Geospatial Analysis of Social Media posts with Elasticsearch

Michael Kaisser, Berlin Buzzwords, 26.5.2014
mkaisser @ txtdata.net

Aims

Use Elasticsearch to

- Browse and discover what's happening on Social Media
  - Around you
  - Anywhere else in Berlin
- Perform some advanced analytics to:
  - Detect trends and activity hotspots
  - See what's interesting right now

(This talk is on an introductory level)

Outline

- Social data
  - Twitter, Instagram, Foursquare
  - Fetching, storing and searching with ES
- Geospatial analysis
  - Geohashes
  - Custom-built analysis
- Outlook
  - What could be done better?

The code

- Java code is on GitHub:
  - https://github.com/txtData/socialradar
- Maxim: Keep it simple
- Space for your improvements

Part 1

Social Data

Social Data

- Social Media continuously creates massive amounts of data
- Data can be fetched in real-time
  - Free APIs for low volume, non-commercial use
- Contains text, images, geo-coordinates
  - Lots of things to play with
  - But creating real insights isn’t an easy problem

Social Data

- 500M tweets per day
- 60M Instagram photos a day
- 6M Foursquare check-ins a day

Social Data

- 500M tweets per day
  - Approx. 5% with geo-coordinates
- 60M Instagram photos a day
  - 100% with geo-coordinates*
- 6M Foursquare check-ins
  - 100% with geo-coordinates

Social Data

Twitter data:


```
{
    text: "Looking forward to @berlinbuzzwords!! #bbuzz",
    retweeted: false,
    id: 469094898382434300,
    favorited: false,
    retweet_count: 0,
    created_at: "Wed May 21 12:38:33 +0000 2014",
    id_str: "469094898382434304",
  - user: {
        name: "txtData",
        screen_name: "txtData",
        id_str: "2495955444"
    },
  - coordinates: {
        type: "Point",
      - coordinates: [
            13.425812,
            52.537903
        ]
    }
}
```


(Lots of fields omitted)

Social Data

Instagram data:


```
{
  - location: {
        latitude: 52.496616,
        name: "Berlin Görlitzer Park",
        longitude: 13.438537,
        id: 230259983
    },
    filter: "Mayfair",
    created_time: "1400674462",
    link: http://instagram.com/p/oQjJDfqbmm/,
  - images: {
      - standard_resolution: {
            url: http://origincache-prn.fbcdn.net/926206_6521954351201387_1986134046_n.jpg,
            width: 640,
            height: 640
        }
    },
    caption: null,
    type: "image",
    id: "725234094011597990_32107669",
  - user: {
        username: "royaasoliin",
        full_name: "Roya Asoliin",
        id: "32107169"
    }
}
```


(Lots of fields omitted)

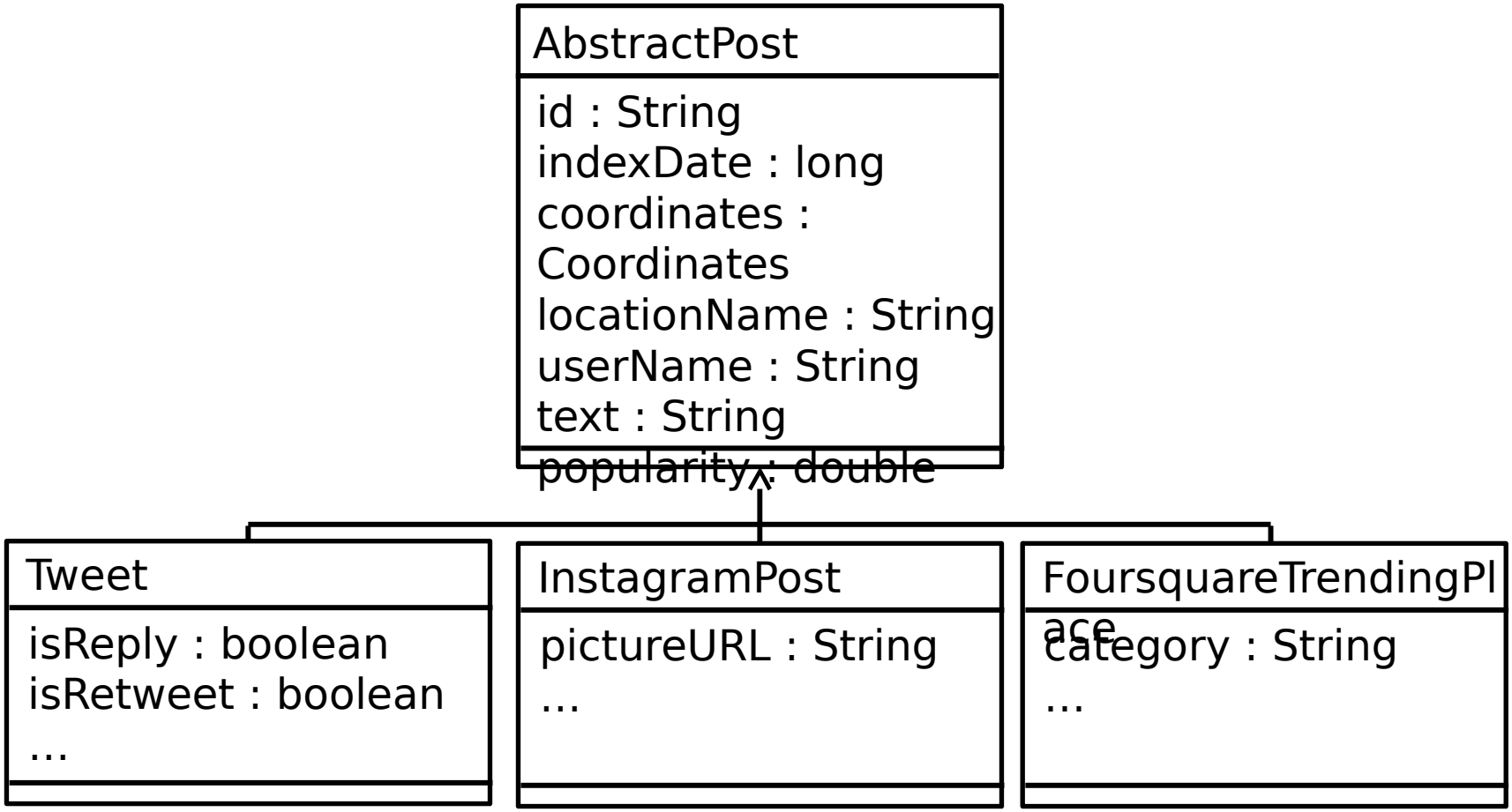
Social Data

Foursquare data:


```
{
    id: "4adcda8ef964a520a34a21e3",
    name: "Wochenmarkt Winterfeldtplatz",
  - location: {
        address: "Winterfeldtplatz",
        lat: 52.495846,
        lng: 13.35444
    },
  - stats: {
        checkinsCount: 2025,
        usersCount: 679,
        tipCount: 26
    },
    url: http://wplatz.winterfeldt-markt.de,
  - hereNow: {
        count: 6
    }
}
```


(Lots of fields omitted)

Social Data


AbstractPost
id : String
indexDate : long
coordinates : Coordinates
locationName : String
userName : String
text : String
popularity : double
Tweet
isReply : boolean
isRetweet : boolean
...
InstagramPost
pictureURL : String
...
FoursquareTrendingPlace
category : String
...

Social Data

Twitter data:


```
{
    text: "Looking forward to @berlinbuzzwords!! #bbuzz",
    retweeted: false,
    id: 469094898382434300,
    favorited: false,
    retweet_count: 0,
    created_at: "Wed May 21 12:38:33 +0000 2014",
    id_str: "469094898382434304",
  - user: {
        name: "txtData",
        screen_name: "txtData",
        id_str: "2495955444"
    },
  - coordinates: {
        type: "Point",
      - coordinates: [
            13.425812,
            52.537903
        ]
    }
}
```


(Lots of fields omitted)


```
{
    id: "TW_469094898382434300",
    indexDate: 1400725433,
  - coordinates: {
        lat: 52.537903,
        lon: 13.425812
    },
    locationName: "Berlin",
    userName: "txtData",
    text: "Looking forward to @berlinbuzzwords!! #bbuzz",
    popularity: 1,
    retweetCount: 0,
    favoriteCount: 0
}
```


(Lots of fields omitted)

Querying ES

Setting up a mapping:

```
{
    id: "TW_469094898382434300",
    indexDate: 1400725433,
  - coordinates: {
        lat: 52.537903,
        lon: 13.425812
    },
    locationName: "Berlin",
    userName: "txtData",
    text: "Looking forward to @berlinbuzzwords!! #bbuzz",
    popularity: 1,
    retweetCount: 0,
    favoriteCount: 0
}
```

```
{
  - posts: {
      - properties: {
          - coordinates: {
                type: "geo_point"
            }
        }
    }
}
```

Querying ES

```
GeoDistanceFilterBuilder gdFilter = FilterBuilders.geoDistanceFilter("coordinates")
        .point(lon,lat)
        .distance(distanceInMeters, DistanceUnit.METERS)
        .optimizeBbox("memory")
        .geoDistance(GeoDistance.ARC);

SearchResponse response = CLIENT.prepareSearch(POST_INDEX)
        .setQuery(QueryBuilders.filteredQuery(QueryBuilders.matchAllQuery(), gdFilter))
        .addSort(new GeoDistanceSortBuilder("coordinates").point(lon, lat))
        .execute()
        .actionGet();
```

```
{
  - filtered: {
      - query: {
          match_all: { }
        },
      - filter: {
          - geo_distance: {
              - coordinates: [
                    13.425812,
                    52.537903
                ],
                distance: "1000.0m",
                distance_type: "arc",
                optimize_bbox: "memory"
            }
        }
    }
}
```

Querying ES

Geo query:

```
{
  - filtered: {
      - query: {
          match_all: { }
        },
      - filter: {
          - geo_distance: {
              - coordinates: [
                  13.425812,
                  52.537903
                ],
                distance: "1000.0m",
                distance_type: "arc",
                optimize_bbox: "memory"
            }
        }
    }
}
```

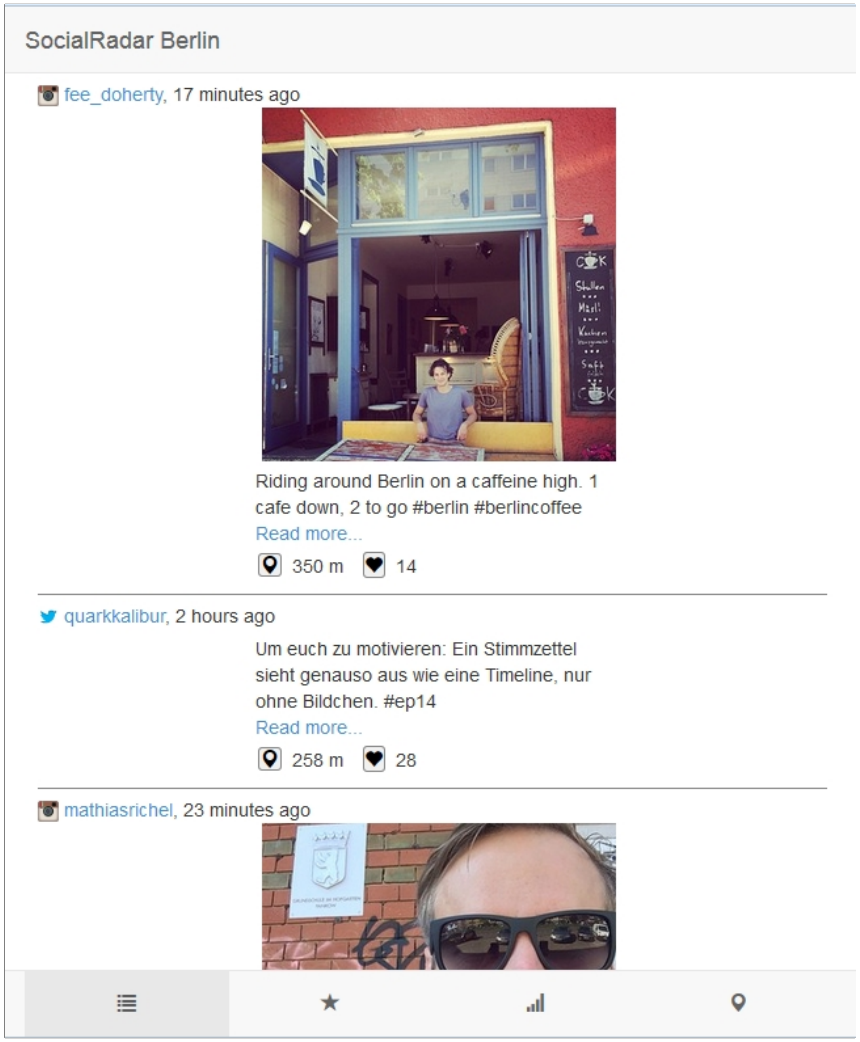
Combined text and geo query:

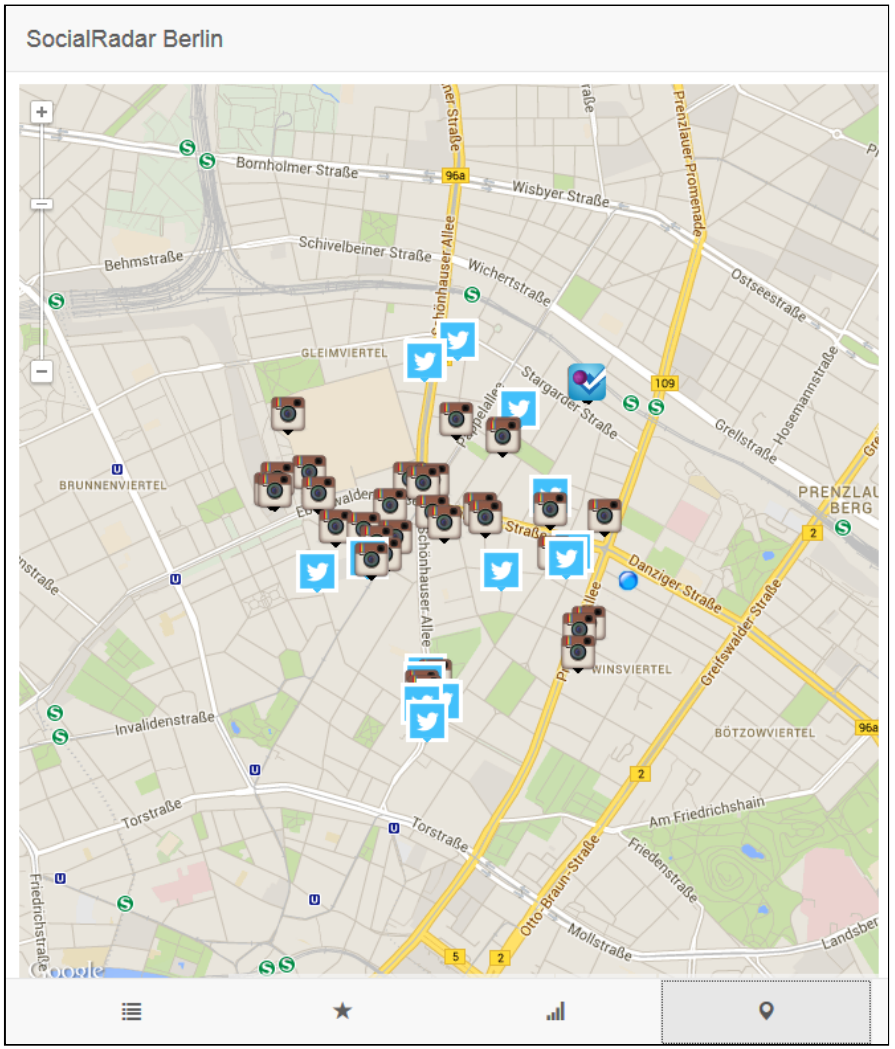
```
{
  - filtered: {
      - query: {
          - query_string: {
              query: "this AND that OR thus"
            }
        },
      - filter: {
          - geo_distance: {
              - coordinates: [
                  13.425812,
                  52.537903
                ],
                distance: "1000.0m",
                distance_type: "arc",
                optimize_bbox: "memory"
            }
        }
    }
}
```

Social Data

Demo Part I

Demo


SocialRadar Berlin
fee_doherty, 17 minutes ago
Riding around Berlin on a caffeine high. 1 cafe down, 2 to go #berlin #berlincoffee
Read more...
350 m
14
quarkkalibur, 2 hours ago
Um euch zu motivieren: Ein Stimmzettel sieht genauso aus wie eine Timeline, nur ohne Bildchen. #ep14
Read more...
258 m
28
mathiasrichel, 23 minutes ago


SocialRadar Berlin
Bornholmer Straße
Wisbyer Straße
Schivelbeiner Straße
Behmstraße
Wichertstraße
GLEIMVIERTEL
Stargarder Straße
Grellstraße
Hosemannstraße
BRUNNENVIERTEL
PRENZLAUER BERG
Danziger Straße
Greifswalder Straße
WINSVIERTEL
BÖTZOWVIERTEL
Invalidenstraße
Torstraße
Am Friedrichshain
Otto-Braun-Straße
Mollstraße
Google

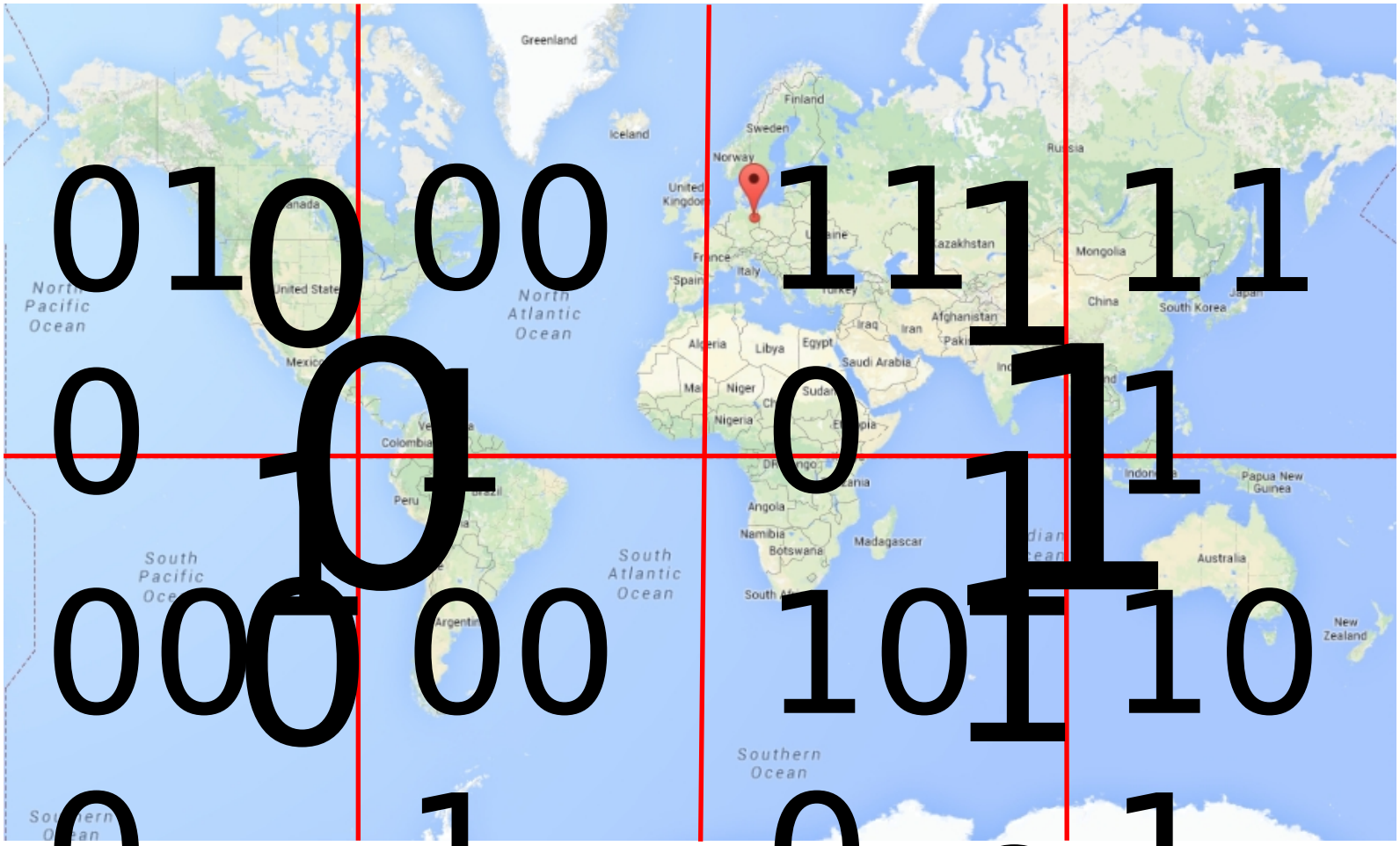
Part II

Geospatial analysis

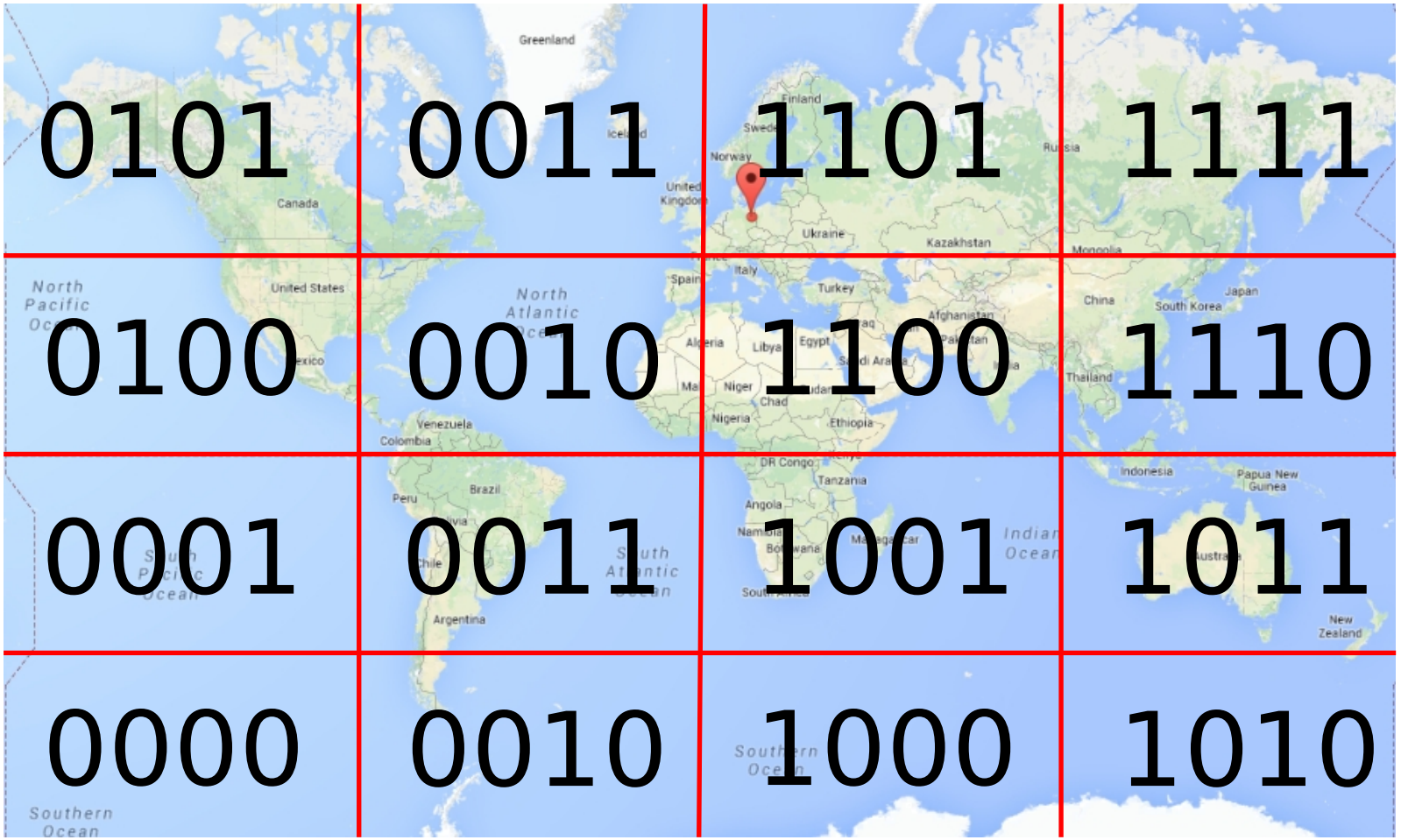
Geohashes

- Geocode system invented by Gustavo Niemeyer
- Subdivides space into buckets of arbitrary precision
- Each location has a unique hash
- Location can be derived from hash
- Public domain

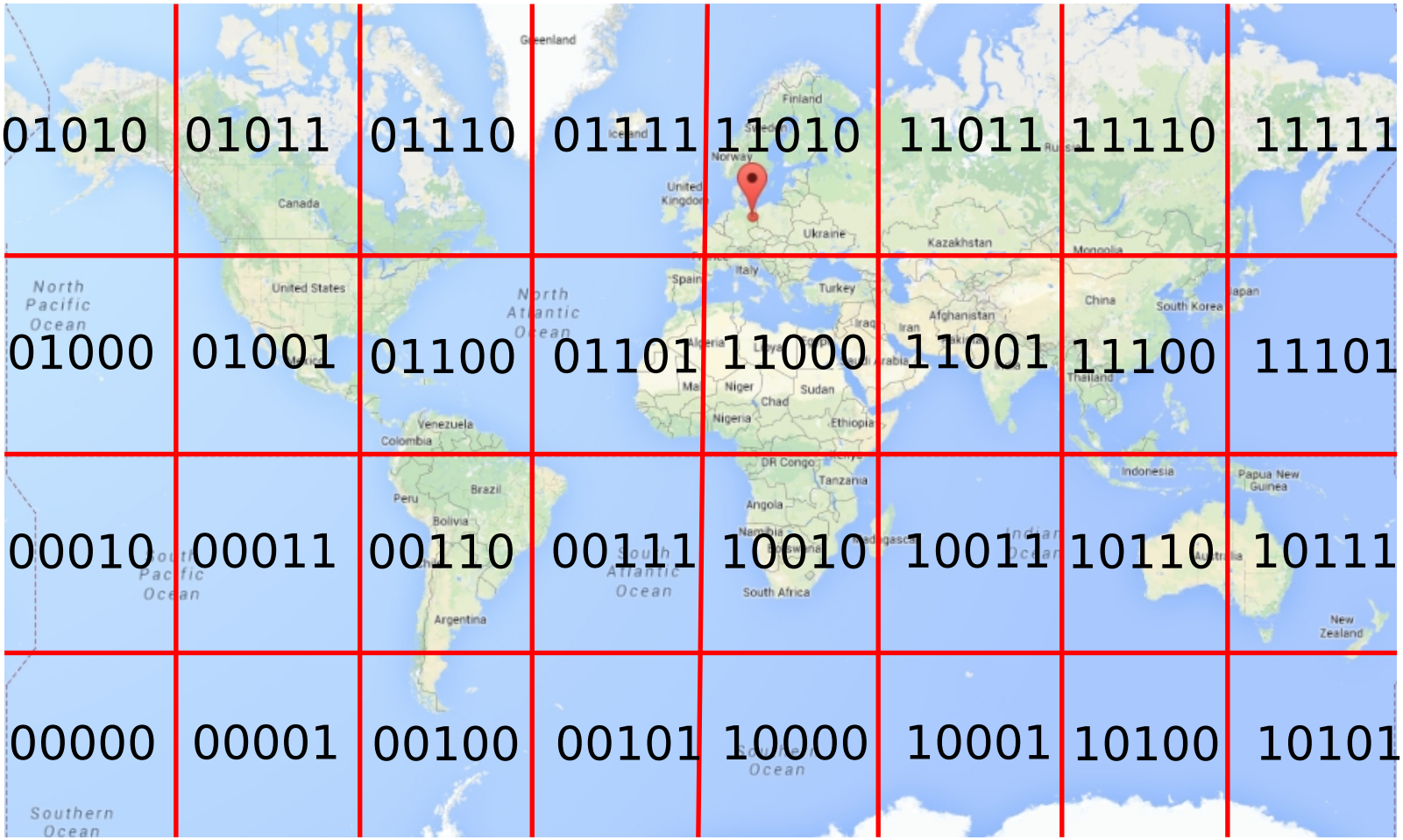
Geohashes


Greenland
Iceland
Finland
Sweden
Norway
Russia
United Kingdom
France
Spain
Italy
Kazakhstan
Mongolia
China
South Korea
Afghanistan
Iraq
Iran
Egypt
Libya
Algeria
Saudi Arabia
Niger
Nigeria
Sudan
Angola
Namibia
Botswana
Madagascar
Australia
Papua New Guinea
New Zealand
Peru
North Pacific Ocean
North Atlantic Ocean
South Pacific Ocean
South Atlantic Ocean
Southern Ocean
01000 11111
01010 10111
00000 10110
00100 10010

Geohashes


0101
0011
1101
1111
0100
0010
1100
1110
0001
0011
1001
1011
0000
0010
1000
1010
Greenland
Canada
United States
Venezuela
Colombia
Peru
Brazil
Argentina
Chile
Finland
Norway
Ukraine
Kazakhstan
Russia
Mongolia
Spain
Italy
Turkey
Egypt
Libya
Algeria
Niger
Chad
Nigeria
Ethiopia
DR Congo
Tanzania
Angola
China
Japan
South Korea
Thailand
Indonesia
Papua New Guinea
New Zealand
Southern Ocean

Geohashes

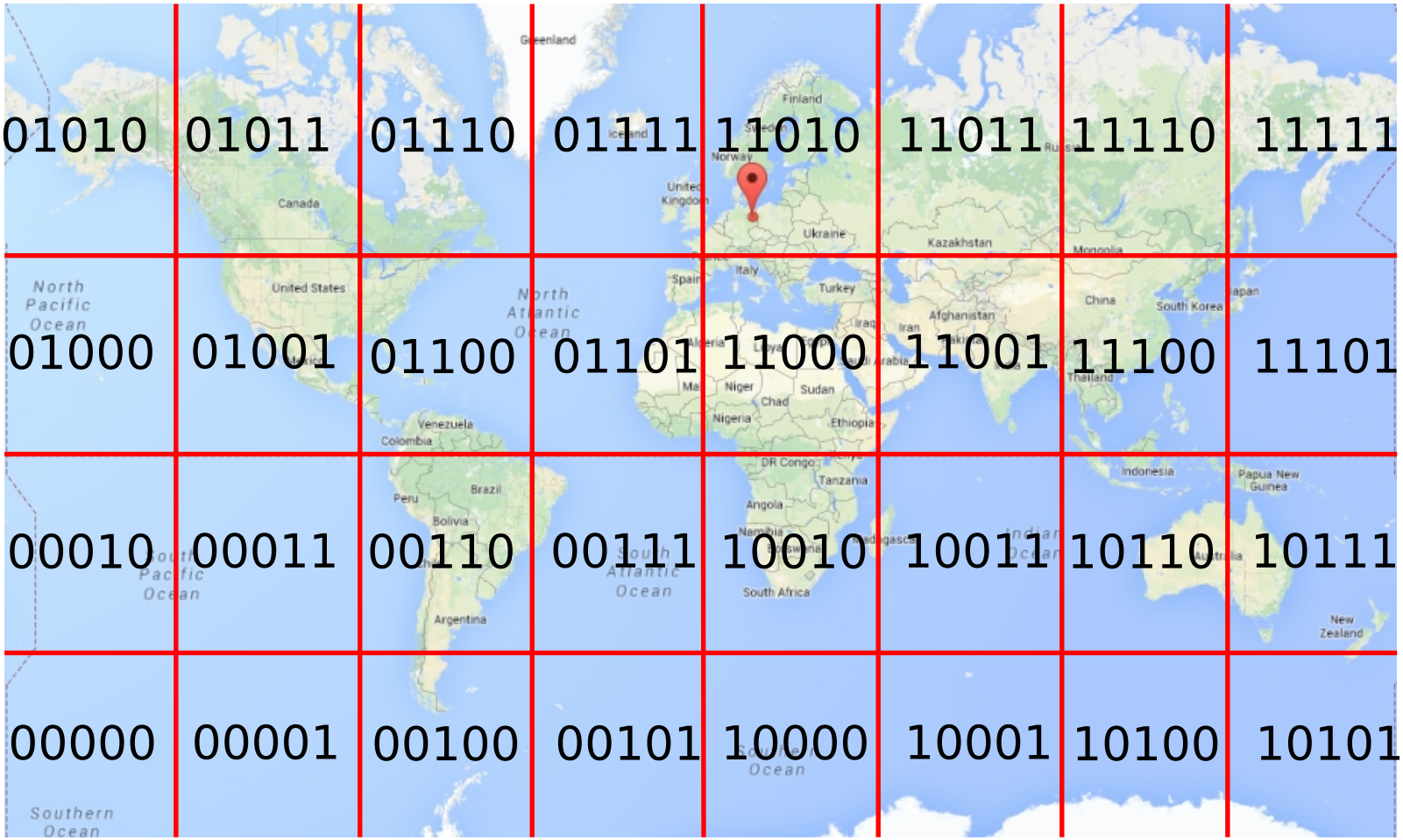

01010
01011
01110
01111
11010
11011
11110
11111
01000
01001
01100
01101
11000
11001
11100
11101
00010
00011
00110
00111
10010
10011
10110
10111
00000
00001
00100
00101
10000
10001
10100
10101
Greenland
Canada
United States
Finland
Norway
Ukraine
Kazakhstan
Mongolia
China
South Korea
Japan
Italy
Spain
Turkey
Afghanistan
Iran
Thailand
Indonesia
Papua New Guinea
Niger
Sudan
Chad
Nigeria
Ethiopia
DR Congo
Tanzania
Angola
South Africa
Venezuela
Colombia
Peru
Brazil
Bolivia
Argentina
New Zealand
North Pacific Ocean
North Atlantic Ocean
South Pacific Ocean
South Atlantic Ocean
Indian Ocean
Southern Ocean

Geohashes

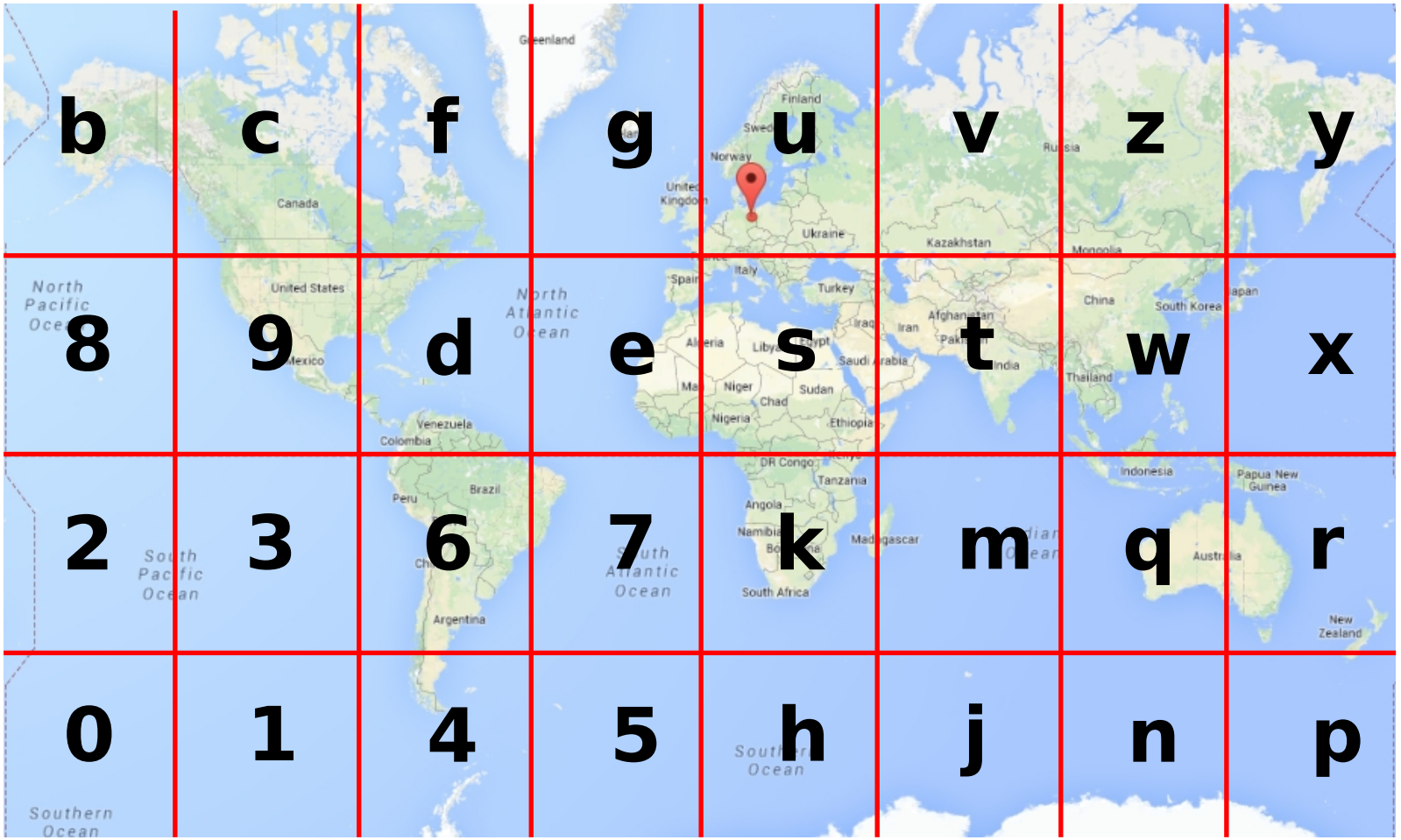
- We are in square 11010 (binary)
- Or in 26 (decimal)
- Or in u (base 32)

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Base 32	0	1	2	3	4	5	6	7	8	9	b	c	d	e	f	g
Decimal	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Base 32	h	j	k	m	n	p	q	r	s	t	u	v	w	x	y	z

Geohashes


01010
01011
01110
01111
11010
11011
11110
11111
01000
01001
01100
01101
11000
11001
11100
11101
00010
00011
00110
00111
10010
10011
10110
10111
00000
00001
00100
00101
10000
10001
10100
10101
Greenland
Canada
United States
Finland
Norway
United Kingdom
Ukraine
Kazakhstan
Mongolia
China
South Korea
Japan
Italy
Spain
Turkey
Iraq
Iran
Afghanistan
Thailand
Niger
Sudan
Chad
Nigeria
Ethiopia
Venezuela
Colombia
DR Congo
Tanzania
Angola
South Africa
Peru
Brazil
Bolivia
Argentina
Indonesia
Papua New Guinea
New Zealand
North Pacific Ocean
North Atlantic Ocean
South Pacific Ocean
South Atlantic Ocean
Indian Ocean
Southern Ocean

Geohashes


b
c
f
g
u
v
z
y
8
9
d
e
s
t
w
x
2
3
6
7
k
m
q
r
0
1
4
5
h
j
n
p
Greenland
Canada
United States
Mexico
Venezuela
Colombia
Peru
Brazil
Argentina
Finland
Norway
United Kingdom
Ukraine
Kazakhstan
Russia
Mongolia
Italy
Spain
Turkey
Iraq
Iran
Afghanistan
Pakistan
India
China
South Korea
Japan
Thailand
Indonesia
Papua New Guinea
Australia
New Zealand
Algeria
Libya
Egypt
Saudi Arabia
Niger
Chad
Sudan
Nigeria
Ethiopia
DR Congo
Tanzania
Angola
Namibia
Madagascar
South Africa
North Pacific Ocean
North Atlantic Ocean
South Pacific Ocean
South Atlantic Ocean
Southern Ocean

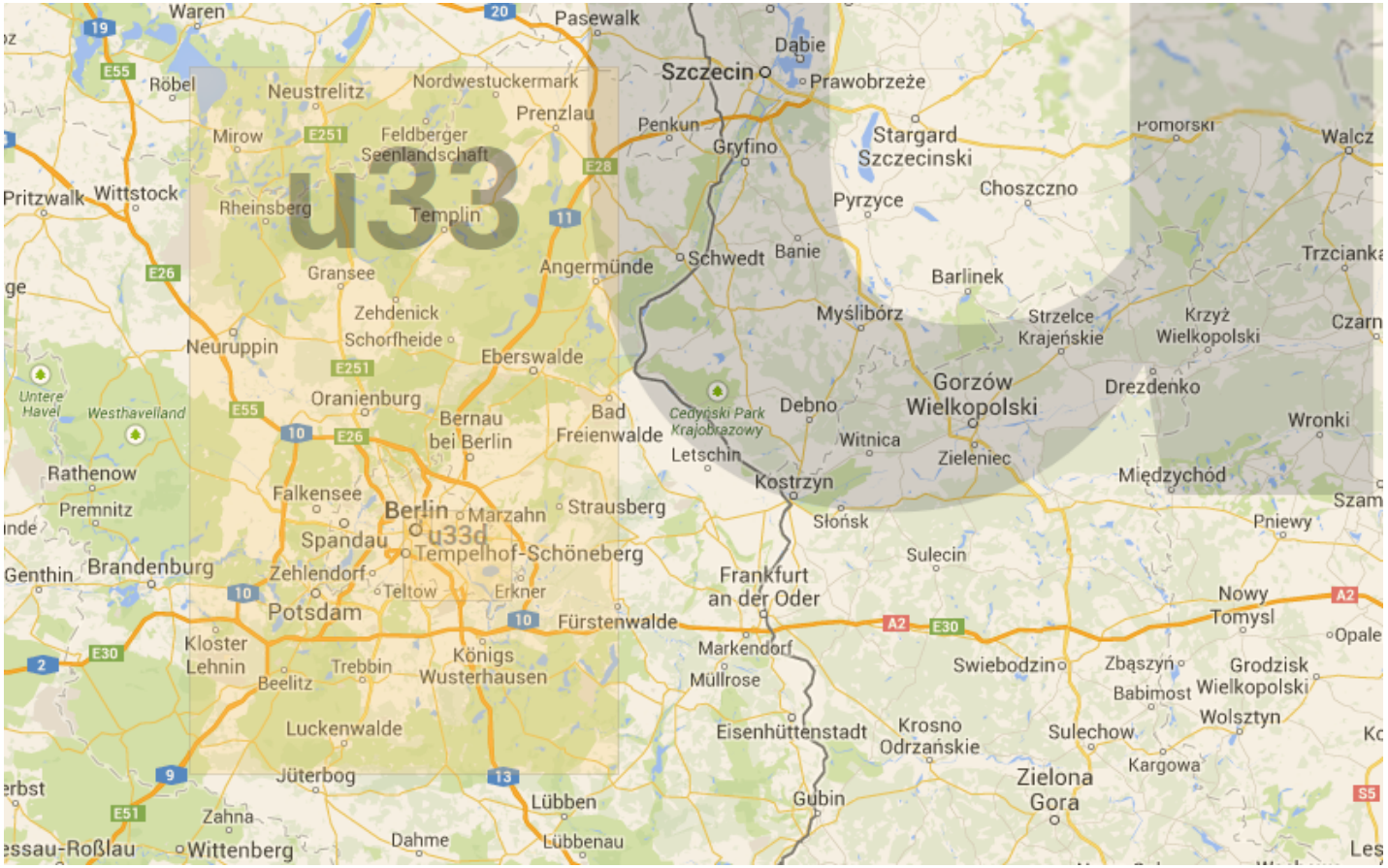
Geohashes

Sceenshots from http://geohash.gofreerange.com/

Geohashes

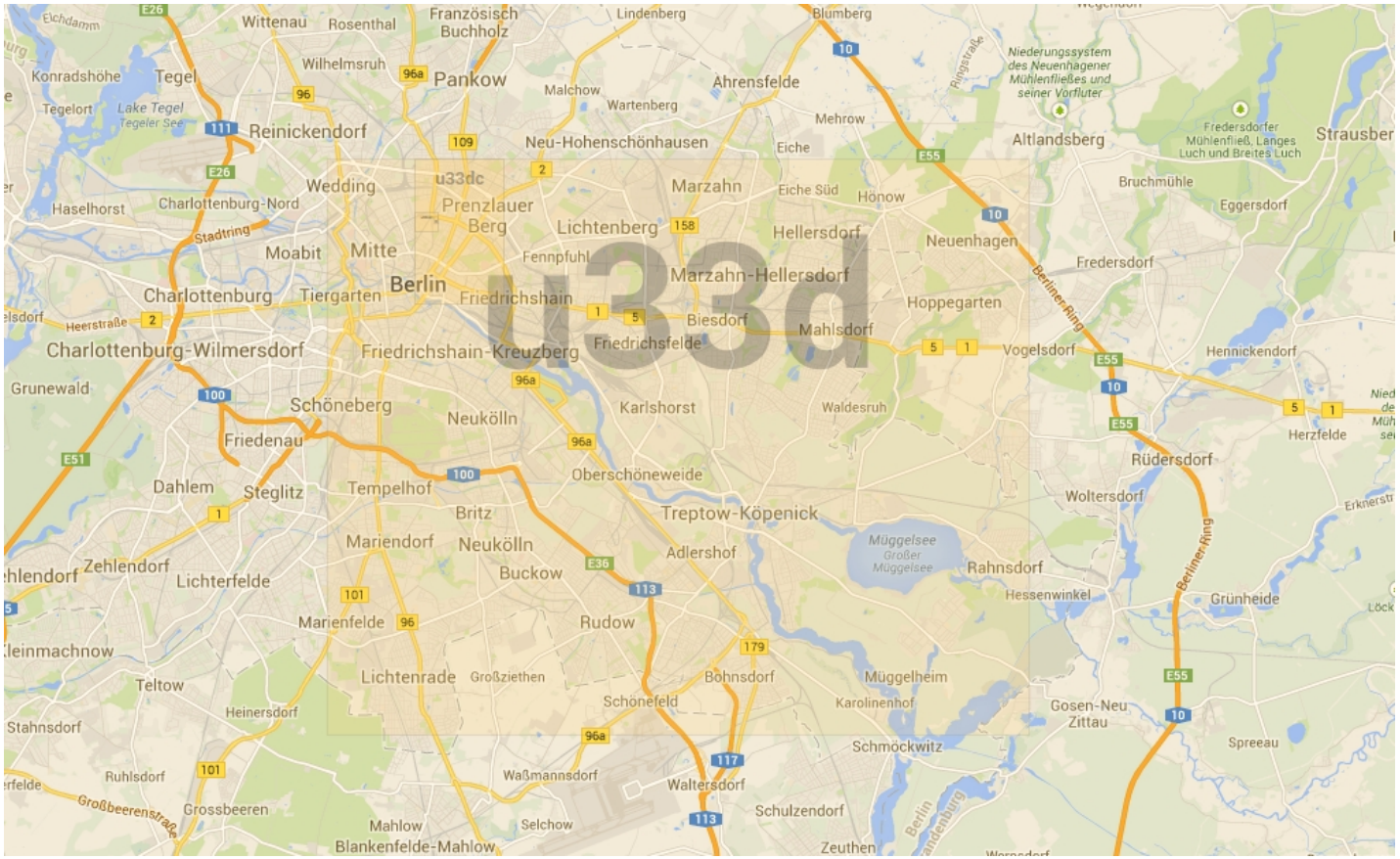
Sceenshots from http://geohash.gofreerange.com/

Geohashes



Sceenshots from http://geohash.gofreerange.com/

Geohashes



Sceenshots from http://geohash.gofreerange.com/

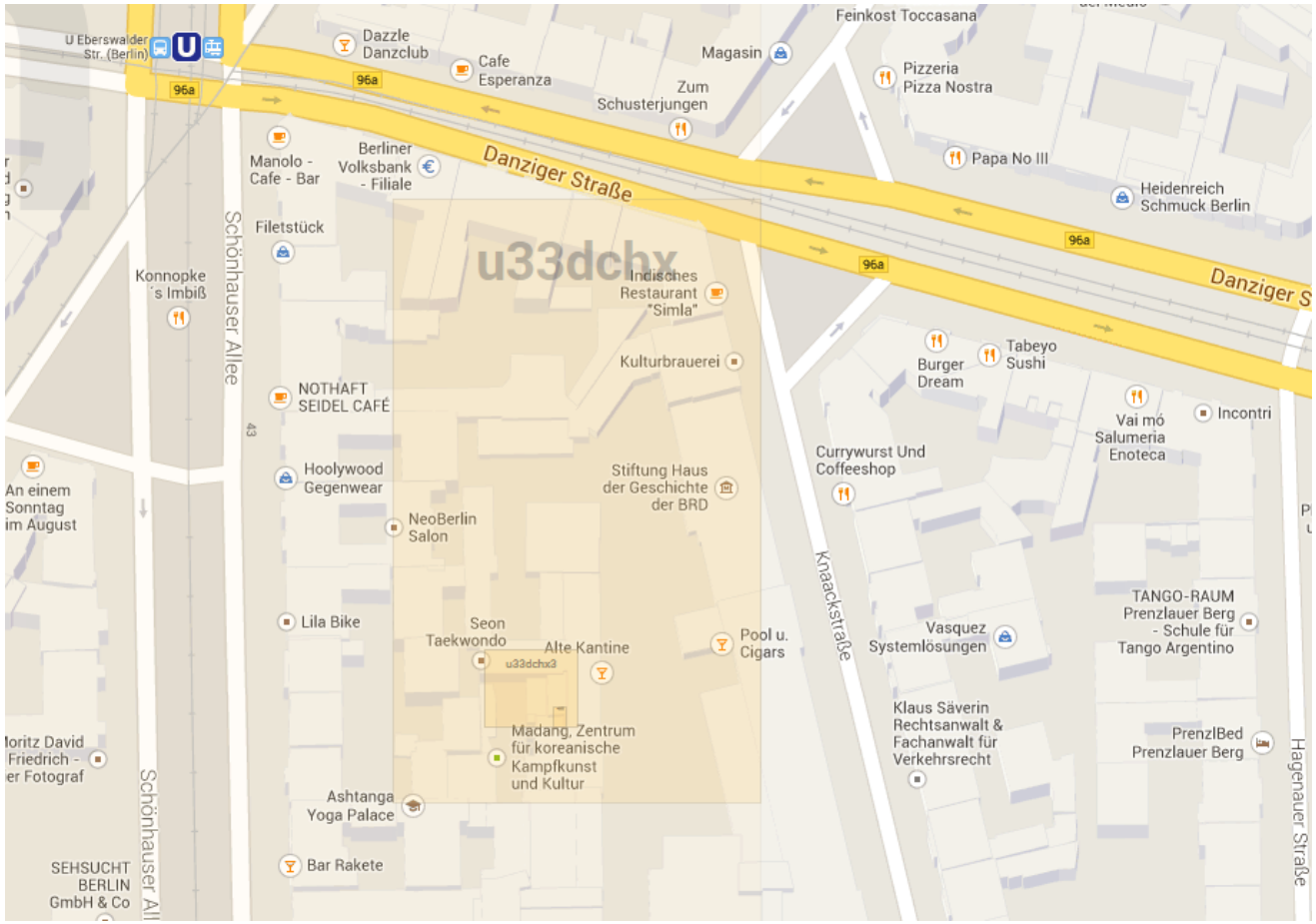
Geohashes

Sceenshots from http://geohash.gofreerange.com/

Geohashes

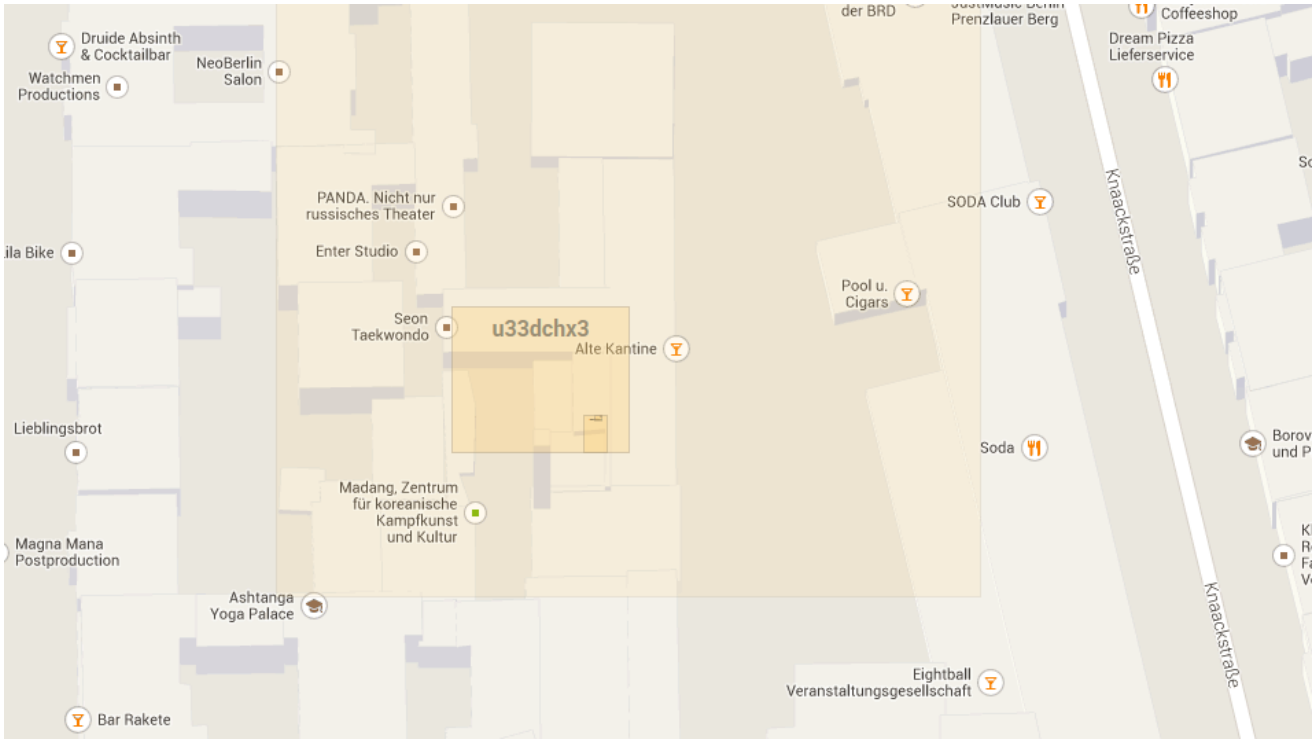
Sceenshots from http://geohash.gofreerange.com/

Geohashes



Sceenshots from http://geohash.gofreerange.com/

Geohashes



Sceenshots from http://geohash.gofreerange.com/

Geohashes

Now what is this good for?

- If we index posts with their Geohashes ...
- ... we can check which buckets have the most posts in them ...
- ... and possibly use this to identify „hotspots“ in the city

Geohashes

Adding geohashs to our mapping:

```
{
  - posts: {
      - properties: {
          - coordinates: {
                type: "geo_point"
            }
        }
    }
}
```

```
{
  - posts: {
      - properties: {
          - coordinates: {
                type: "geo_point",
                geohash: true,
                geohash_prefix: true,
                geohash_precision: 10
            }
        }
    }
}
```

Using aggregations on the geohash field:

```
SearchResponse response = ElasticsearchConnectionManager.getClient()
        .prepareSearch(ElasticsearchConnectionManager.getInstance().getPostIndexName())
        .setQuery(QueryBuilders.filteredQuery(qb, nrFilter))
        .addAggregation(terms("geohash").field("coordinates.geohash").include(".{7}").size(50))
        .execute()
        .actionGet();

Terms  terms = response.getAggregations().get("geohash");
Collection<Terms.Bucket> buckets = terms.getBuckets();
```

(Note: Newer ES-Geohash features available)

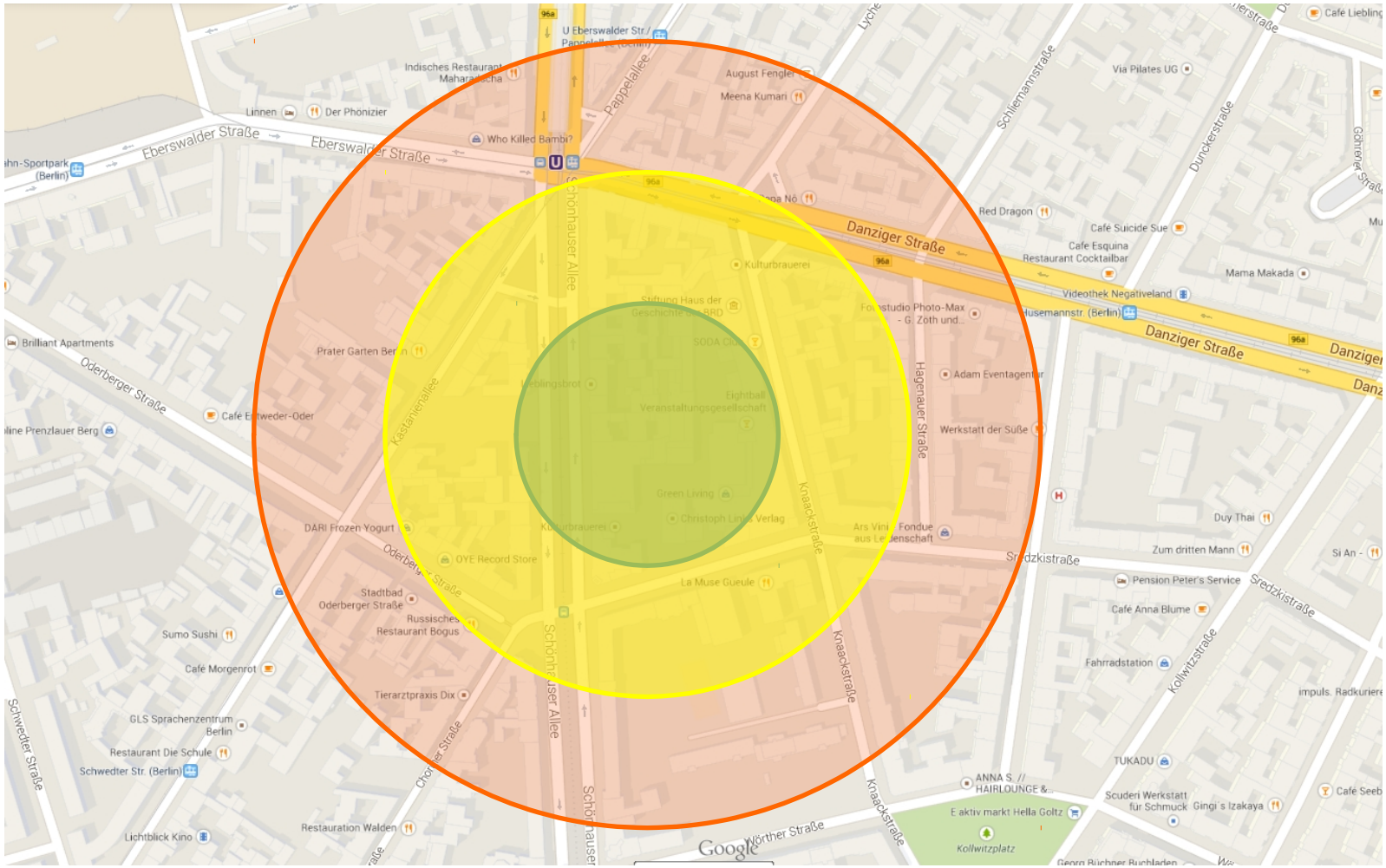
Geohashes

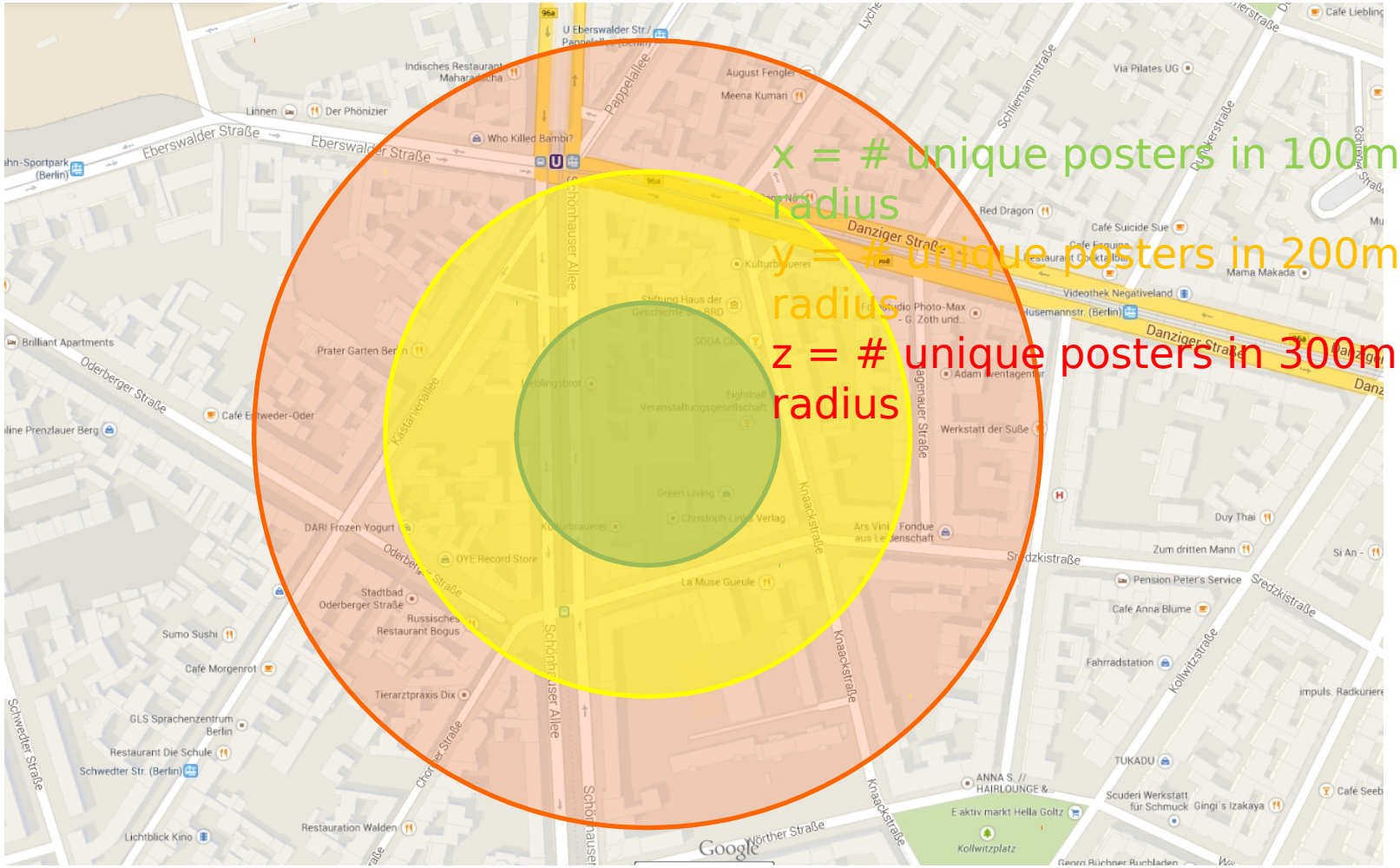
Are we done now? Not quite!

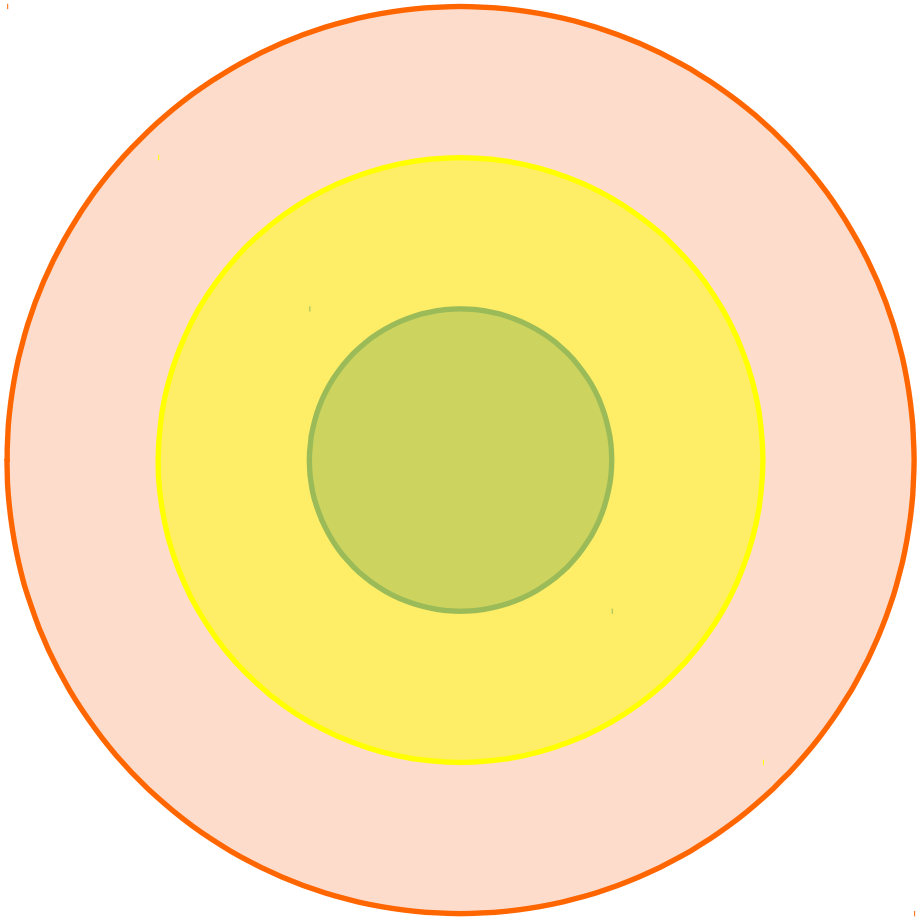
Issues:

- Monologues
- Some areas more active than others

 Hypotheses need to be verified


Eberswalder Straße
Danziger Straße
Schönhauser Allee
Pappelallee
Kastanienallee
Hagenauer Straße
Knaackstraße
Schliemannstraße
Dunckerstraße
Oderberger Straße
Schwedter Straße
Kollwitzstraße
Sredzkistraße
Kollwitzplatz
Google


x = # unique posters in 100m radius
y = # unique posters in 200m radius
z = # unique posters in 300m radius
Eberswalder Straße
Danziger Straße
Schönhauser Allee
Pappelallee
Knaackstraße
Kastanienallee
Oderberger Straße
Schliemannstraße
Kollwitzstraße
Sredzkistraße
Google



x = # unique posters in 100m radius

y = # unique posters in 200m radius

z = # unique posters in 300m rac

$$relativeScore = \max(\frac{x}{y}, \frac{y}{z})$$

$$absoluteScore = \log_{10} x + \log_{10} y$$

Social Data

Demo Part II

Demo

SocialRadar Berlin

Hot right now!

Rathaus Schöneberg: 4 posts from around here.
Read more...
7681 m

Hot right now!

East Side Gallery: 3 posts from around here.
Read more...
3724 m

Hot right now!

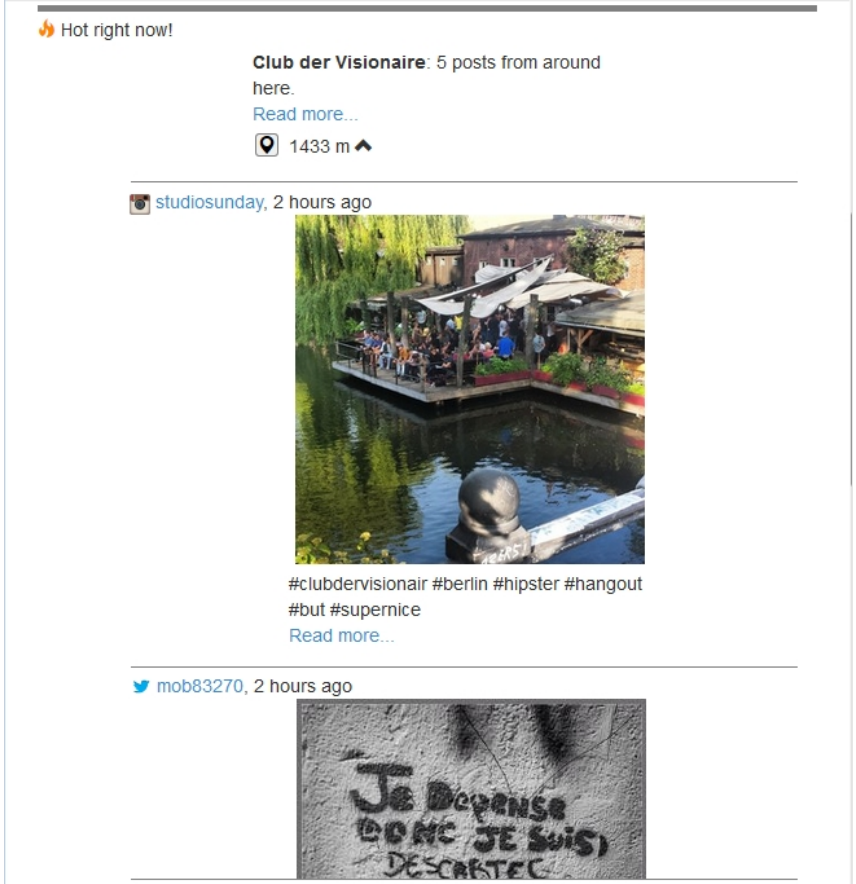
Somewhere in Berlin: 5 posts from around here.
Read more...
786 m

Hot right now!

Somewhere in Berlin: 4 posts from around here.
Read more...
794 m

Hot right now!

Club der Visionaire: 5 posts from around here.
Read more...
1433 m



Open issues

- Center of geohash not always center of event
- Some locations trending every day
  - Train station, airport, Brandenburg Gate, East Side Gallery
- Instagram coordinates not always where photo was taken

 More sophisticated analysis of hotspots needed (last year’s talk)

From last year‘s talk

Tweet cluster:

- Suspicious package in #GrandCentral #NYC #bomb threat possibility not sure?? http://t.co/VwU7SP3X
- Suspicious package found in Grand Central Station... the 456 train..the trains are closed !! [pic]: http://t.co/9YPki4k2
- Something happened in the #456 #trainstation in #GrandCentral #NYC http://t.co/GGKvQura
- Accident on the #456train in #midtown #NYC http://t.co/fj2mJJmf

Textual features

Feature Group	#	Brief Description
Common Theme	1	Calculates the n-gram overlap between different tweets in the cluster.
Near Duplicates	1	Indicating how many tweets in the cluster are near-duplicates of other tweets in the cluster.
Positive Sentiment	3	Indicating positive sentiment in the cluster.
Negative Sentiment	3	Indicating negative sentiment in the cluster.
Overall Sentiment	2	Indicating the overall sentiment tendency of the cluster.
Sentiment Strength	3	Indicating the sentiment strength of the cluster.
Subjectivity	2	Indicating whether tweeters make subjective reports rather than just sharing information, e.g., links to newspaper articles.
Present Tense	2	Indicating whether tweeters talk about the here & now rather than making general statements.
# Ratio	1	Number of hashtags relative to the number of posts in the cluster.
@ Ratio	1	Number of @s relative to the number of posts in the cluster.
RT Ratio	1	Fraction of tweets in the cluster that are retweets.
Semantic Category	13	Indicating whether the cluster belongs to certain event categories, e.g., "sport event" or "fire".

Other features

Feature Group	#	Brief Description
Link ratio	1	Indicating the number of posts that contain links.
Foursquare ratio	1	Fraction of tweets originating from Foursquare.
Tweet count	1	Score based on how many tweets there are in the cluster.
Poster count	2	Score based on how many different users posted the tweets in the cluster.
Unique coordinates	2	Score based on how many unique locations the posts are from.
Special location	1	Fraction of tweets that are from a certain known "bad" location, e.g., airports or train stations.

Can you make it better?

https://github.com/txtData/socialradar

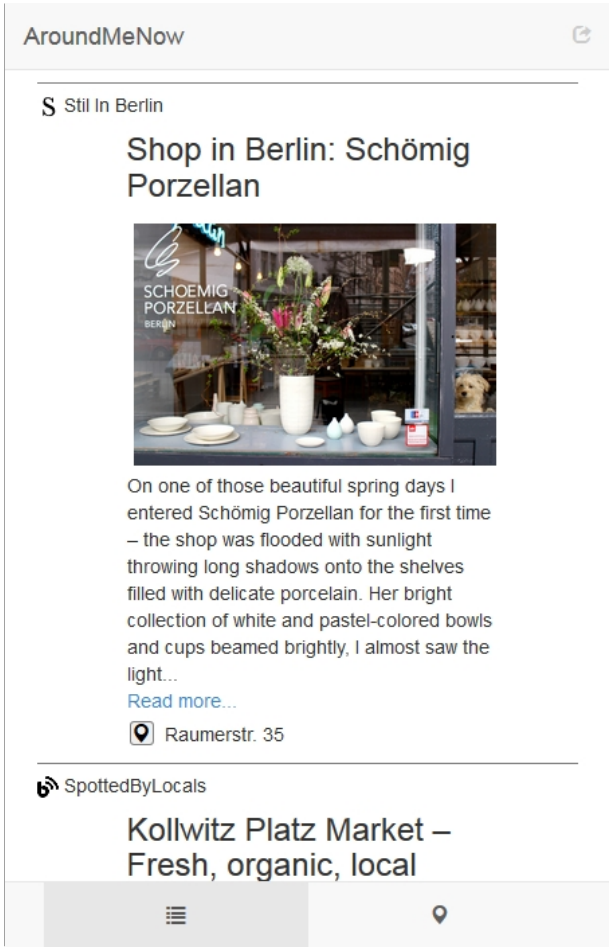
Running the code

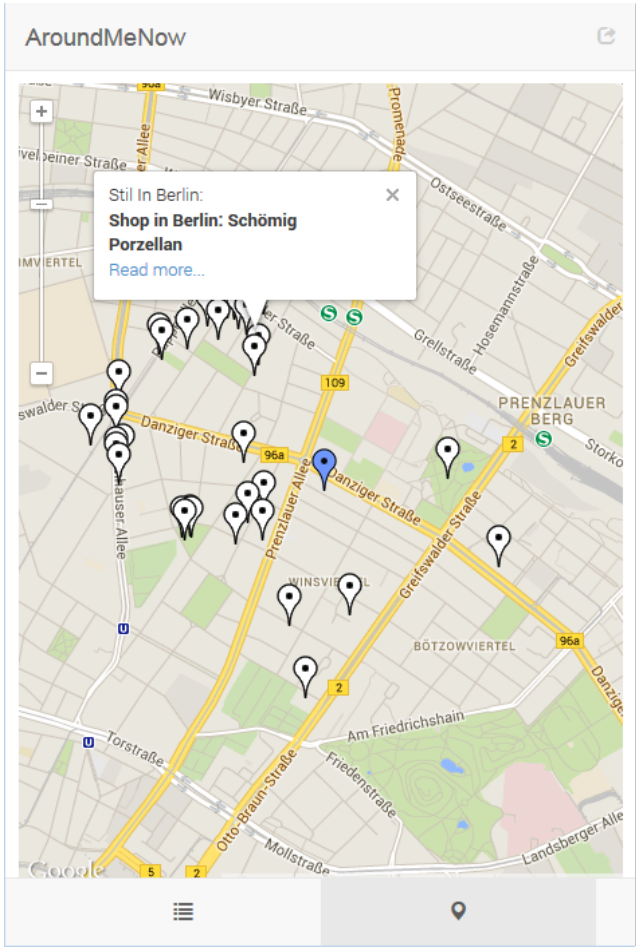
- Download and install Elasticsearch
  - http://www.elasticsearch.org/
- Download Java code from Github
  - https://github.com/txtData/socialradar
- **Get your API tokens!**
  - https://dev.twitter.com/
  - http://instagram.com/developer/
  - https://developer.foursquare.com/
- Compile Java code
- Run ElasticsearchConnectionManager (sets up mappings)
- Run fetchers:
  - InstagramFetcher
  - TwitterStreamingFetcher
  - Foursquare
- Wait a minute or two (to fill the database)
- Run HotspotFinder

Excursus: Putting blogs on a map

- LODE: OSM-based, recognizes nearly all forms of location names in text:
  - Addresses
  - Streets
  - Neighborhoods
  - Points of Interest (POIs)
  - Cities, states, countries etc.
  - Informal location names
- TEGO: Makes articles and blogs browsable by location
- Based on ES, similar to SocialRadar

Excursus: Putting blogs on a map





Any questions? Feel free to ask...

Thank YOU

(Or write me at mkaisser@txtdata.net)