng_list[-1] = latlng_list[-1].rstrip(",")
21         return latlng_list
```

Some additional cleanup

- Current dataset is still noisy, partly due to:
  - Noun chunker inconsistencies
  - Case sensitivity
  - Presence in initial articles
  - Stray punctuation symbols
  - Abbreviations tied to map annotations
    e.g. *SAF-San Andreas fault*

Geocoding Demo



Statistics

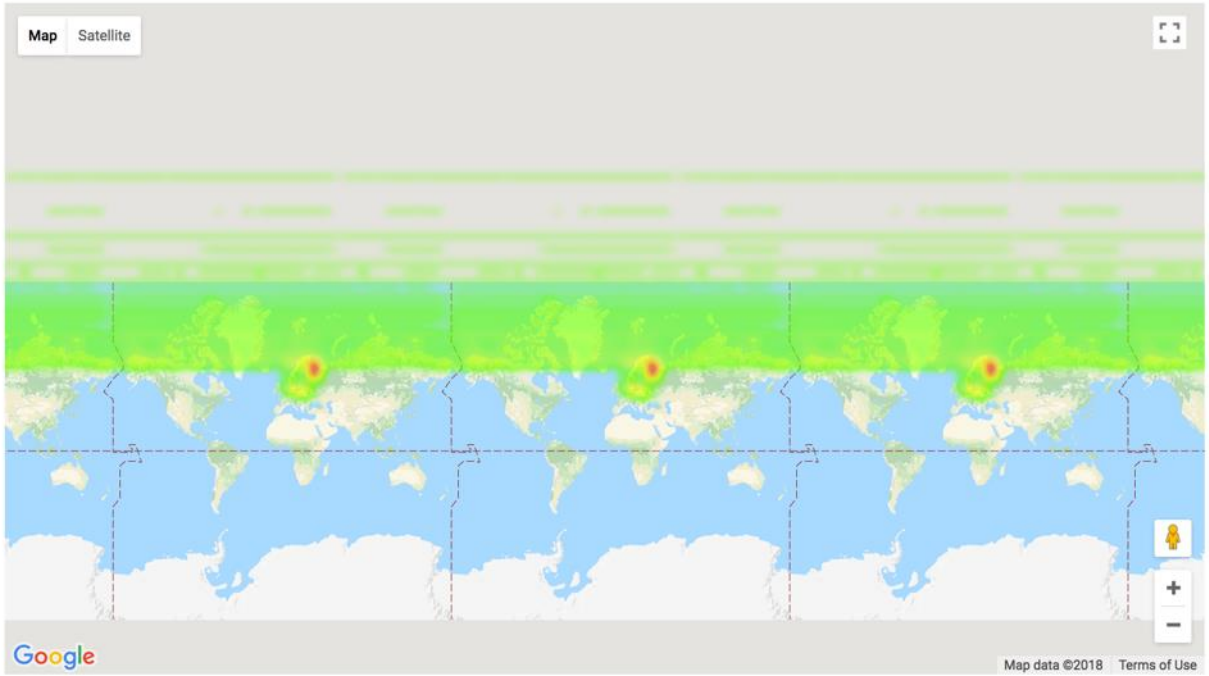
- In the holdout 20% test data:
  - About 1/2 of text strings matched a location
  - About 1/3 of text strings contain a “trigger phrase”
- 285,453 entries in the current Drupal index
  - Still a fair amount of noise and duplication
  - Still likely over 100k unique Geology Entities

Challenges

- Data Cleanup
  - Noun Chunker
  - Punctuation & Case
- Optimal Geohash precision
- Stippled hotspots converted from geohash
- Polar Area Maps

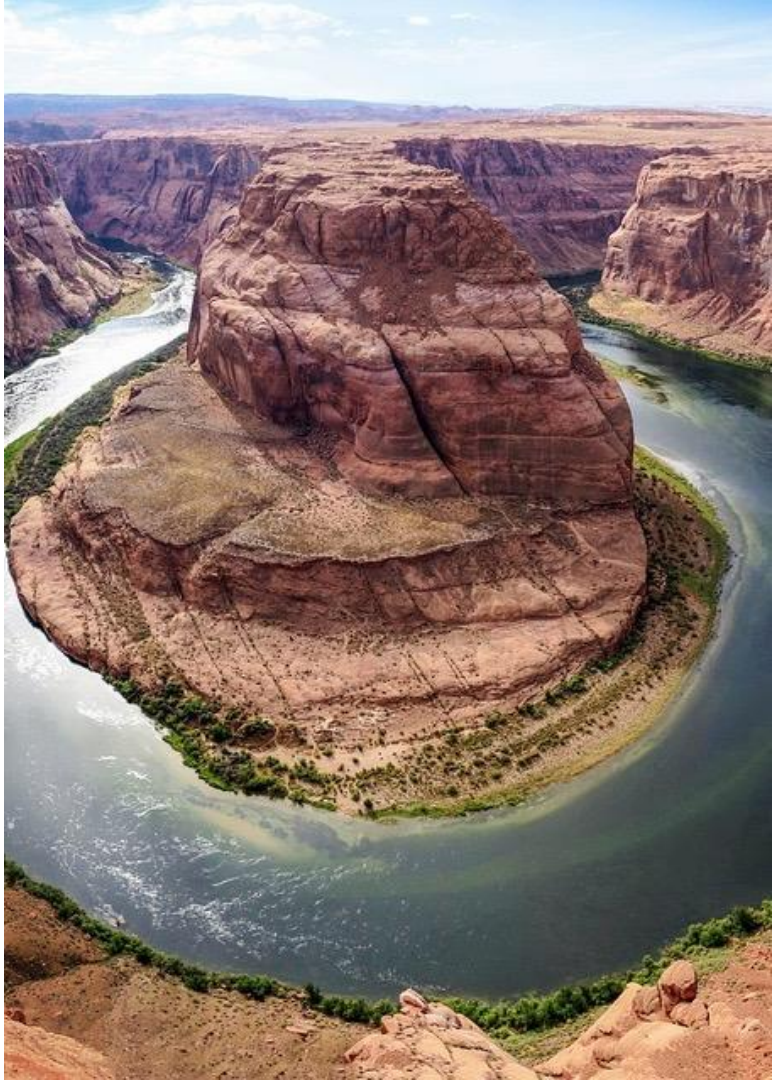
Polar Area Maps

Baltic Shield

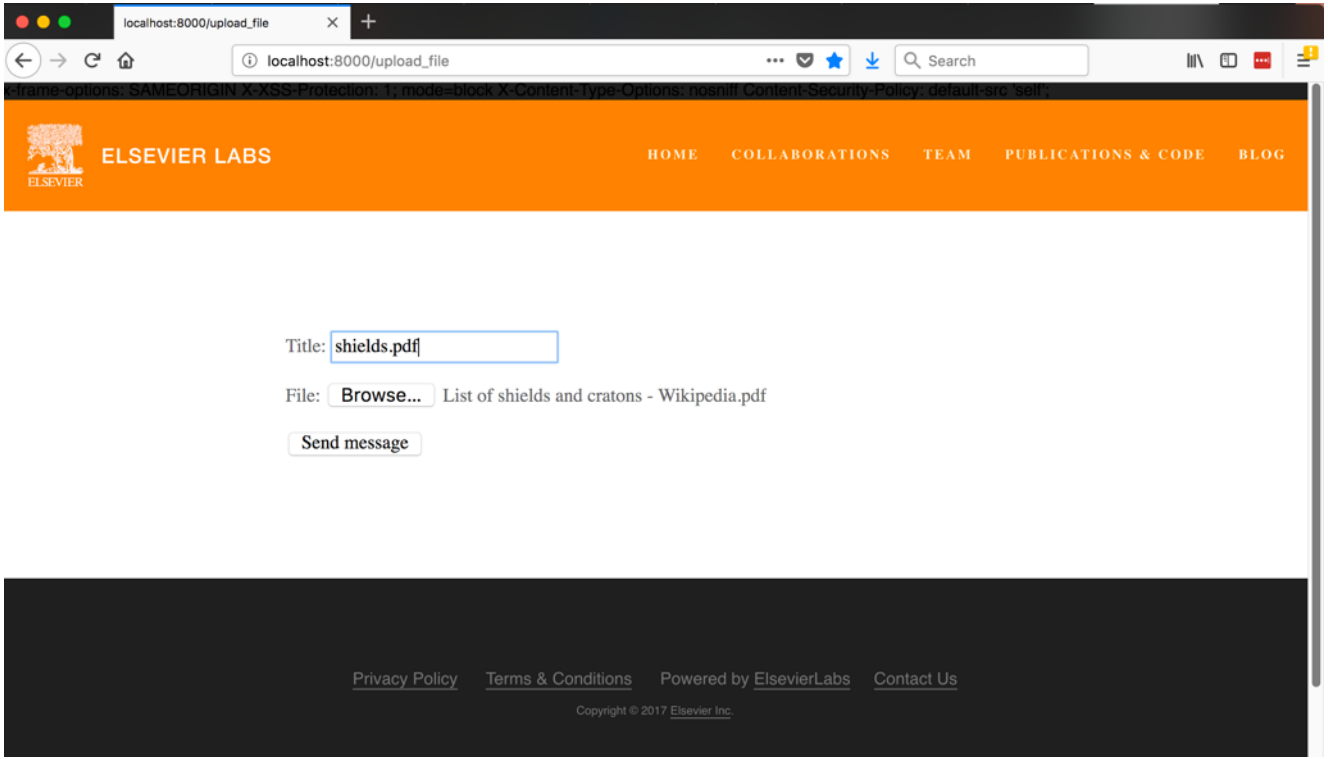


Use Cases

- Scaling Geolocation Work
  - Public Domain Data
  - 3rd Party Data Integration
  - Non-Cartographic Content
- Analogs
- Document Analysis



Document Analysis Demo



Conclusion

- Geofacets Product
- Manual Georeferncing of Map Insets from Geology Articles
  - Difficult to Scale to New Data Sources
  - Georeferening, Not Geocoding
- How to build a Geolocation Engine for Text Based Geocoding

ELSEVIER

Thank you

Please Evaluate the Sessions!
Sign in and vote at 2018.foss4g-na.org

Corey A Harper – Elsevier Labs
@chrpr – c.harper@elsevier.com

Geology Images all CC-BY and CC0

- https://pixabay.com/en/desert-formation-geology-erosion-2480655/
- https://pxhere.com/en/photo/290464
- https://commons.wikimedia.org/wiki/File:Granodiorite_of_the_Fennoscandian_Shield.jpg
- https://commons.wikimedia.org/wiki/File:A_melhor_coisa_pra_fazer_em_Noronha.jpg
- https://www.publicdomainpictures.net/en/view-image.php?image=106887&picture=geology
- https://www.maxpixel.net/Geology-Horseshoe-Desert-High-Angle-Shot-1867005