D2.2 Content Store Evaluation

Spatial Accuracy of User Generated Data

Deliverable lead contractor: WIGeoGIS

Main contributors:
Shoaib Burq shoaib@zedat.fu-berlin.de
Georg Magenschab gm@wigeogis.com
Dimitris Skoutas dskoutas@imis.athena-innovation.gr
George Lamprianidis glampr@imis.athena-innovation.gr
George Papatheodorou papatheodorou@imis.athena-innovation.gr

Abstract

This deliverable documents the evaluation of the GeoStream Content Store to account for the spatial uncertainty in user-generated content streams.

Research Center “ATHENA”, Greece
Freie Universität Berlin, Germany
Fraunhofer, Germany
WIGeoGIS, Austria
Michael Mueller Verlag, Germany
TALENT, Greece

Defining Spatial Accuracy

In the context of GeoStream, we looked at two sources of uncertainty, spatial and non-spatial:

- Non-spatial uncertainty arises from the fuzzy matching during the schema fusion process. The first step in being able to use multiple data sources is to match the disparate schemas to GeoStream's internal reference schema. This fusion is done via matching rules that employ string similarity and semantic similarity to map the categories of a provider's point of interest (POI) to the GeoStream categories. This process is fuzzy and stochastic and provides a broad range of match scores. This process was described in Deliverable 2.1: UGCS Content Store.
- Spatial uncertainty comes from errors or inaccuracies in location information collected during the data retrieval process.

In this report, we focus on the latter, namely the Spatial Uncertainty.

Error Assessment Process

In order to quantify the error, we compare GeoStream data to Yellow Pages data for the city of Vienna. Yellow Pages is the most authoritative data source for this purpose, and is the one currently used by WIGeoGIS in its products. However, it is a commercial dataset that has to be purchased and is not publicly available for use. Hence, the goal of this study was to evaluate the data coverage of the crowdsourced data collected from Web sources w.r.t. this authoritative, commercial dataset.

Formally we define our error in statistical terms: i.e. the error in the coordinates of a POI from a provider is the deviation of the location's coordinates from the Yellow Pages (or true) location. We derive the overall error for a provider by taking the mean of the POIs deviation. This can be represented by the following formula:

Given that *Provider P* has the coordinates: X_p and *Yellow Page* has the coordinates: X_y the error for a given POI is $\varepsilon_i = X_p - X_y$. The subtraction here is performed in a cartesian coordinate system. Thus the difference is in a unit of distance. From this we can derive the *mean error* as: $\mu_e = \sum_{i=1}^{n} \left(X_p - X_y \right)^2 / n$

Crowdsourced POI Datasets

For our evaluation, we study the spatial uncertainty of locations of POIs in the GeoStream content store, which have been collected from Web sources. These sources include both POIs retrieved from commercial providers' APIs, such as Foursquare and Google Places, as well as from open data sources such as OpenStreetMap and DBpedia. Table 1 lists the data providers selected for the analysis:

Data sources	URL
DBpedia	http://dbpedia.org
OpenStreetMap	http://www.openstreetmap.org
Wikimapia	http://wikimapia.org
Google places	https://developers.google.com/places
FourSquare	https://foursquare.com
Eventful	http://eventful.com

Table 1. Data Sources

Yellow Pages Directory

In the conducted evaluation, the POI data from the aforementioned sources are contrasted to those found in the Vienna Yellow Pages dataset. For this purpose, the Yellow Pages dataset was treated as yet another provider and the contained POIs

were added in the content store. The client for importing the data made use of the data dictionary provided by WIGeoGIS that accompanied the data to map fields from the Yellow Pages to internal GeoStream attributes. The code for this client is listed in Appendix 1. The attribute mapping is shown below in a code snippet from the client:

```
mapping = {
  provider_id:  "SID",                # HEROLD-Number
  external_url: "HTTP",               # Homepage of Company
  name:         "FIRMA",              # Name of Company
  description:  "OENACE_BEZ",         # Classification of the business activity
  phone:        "TELEFON",            # Phone Number of Company
  address:      ["STRASSE", "HNR"],
  postal_code:  "PLZ",
  city:         "ORT",
  state:        "BL"
}
```

GeoStream Content: Venues

The GeoStream schema is made up of a large number of tables and associations. A subset of this is shown in Figure 1. Note that the Area table has a central position since all data collection begins by specifying an area of interest (e.g. a city). The table Venue is also a central table where (most of) the information about collected POIs is stored.

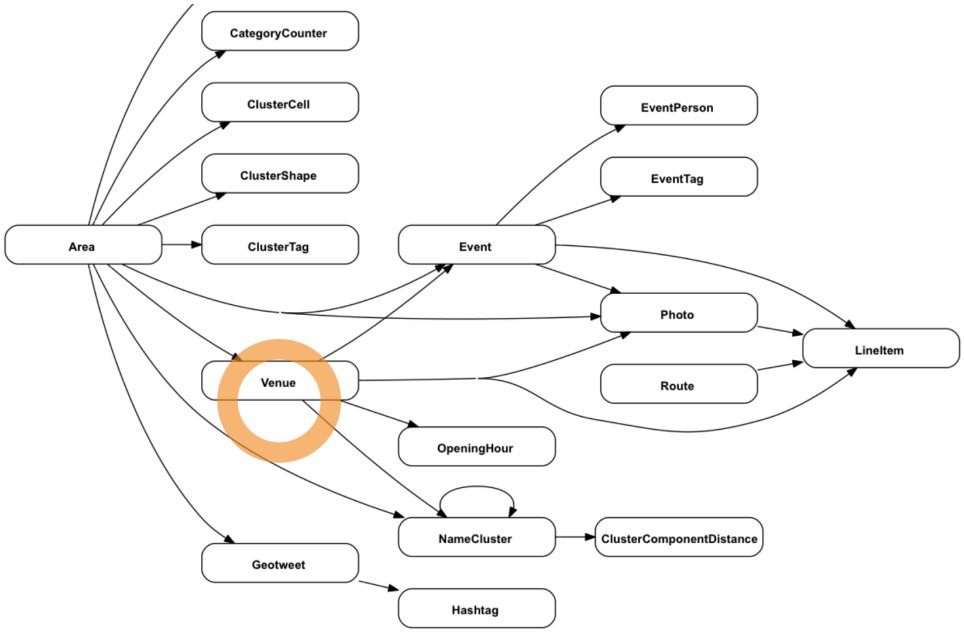

Figure 1. Subset of the GeoStream Schema focusing on the Venues table.

In Figure 2 we see a detailed schema for the Venue tables. This includes coordinates, references to the provider's URL as well as the names and addresses of venues.

Figure 2. Venues Schema.

Uncertainty Calculation Process

The Vienna Yellow Pages dataset contains 65,499 POIs. These were compared to POIs collected from the GeoStream providers. The process of finding spatial uncertainty is summarized in Algorithm 1. It is run for each provider and involves first filtering Provider POIs by proximity to the Yellow Pages POIs. This is followed by filtering based on similarity POI names. Finally, the spatial distance of a pair of POIs from the Yellow Pages and the GeoStream dataset is calculated.

The name similarity was calculated using the Jaro Winkler algorithm for string similarity. This method was chosen because it was developed specifically to match short strings, such as names [1].

```
Input: Provider Name being assessed for spatial accuracy

FOREACH POI A from Yellow Pages
  FIND POIs from Provider B WITHIN a distance of 200m
  FOREACH POI B in found POIs from B
    calculate the Jaro-Winkler distance between the name of Venue A and Venue B
    IF Jaro-Winkler distance > 0.9
      calculate the difference in the coordinates of Venue A and B
    ENDIF
  ENDFOREACH
ENDFOREACH
```

Algorithm 1. Calculating spatial uncertainty.

Evaluation Results

Figure 3 and Table 2 show the results of matches for four of the six initially selected providers (see Table 1). This is because there were too few matches with the other two providers. The POI data in the GeoStream content store used for this evaluation was obtained by harvesting user-generated content in the area of Vienna over a period of a few days. It is expected that not every business listed in the Yellow Pages can be found in the GeoStream dataset since Yellow Pages is a much more comprehensive business directory, while, on the other hand, crowdsourced POIs have a much wider scope. Nevertheless, we were able to match a significant number of POIs, as shown in the table below.

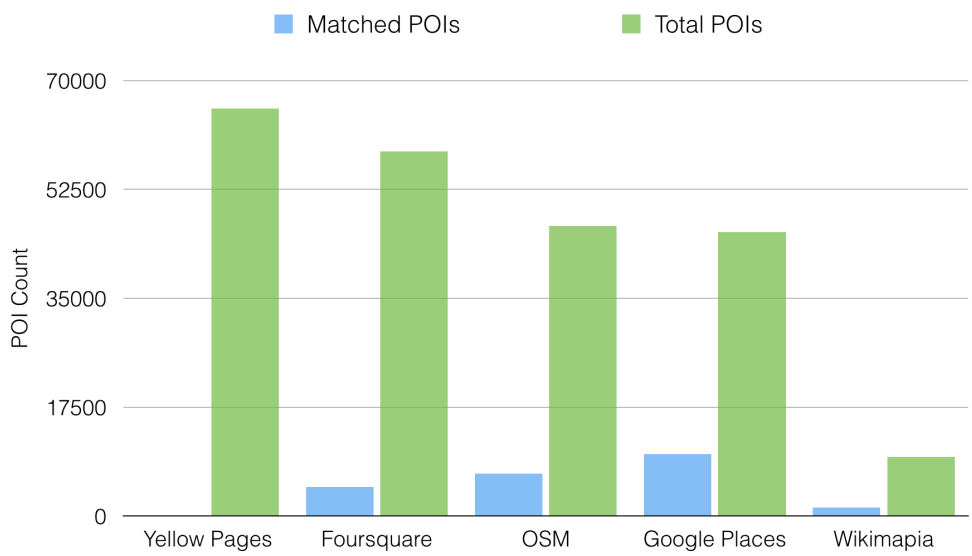

Figure 3. Summary of POIs matched.

	Foursquare	OpenSteetMap	Google Places	Wikimapia
Total POI's	58646	46645	45674	9519
Matched POI's	4630	6854	9953	1361
Percent matched	8%	15%	22%	14%

Table 2. Summary of POIs matched.

The main concern for this evaluation was the distribution of error distance (measured in meters) for each provider. This is shown in Figure 4. The plots show the sample size (N) and bandwidth parameter for the kernel. Note that the kernel bandwidth is chosen for optimal smoothing of the density curve based on the total sample size. If the bandwidth is too high the curve is oversmoothed and if its too look the curve is undersmoothed. The density plots show the spatial error on the x-axis and allow for a quick assessment of the distribution of the error. It is clear from the plots for each provider that the mean error is relatively low and has a long-tail.

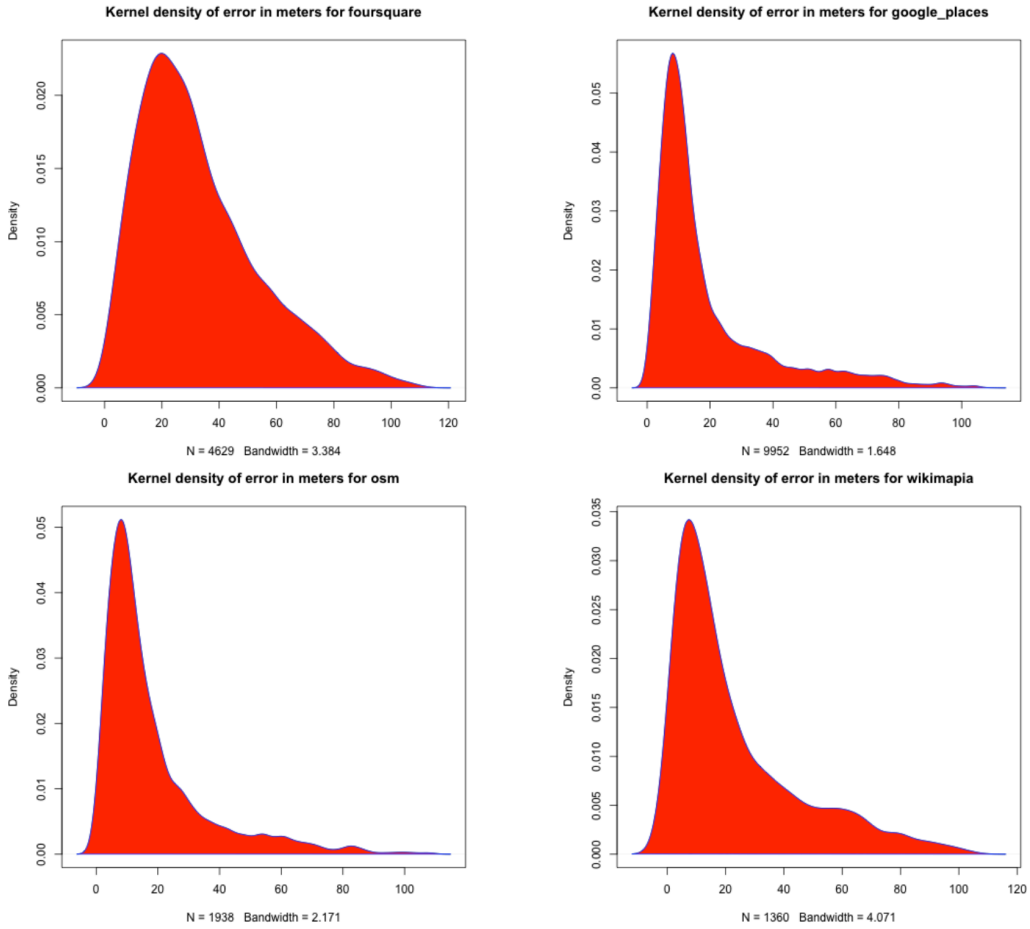

Figure 4. Kernel density plot of error distances (X-axis shows error in meters).

The summary statistics for these distributions are shown in Table 3. As can be seen, the largest mean is 33 meters, for the case of Foursquare.

Provider	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
foursquare	0.4671	17.2600	28.4700	32.9700	44.5100	110.6000
osm	0.2436	7.1030	11.9100	18.0500	21.7900	108.4000
google_places	0.1276	7.3020	11.8300	19.0500	22.7700	109.1000
wikimapia	0.1854	7.2180	14.9000	23.3200	32.8800	103.9000

Table 3. Summary statistics of spatial error in meters.

Figure 5 shows a box-plot of the spatial distance error grouped by categories. The plot reveals some important features of the dataset. The vertical line and the number on each of the box plots represents the median error. As can be seen, the values are relatively low, with the highest being for the “Athletics Sports” category. The rectangular box of the box plot represents the upper and lower bounds of the standard deviation, i.e. the upper and lower quantiles. The blue/gray cloud of points show the distribution of sample points. We can see that the variance is small for some values. The small variance also corresponds to the low number of sample points, and in some cases high error values. The red points represent the outliers.

Another important conclusion we can draw from this box plot is that there is no significant difference in the error from commercial sources (Foursquare and Google Places) and non-commercial, user-generated sources (OpenStreetMap and Wikimapia).

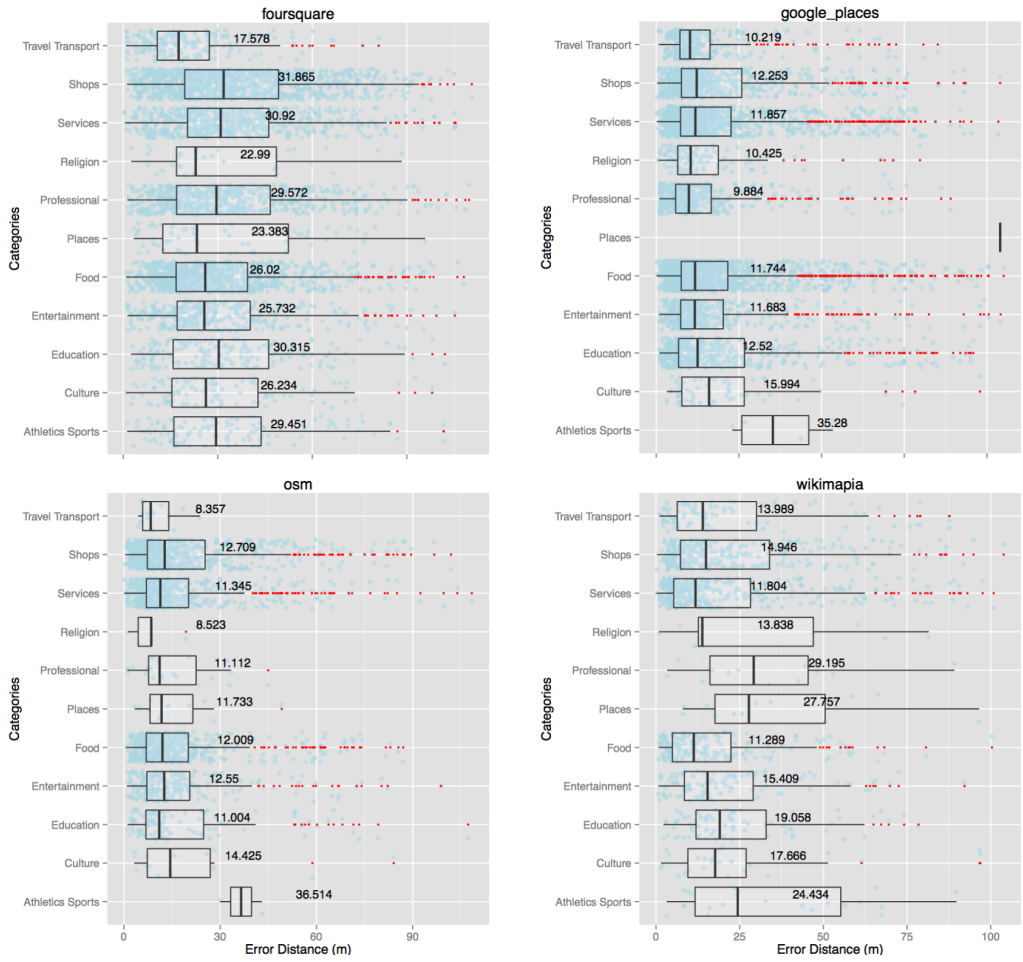

Figure 5. Box-plot: Distribution of spatial error by top level GeoStream categories.

Lastly, in order to ensure that there was no systematic correlation between the spatial error and the name matching, we calculated the correlation coefficients for spatial error and name matching distance.

The result of this measurement is interpreted as follows: (a) if the correlation coefficient is close to 1, it indicates that the variables are positively linearly related and the scatter plot falls almost along a straight line with positive slope; for –1, it indicates that the variables are negatively linearly related and the scatter plot almost falls along a straight line with negative slope, and (c) for 0, it indicates a weak linear relationship between the variables. In Table 4, we can see the results are very close to 0 and the scatter plots show a random distribution of points (Figure 6).

Provider	Correlation Coefficient
foursquare	–0.0441
osm	–0.0898
google places	–0.0317
wikimapia	–0.0775

Table 4. Correlation coefficient for spatial error and name match.

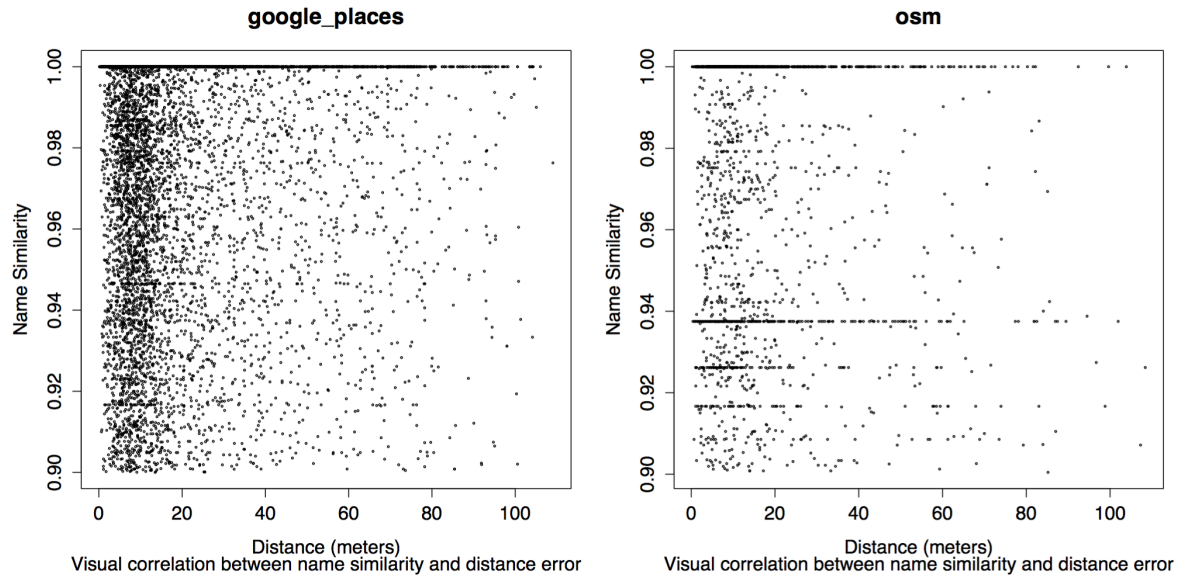

Figure 6. Scatter plots of error and name similarity.

In the scatter plot displayed in Figure 6, for some providers we see horizontal artifacts. These correlate to the string matching algorithm giving similar values for certain POI names. The reason behind this is the fact that the name of companies in Yellow Pages are often suffixed by the type of legal entity (e.g. GmbH or Ug). Sometimes in GeoStream these names lack the suffix. Hence the match value for such strings tend to cluster together.

Conclusion

According to the results of the experimental analysis conducted as described above,

the following main conclusions can be drawn as a summary:

- The mean error distance of the crowdsourced POIs w.r.t. the ones found in the Yellow Pages dataset is less than 35 meters. This observation shows that in this experiment no significant differences in spatial quality of the crowedsourced data w.r.t. the commercial data arise.
- The top level categories have a larger match sample and generally exhibit greater accuracy. However, for some of the categories the sample size was too small for prediction and a lower accuracy was observed. Although there is overlap between the Yellow Pages data and GeoStream data, Yellow Pages data has a different market focus. For example it includes service providers for both the Business-to-Businees (B2B) and Business-to-Consumer (B2C) markets. GeoStream data sources are much more consumer oriented.

From the point of view of spatial accuracy, there is little difference between crowdsourced and commercial data. There is also little difference in coverage (completness) between the commercial and non-commercial sources.

References

[1] William W. Cohen, Pradeep D. Ravikumar, Stephen E. Fienberg: A Comparison of String Distance Metrics for Name-Matching Tasks. IIWeb 2003: 73–78

Appendices

1. Yellow Pages Client

```
class Client::YellowPages
  include Client

  attr_accessor :input_file

  def initialize(input_file = nil)
  end

  def retrieve_data_within(area, date = nil)
    retrieve(area)
  end

  def retrieve!(bbox)
    mapping = {
      provider_id:  "SID",
      external_url: "HTTP",
      name:         "FIRMA",
      description:  "OENACE_BEZ",
      phone:        "TELEFON",
      address:      ["STRASSE", "HNR"],
      postal_code:  "PLZ",
      city:         "ORT",
      state:        "BL"
    }

    attributes = {
      area_id:  area.id,
      provider: "yellow_pages",
      country:  area.country
    }

    RGeo::Shapefile::Reader.open(input_file) do |file|
      file.each do |record|
```

```
          attrs = {}
          mapping.each_pair do |k, v|
            attrs[k] = Array.wrap(v).map { |vv| record.attributes[vv] } .join(" ")
          end
          attrs.merge!(attributes)
          attrs.merge!({coordinates: record.geometry})
          venue = Venue.unscoped.
                  where(provider: "yellow_pages", provider_id: attrs[:provider_id]).
                  first_or_initialize
          venue.from_herold(attrs, area)
        end
      end
    end

end
```

2. Code for Spatial Uncertainty Statistics in R

```
library(plyr)
library(ggplot2)

providers     = c('foursquare', 'osm', 'google_places', 'wikimapia', 'dbpedia', 'ev
root_dir      = '~/Documents/03_current_projects/geostream_rails/results/'
footer_size   = 9

for(i in 1:length(providers)) {
  ifile = paste(root_dir,'spatial_match_yellow_pages_',providers[i],'_0.001_1000000
  nrows <- sapply( ifile, function(f) nrow(read.csv(f)) )
  n = sum(nrows) - footer_size
  data = read.csv(ifile, nrows=n)
  d <- density(data$distance)
  plot(d, main=paste("Kernel density of error in meters for", providers[i]))
  polygon(d, col="red", border="blue")

  # only use level 1 categories
  filtered_data = subset(data,
        gs_cat_1 == "Professional" | gs_cat_1 == "Entertainment" |
        gs_cat_1 == "Culture" | gs_cat_1 == "Education" | gs_cat_1 == "Food" |
        gs_cat_1 == "Places" | gs_cat_1 == "Shops" | gs_cat_1 == "Services" |
        gs_cat_1 == "Travel Transport" | gs_cat_1 == "Athletics Sports" | gs_cat_1

  c_meds <- ddply(filtered_data, .(gs_cat_1), summarise, med = median(distance))

  # Boxplot
  print(ggplot(filtered_data, aes(x = gs_cat_1, y = distance)) +
    geom_point(colour = "lightblue", alpha = 0.6, position = "jitter") +
    geom_boxplot(outlier.colour = "red", outlier.size = 1, alpha = 0.1) +
    coord_flip() +
    geom_text(data = c_meds, aes(y = med, x = gs_cat_1, label = round(med,3), hjust
    scale_x_discrete(name="Categories") +
    scale_y_continuous(name="Error Distance (m)") + labs(title=providers[i])
  )

  print(providers[i])
```

```
  print(length(data$distance))
  print(summary(data$distance))

  # scatter plot of error and name similarity
  plot(data$distance, data$name_similarity, cex=0.3, main=providers[i], sub="Visual

  # correlation cooefficient
  print( cor(data$distance, data$name_similarity))
}
```

3. Code for Spatial Uncertaintly Data Collection

```
#!/usr/bin/env ruby

require 'fuzzystringmatch'
require 'timeout'

jarow   = FuzzyStringMatch::JaroWinkler.create( :native )


# sqlite db incase we want to compare the categories from Vienna
#
# db                  = SQLite3::Database.open './db/Herold_Wien_WGS84.sqlite'
# db.results_as_hash  = true
# stm                 = db.prepare "SELECT * FROM Herold_Wien_WGS84 WHERE SID = :pr

# ["Education", "Athletics Sports", "Religion", "Culture", "Services", "Entertainme
# categories_en = CategoryCounter.select(:category).uniq.pluck :category # not used

# => ["foursquare", "eventful", "wikimapia", "dbpedia", "yellow_pages", "lastfm", "
providers_en = ["foursquare", "eventful", "wikimapia", "dbpedia", "yellow_pages", "

area_id     = 3
provider_a  = "yellow_pages"
provider_b  = ( ARGV[0] if ARGV[0] and providers_en.include?(ARGV[0]) )
abort("a valid provider not given. select from #{providers_en.inspect}") if provide

total_venues_provider_a = Venue.where(provider: provider_a).where(area_id: area_id)
total_venues_provider_b = Venue.where(provider: provider_b).where(area_id: area_id)

min_max_x_y_hash      = Area.find(area_id).bbox.to_min_max_x_y_hash
geo_factory           = RGeo::Geographic.spherical_factory(srid: 4326)
radius_tolerance      = 0.001
max_checks            = 1000_000
total_matches         = 0
total_checks          = 0
ofile                 = "spatial_match_#{provider_a}_#{provider_b}_#{radius_tolerar
output                = CSV.open(ofile, 'w+')
headers               = ['venue_a.id', 'venue_b.id', 'venue_a.name', 'venue_b.name'
```

```
output << headers
provider_a_matched_ids = []

Query = Struct.new(nil) do
  def self.st_distance(point_a, point_b)
    rec = ActiveRecord::Base.connection.execute(
      ActiveRecord::Base.send(:sanitize_sql_array,
        "SELECT ST_Distance(ST_GeographyFromText('SRID=4326;#{point_a.as_text}'), S
      )
    )
    rec.first["distance"]
  end

  def self.find_categories(venue)
    results = ActiveRecord::Base.connection.execute("SELECT * from pois_with_categ
    if results.count > 0
      rec = results.first
      [rec["gs_cat_1"], rec["gs_cat_2"], rec["gs_cat_3"]]
    else
      cats = venue.categories.order('primary_category desc').pluck(:name)[0..2]
      return cats if cats.blank?

      sql_cats =  cats.collect{|a| "'#{a.sub("'", "\\\'")}'"}.join(',')
      stmt = "SELECT * from pois_with_categories where source_category IN (#{sql_ca
      results = ActiveRecord::Base.connection.execute( ActiveRecord::Base.send(:sar
      if results.count > 0
        rec = results.first
        [rec["gs_cat_1"], rec["gs_cat_2"], rec["gs_cat_3"]]
      else
        cats
      end
    end
  end
end

venues_a = venues_a = Venue.where(area_id: area_id).where(provider: provider_a).loa

venues_a.each do |venue_a|
  sample_point = venue_a.coordinates
  match_found = false

  # try to find matching venue from provider_b
  venues_b = Venue.where(area_id: area_id).where(provider: provider_b)
```

```
    .where('st_dwithin(coordinates, ?, ?)', sample_point, radius_tolerance)
    .order("st_distance(coordinates, ST_Geomfromtext('#{sample_point.as_text}', 432

  venues_b.each do |venue_b|
    total_checks += 1
    name_similarity = jarow.getDistance(venue_a.name, venue_b.name)
    if name_similarity > 0.90
      puts "match found between #{venue_a.name} and #{venue_b.name}"
      provider_a_matched_ids << venue_a.id
      category   = venue_b.categories.order('primary_category desc').pluck(:name).f
      distance   = Query.st_distance(venue_a.coordinates, venue_b.coordinates)
      categories = Query.find_categories(venue_b)
      output << [venue_a.id, venue_b.id, venue_a.name, venue_b.name, venue_a.coordi
      match_found = true
      total_matches += 1
    end
    next if match_found
  end
end

output << ["="*70]
output << ["providers: #{provider_a}, #{provider_b}"]
output << ["total_venues_provider_a: #{total_venues_provider_a}"]
output << ["total_venues_provider_b: #{total_venues_provider_b}"]
output << ["max_checks: #{max_checks}"]
output << ["total_checks: #{total_checks}"]
output << ["total_matches: #{total_matches}"]
output << ["radius_tolerance: #{radius_tolerance}"]

output.close
puts "total_matches: #{total_matches}"
puts "total_checks: #{total_checks}"
puts "total_venues #{provider_a}: #{total_venues_provider_a}"
puts "total_venues #{provider_b}: #{total_venues_provider_b}"
```