Geo::Parser::Text

Geo parsing and Geo coding locations from text.

Ervin Ruci - YAPC::NA 2016

About this talk

1. I gave the first version of this talk at FOSDEM 2016 (Geospatial Devroom)
2. Some members of the Audience expressed surprise at me choice of Perl for such an application (?!)
3. So, here is an expanded version aimed at a Perl friendly audience
4. This talk (and various test data) are hosted at `https://github.com/eruci/openaddresses/tree/master/test`
5. The main application is available as a server image on AWS and `https://metacpan.org/pod/Geo::Parser::Text`

Text => Geo Locations in Text

This and That and the Other street in Porters Lake Nova Scotia

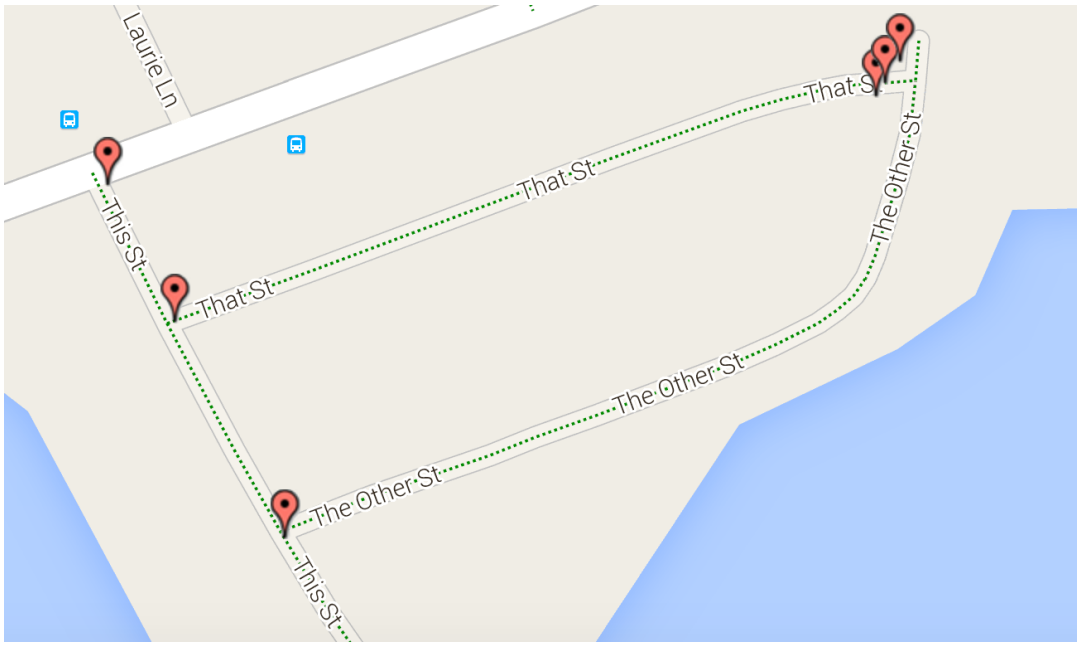
Q. How many locations are in this text?

The Problem: Text => Geo Locations in Text

This and That and the Other street in Porters Lake Nova Scotia

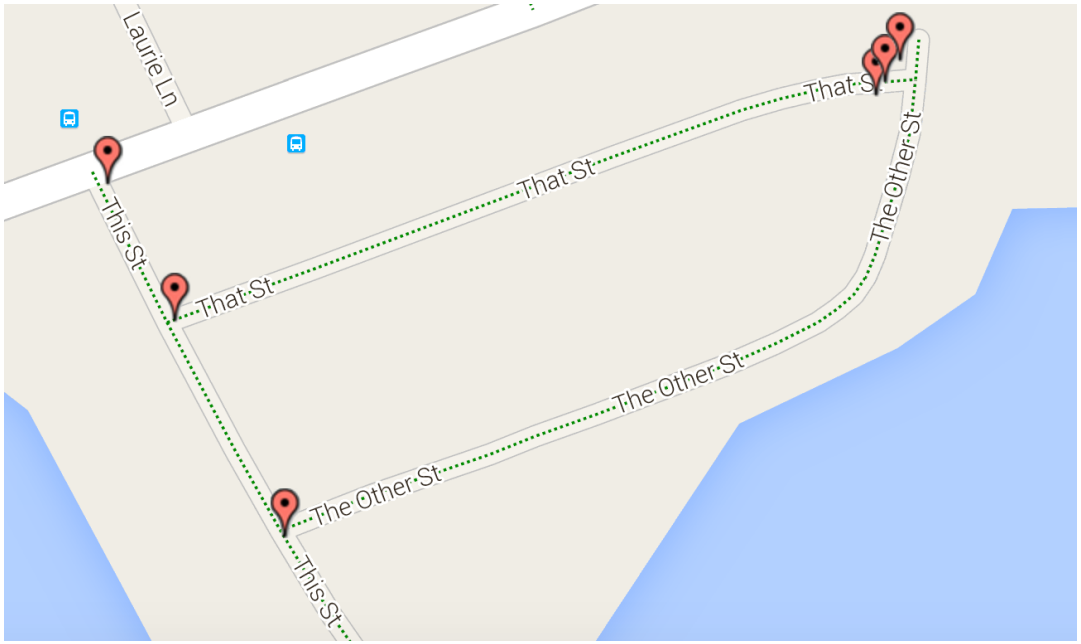
Q. How many locations are in this text?

A. 6. http://geocoder.ca



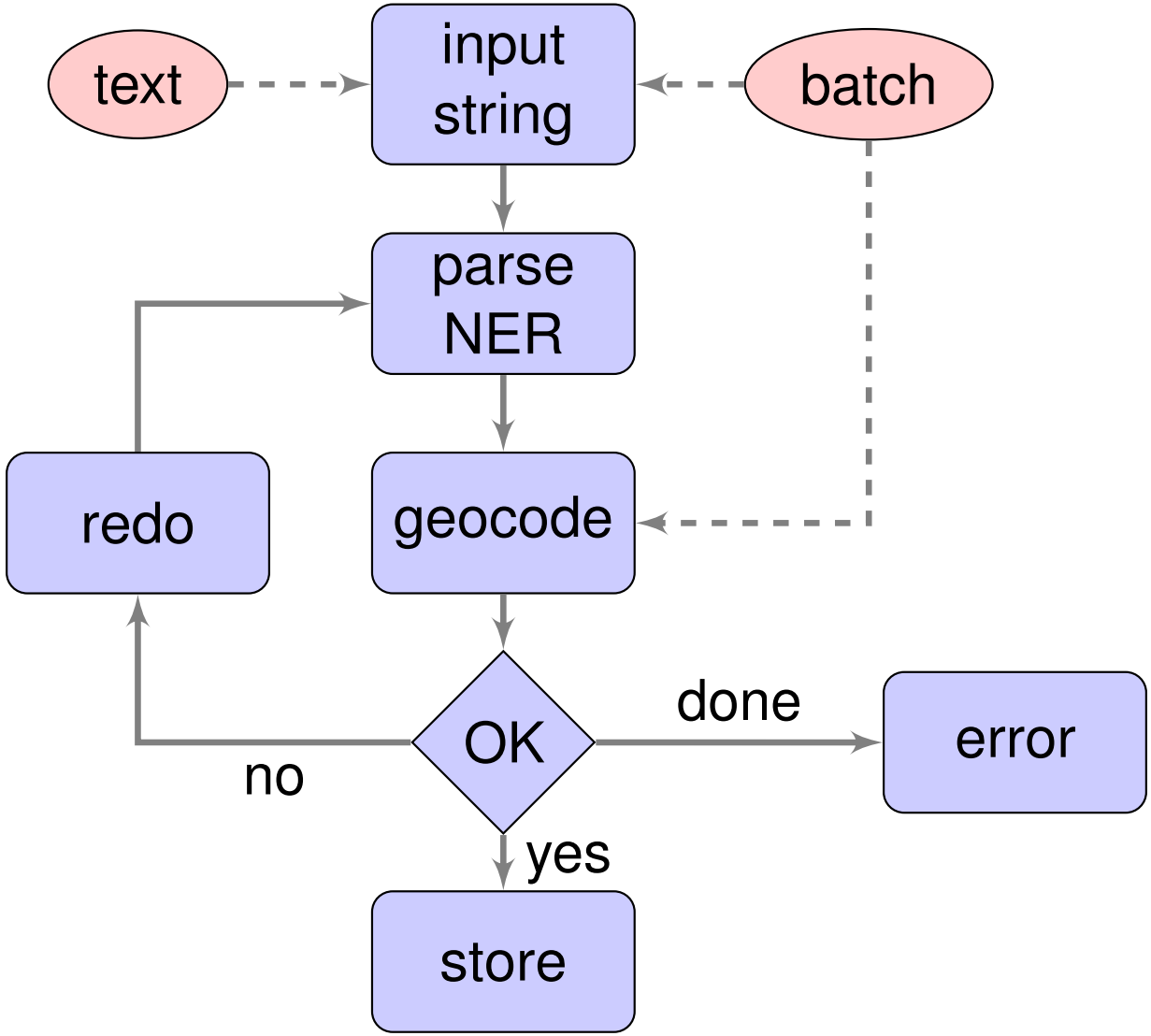
The Problem: Text => Geo Locations in Text

1. THIS ST AND THAT ST, PORTERS LAKE, NS
2. THIS ST, PORTERS LAKE, NS
3. THAT ST, PORTERS LAKE, NS
4. THE OTHER ST AND THAT ST, PORTERS LAKE, NS
5. THE OTHER ST AND THIS ST, PORTERS LAKE, NS
6. THE OTHER ST, PORTERS LAKE, NS



Reqs: Identify addresses, intersections, city names, province/state.

The Solution: Geoparsing and Geocoding



Geocoding Spans Many Fields

1. linguistics (matching/translating across different languages)
2. data processing (normalization, standardization and input)

2.1 data structures (R-trees, KD-trees,...)

3. natural language processing (parsing, named entity recognition)
4. computational geometry (point in polygon)
5. pattern recognition (fuzzy match)
6. geography (dealing with projections)
7. Ai (learning, hidden markov models)

.. .... and a few others (tokenization, data cleanup, UI..)

AND Testing, testing and more testing

And there are Many Geocoders.

And Many more are being built, Plus a few I've tested:

1. Google Geocoder (Coverage: 99%, Accurate 93%) (Canada)
2. HERE.com (Coverage 98%, Accurate 92%) (Canada)
3. Nominatim (Coverage 80%, Accurate 57%) (Canada)
4. Geocoder.ca (Coverage 99%, Accurate 94%) (Canada)
5. Geocode.xyz (Coverage 80%, Accurate 58%) (Spain)
6. Mapzen.com (Coverage 86%, Accurate 80%) (Spain)

Download test data and results here: https://github.com/eruci/openaddresses/tree/master/test

Why create a new Geocoder?

No Geocoder does it all. Google Geocoder (presumably the most complete in the bunch) does not

1. Geocode parcel data (avail as opendata in Canada and USA)
2. Extract location data from text
3. Do address parsing and standardization (incl postal codes)
3. Return all addresses that match a partial address in text
4. Provide 100% coverage (open problem)
5. Provide 100% accuracy (open problem too)

Accuracy and Coverage Differ Because

a. Geocoding/Geoparsing is an imprecise process and various Geocoders fail in various ways.
b. Ambiguities, incomplete data, incorrect data, software bugs, are the main causes

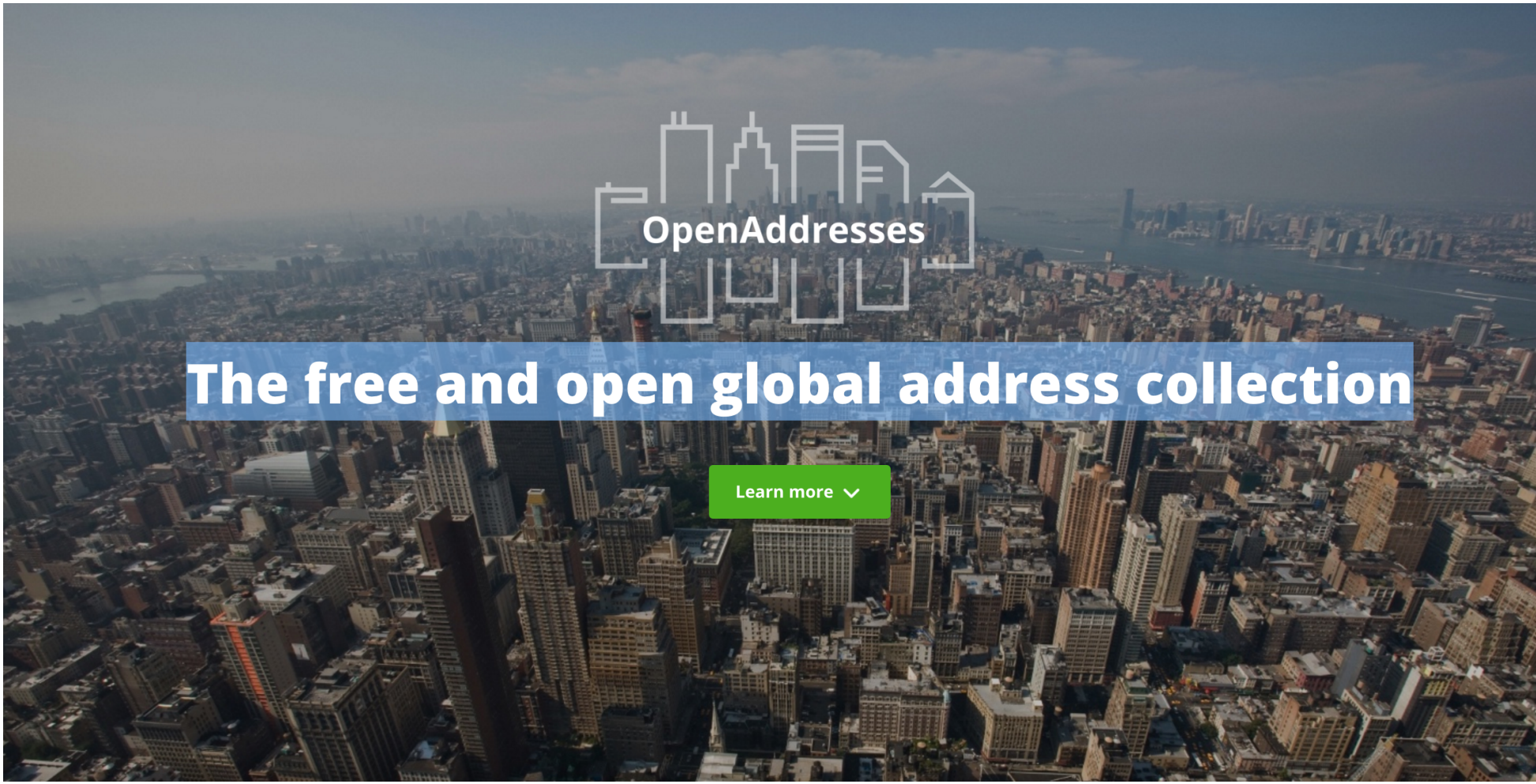
The key ingredients of the solution

1. DATA
2. A good parser

Parsing

1. Fuzzy vs Exact (correct spelling errors)
2. Partial vs Complete (fill in missing location entities)
   - Quick demo: `http://geocoder.ca/textscan`

Data

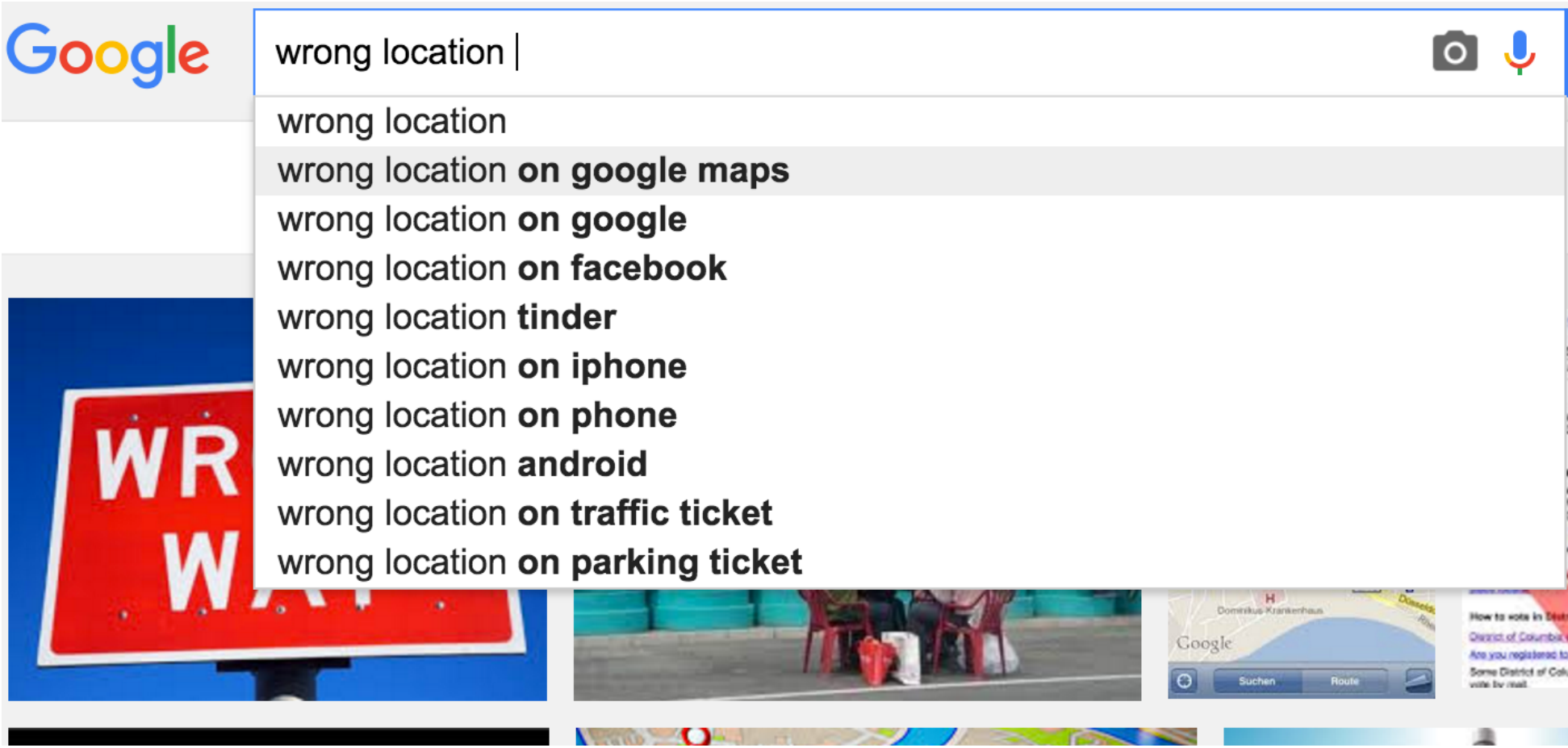


over 218,548,165 addresses (was half that only 6 months ago)

Even Google Maps (presumably the best) Fails!

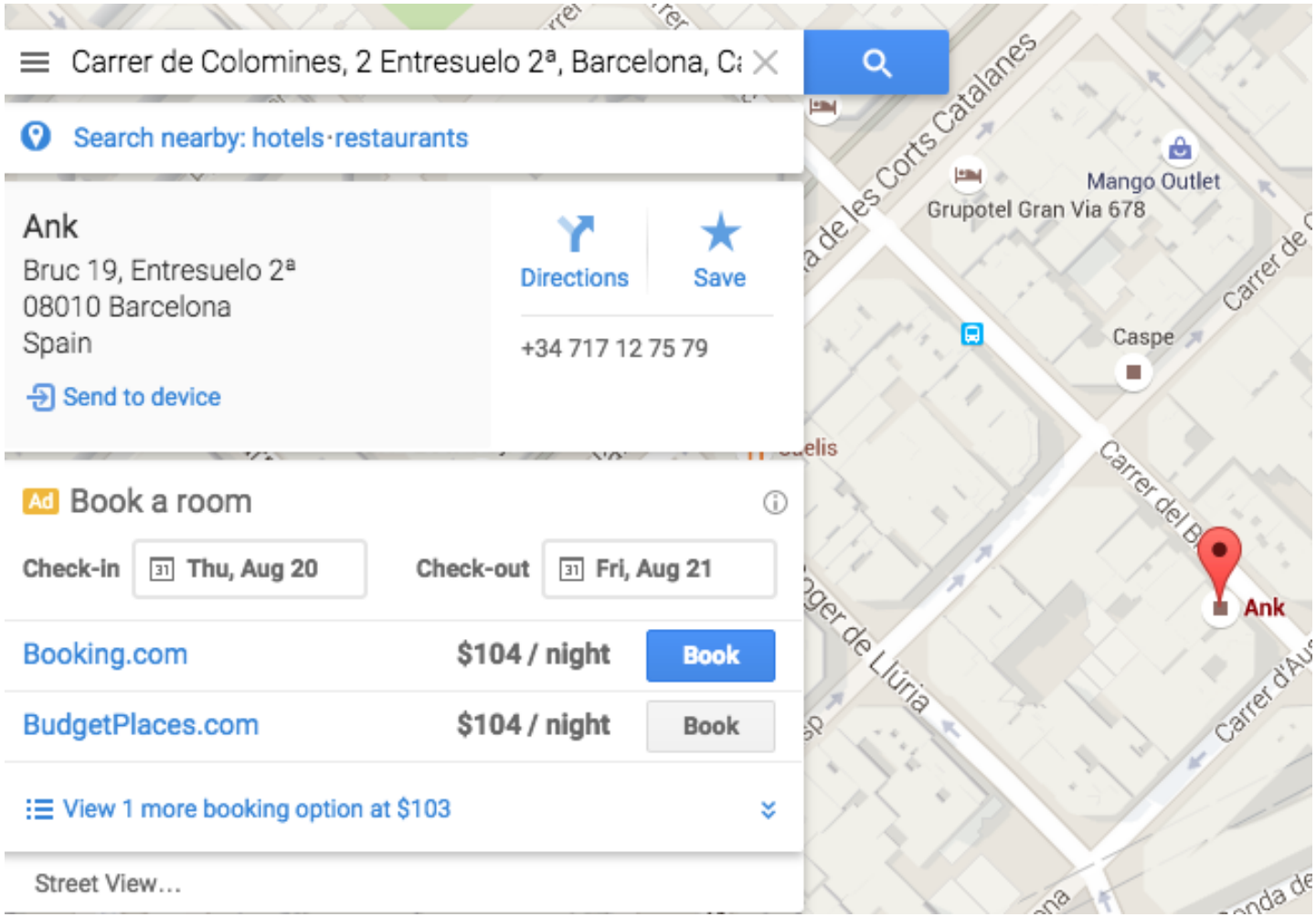
- Even in well mapped big cities.

Even when you Google wrong location you get:



: Wrong Location Google Maps!.

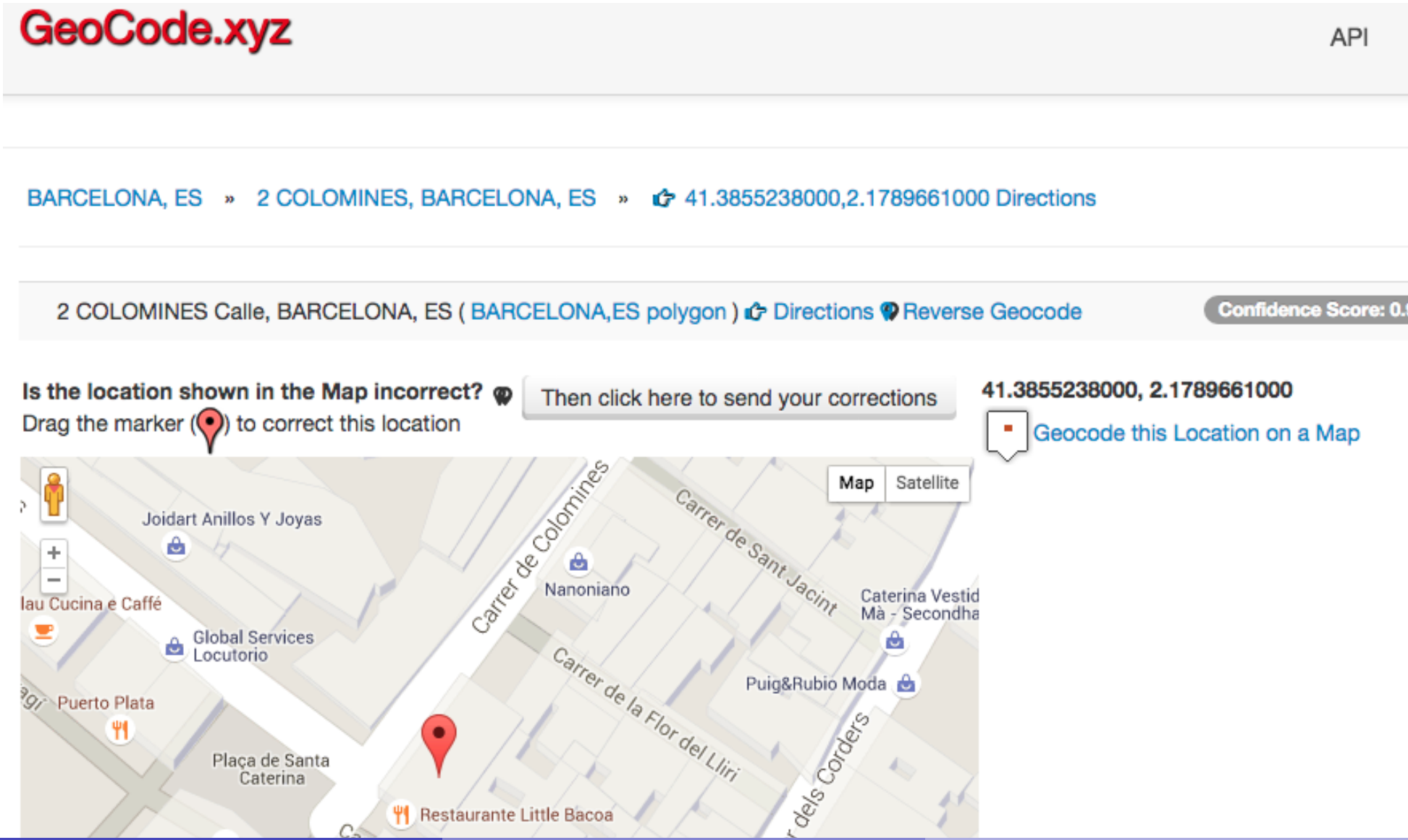
Carrer de Colomines, 2 Entresuelo 200AA, Barcelona, Catalunya 08003, Spain



: my Airbnb in Barcelona: Wrong!.

http://Geocode.xyz - A geocoder for the EU

Fixed with:



Correct, but, Coverage for Spain is limited...

: [Data]

For the latest coverage map see `http://openaddresses.io`

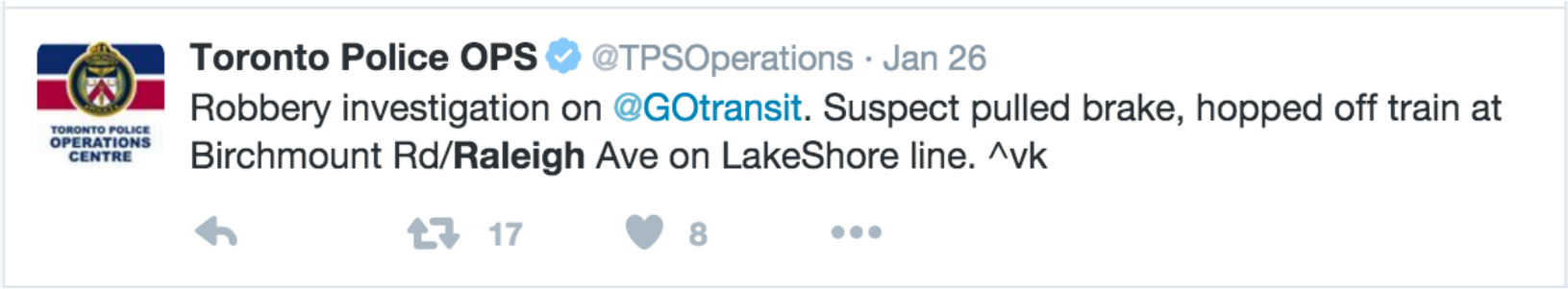
Geoparsing and Geocoding in unstructured text

Text (from wikipedia entries to microblog posts) => Geocoded Locations.

1. extraction
2. disambiguation
3. geolocation

Demo: `http://geocode.xyz`

In case internet does not work for the demo

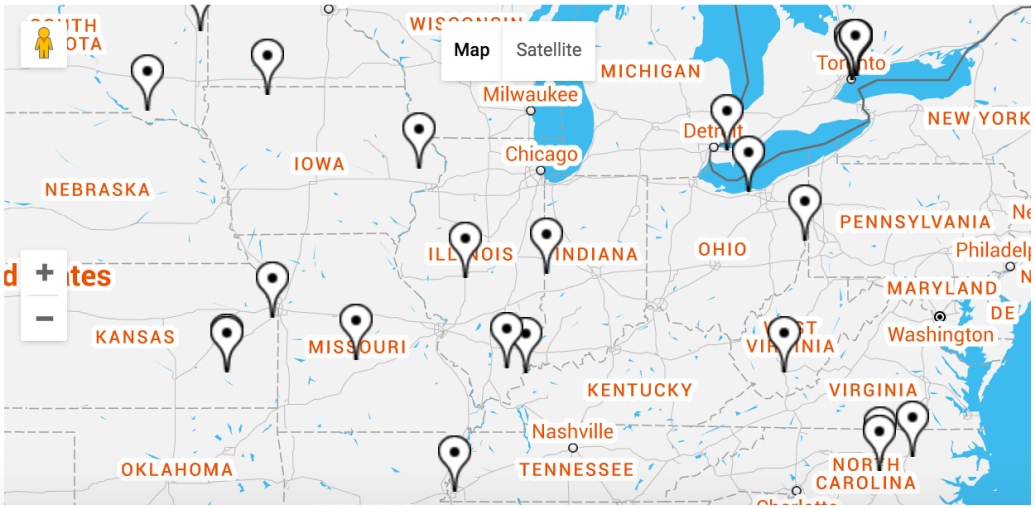


: [Twitter Feed Demo geocoder.ca]

40 locations within a 3484.180 km radius, found in this text:

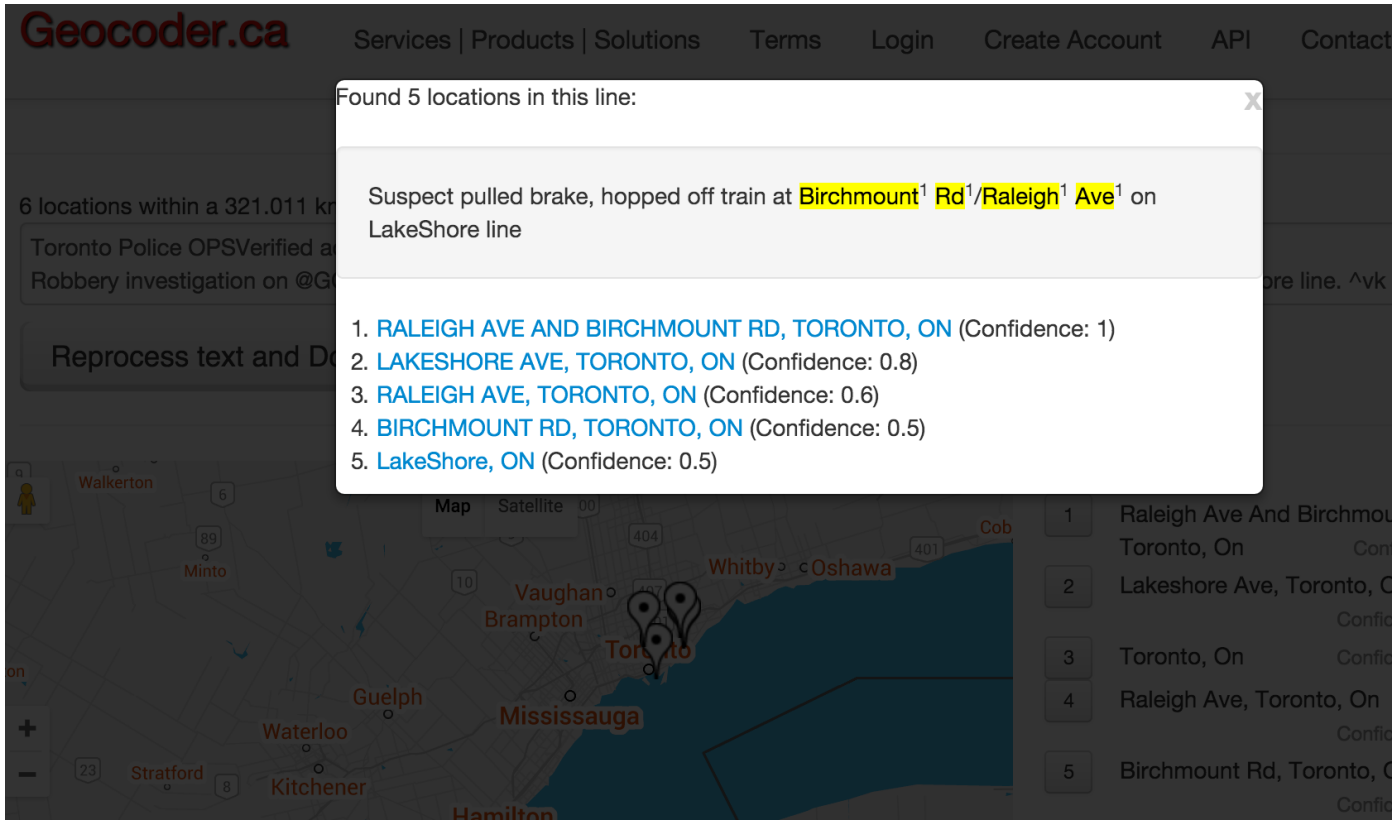
Toronto Police OPSVerified account @TPSOperations Jan 26
Robbery investigation on @GOtransit. Suspect pulled brake, hopped off train at Birchmount Rd/Raleigh Ave on LakeShore line. ^vk

Reprocess text and Download results | On a Map



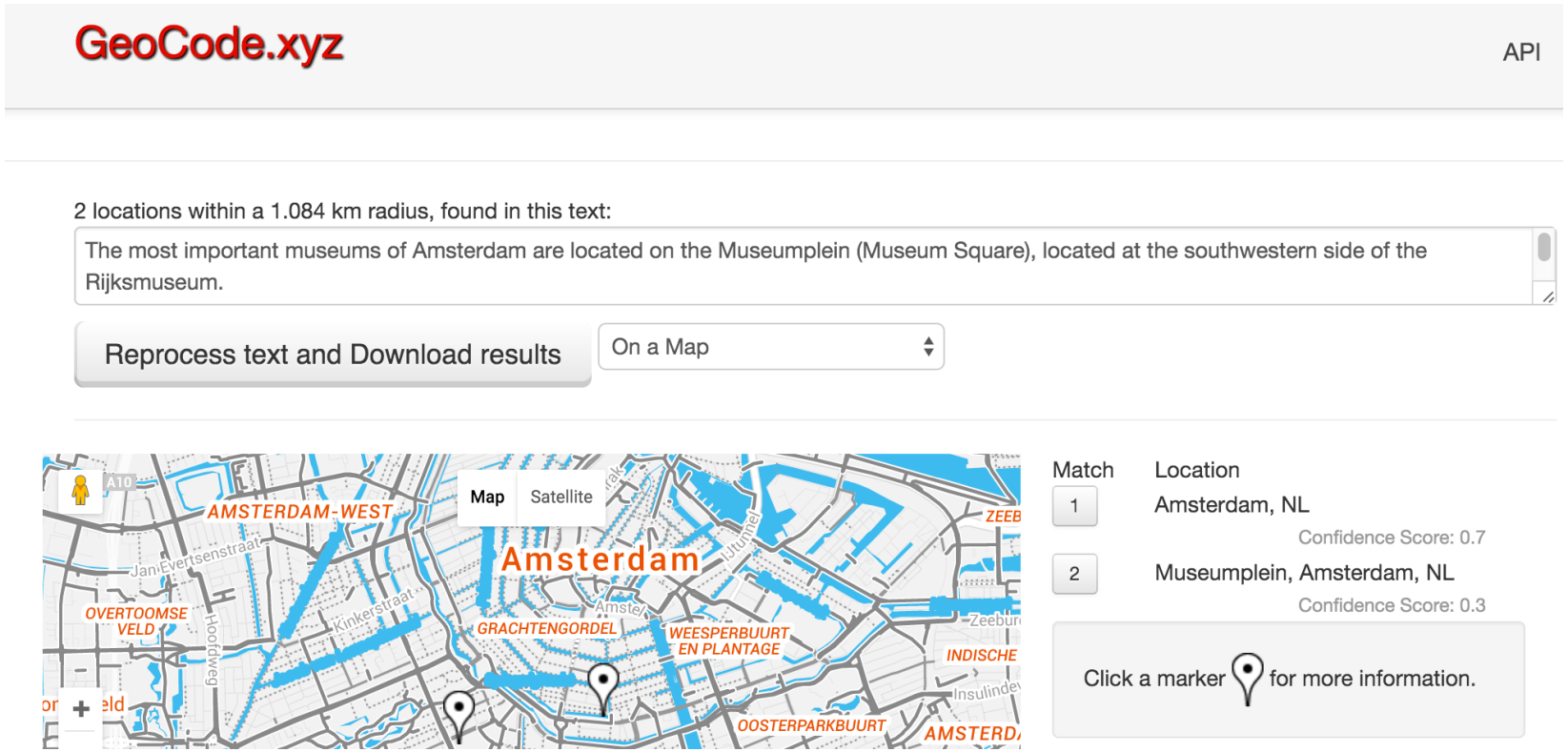
Match	Location	
1	Raleigh Ave And Birchmount Rd, Toronto, On	Confidence Score: 1
2	Raleigh, Nc	Confidence Score: 0.8
3	Toronto, On	Confidence Score: 0.8
4	Raleigh, Ms	Confidence Score: 0.8
5	Raleigh, Fl	Confidence Score: 0.8
6	Toronto, Pe	Confidence Score: 0.8
7	Toronto, Ks	Confidence Score: 0.8
8	Toronto, In	Confidence Score: 0.8
9	Raleigh, Wv	Confidence Score: 0.8
10	Toronto, Tx	Confidence Score: 0.8

In case internet does not work for the demo



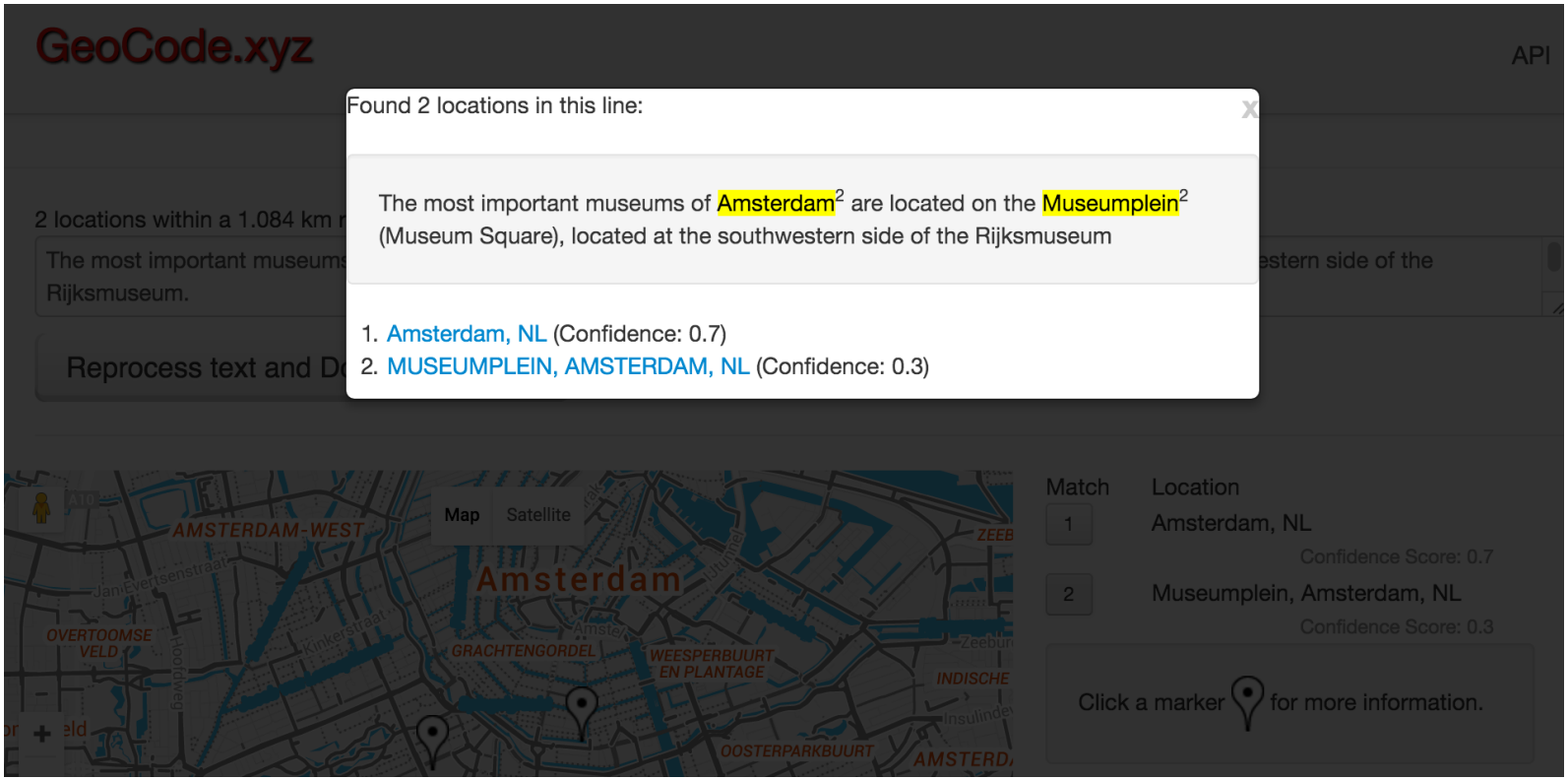
: [Geoparsed]

In case internet does not work for the demo



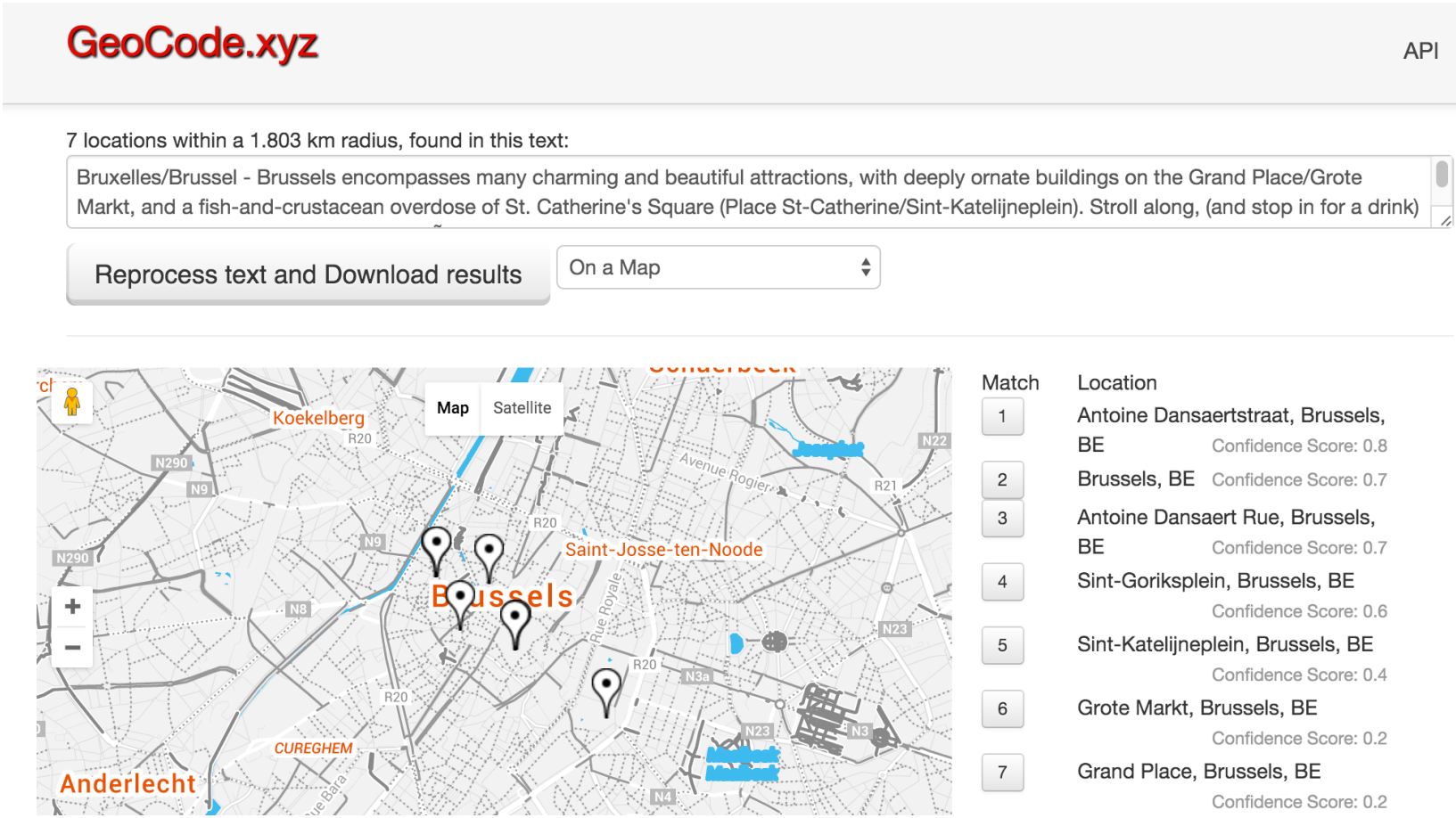
: [Demo geocode.xyz]

In case internet does not work for the demo



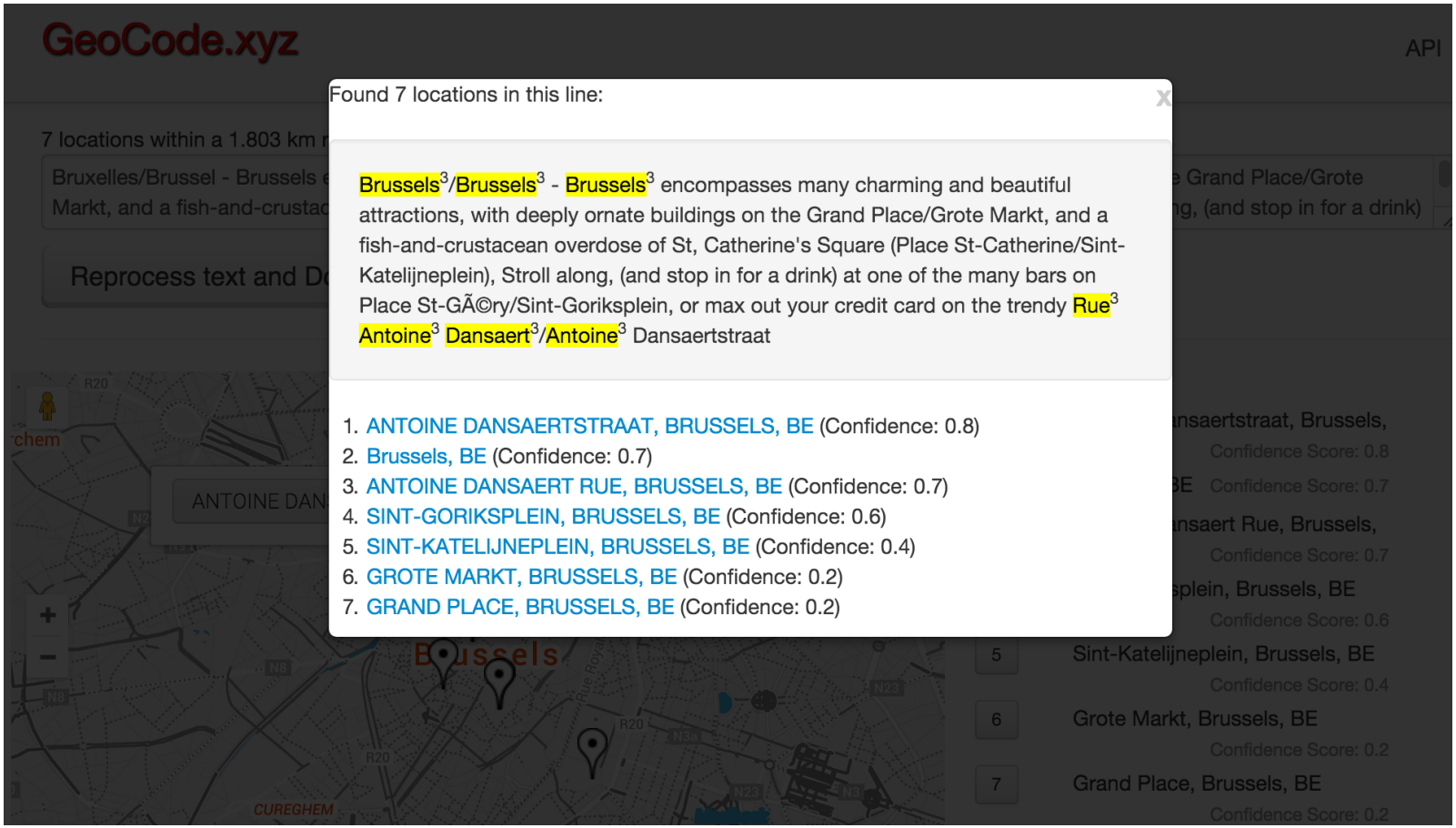
: [Demo geocode.xyz]

In case internet does not work for the demo



: [Demo geocode.xyz]

In case internet does not work for the demo



: [Demo geocode.xyz]

Coding a Geocoder that does this is easy (in theory)

But.. making it recognize over 90% of input at over 90% accuracy requires at least these steps

1. importing and parsing country specific data from openaddresses.io (suffixes, prefixes, city names, numbering schemes)
2. cleaning up errors post import.
3. test and pick away at errors, one at a time

> In theory, there is no difference between theory and practice. But, in practice, there is.

Perl makes easy problems very easy

.. and hard ones, easy. (in both theory and practice)

1. No other language is better at slicing and dicing text.
2. CPAN has lots of valuable bits for any NLP puzzle such as this one
3. Do I need another reason to use perl?

Source Code / Data

Source code and Data: http://geocode.xyz
Just grab the server image on AWS, it is free for a micro instance

Geocoder.ca

If you need help: e: eruci@geocoder.ca twitter: @geolytica
PS. One more thing. The core module is 47355 lines of Perl code.