Geographical data extraction

Need for this data extraction:

The AQI index and pm2.5 values are dependent on its Geographical features and meteorological data. An area which is urbanized and have dense road network, have many restaurants or factories will have high pm2.5 measurement than areas, which are less urbanized. Lijian Han et.al 2016. Various studies such as one conducted at Jianhua Wang et.al at Nagasaki Japan shows that Pm2.5 has high correlation with various meteorological factors. Our Aim is to develop a model which incorporates both meteorological and geographical data and helps us solve the above two problem. More ever solving this problem will also pave the path for better forecasting and anomalies detection with much better accuracy.

The data extraction process:

The geographical data has been obtained from open street map. The open street map provides us data in the form of nodes, ways or relations. Nodes are only a specific latitude and longitude location which shows an entity. ways is combination of many nodes and it gives the area which the geographical entity surrounds or a linear feature of entity. Relation is combination of many ways.

Our idea is to extract all those entities in a buffer with radius 564 meters. This gives us $A = \pi\ 564^2$= 999328 m^2 which is roughly equal to 1 Km^2. For the nodes we counted the number of that specific nodes in that buffer e.g. restaurants, shops, no. of signals etc. for the ways we either calculated the area or calculate length in meters, for relation we have combined those structures and calculated the area.
The flowchart below shows how the data has been extracted.

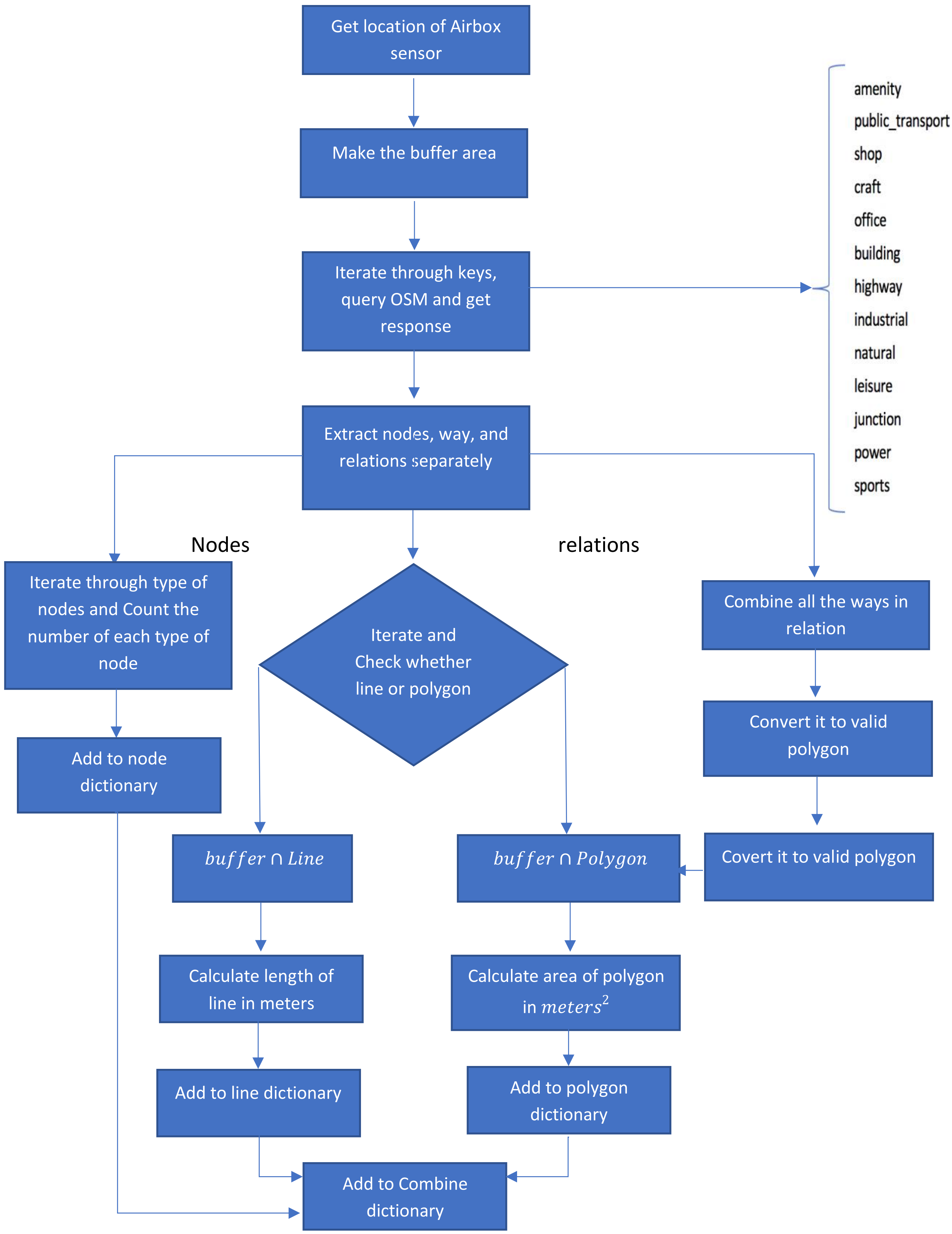


FIG 1. FLOW CHART OF GEOGRAPHICAL DATA EXTRACTION

We already have the sensors locations. We mark that as a center point and make a buffer around that area shown in figure 1. then we query the information for the corresponding tag from open street map through overpass API.

FIG 2: BUFFER AREA

The nodes, way and relation are extracted from the response. Each type of key has several entity values. Tag key ‘amenity’ has values [school, café, restaurant, hospital, place_of_worship, ………]. We extract those entities which lies inside the circle and then we count number of key value and add it to dictionary [school: 3, hospitals: 5, cafe:9, ……]. Figure 2 shows extracted nodes.

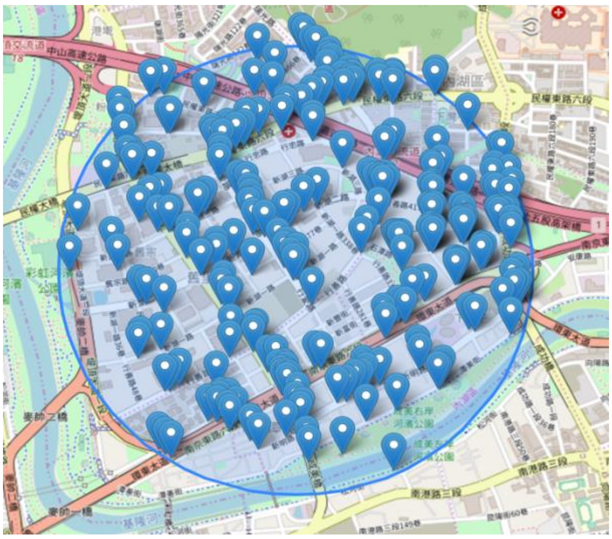
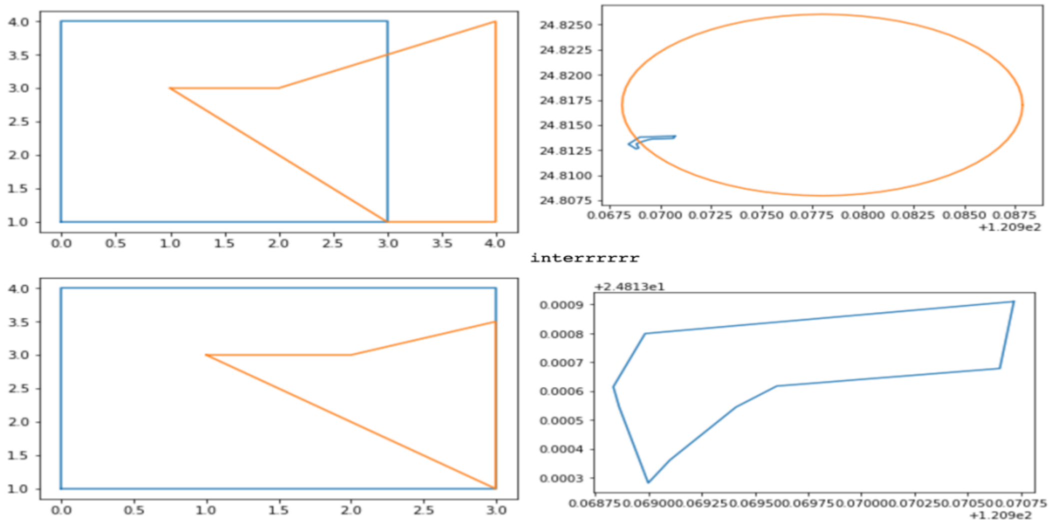


FIG 3: NODES IN THE BUFFER AREA

The ways are further subdivided into two categories, either it will be a polygon or a line. The starting and ending value are compared, if both are same we consider it as polygon and if different we treat it as line.

When we select polygons, some of the polygons reside on the boundary wall of buffer. Part of the area reside inside the buffer while part of the area resides outside the buffer. We need to extract the area which is inside the buffer. Figure 3a and 3b shows such an example. In figure 3a the blue rectangle is the buffer zone while the orange one is polygon crossed the boundary of buffer. The lower figure in 3a shows that boundary inside the buffer is extracted. Figure 3b the upper region shows real scenario from osm when a polygon shown in blue crossed the circular buffer shown in orange color. the lower figure 3b shows the extracted polygon.



4(a) 4(b)

FIG 4: SHOWS THE CROSSING BOUNDARY REGIONS AND THE EXTRACTION OF OVERLAPPING PARTS ONLY

For the polygons structure the overlapping area of polygon and the buffer is calculated and extracted as a new polygon. The meter square area of new polygon is calculated. All of those areas with its feature name is saved in a dictionary. Some of the sub-tags and its values are here [parking: 408.62, marketplace: 10, school: 47005.56, mall: 2021.47, …]. The values are meter square area of that entity in a buffer. Figure 4 shows all the polygons extracted in a single buffer.

For the line structure we again find overlapping part of the line and then find the length. The length of the line is not Euclidean distance from starting to ending point instead we calculate the distance between consecutive points till the end and add it. In this way we get the exact length of the line. This line can be a road or any other feature which exhibits the line feature. Every tag has sub-tags which exhibits line features such as highway has tags [residential: 17109.04, service: 16273.16, secondary: 1785.69, footway: 424.87, ….]. The values are length in meters in that buffer we save these lengths as a dictionary. Figure 5. Shows all the line structure in a single buffer.

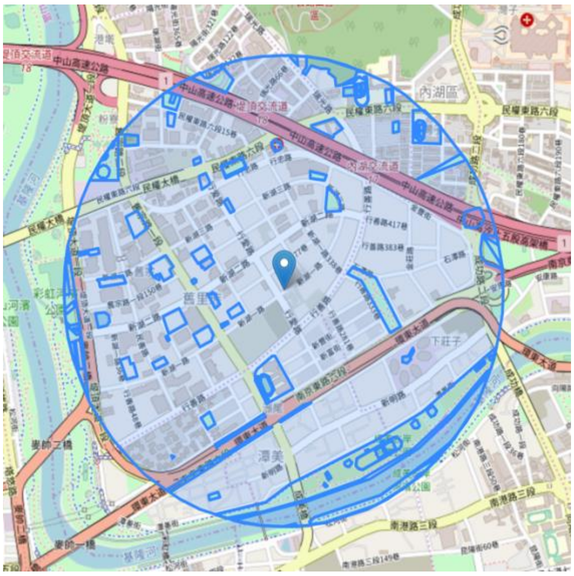


FIGURE 5: POLYGON AREAS INSIDE THE BUFFER

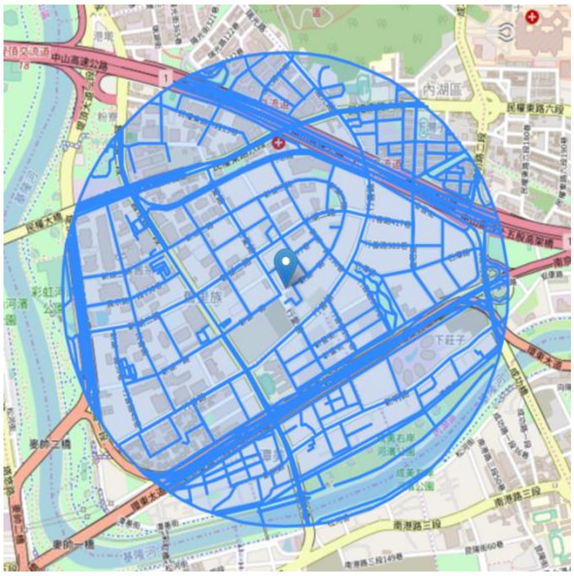


FIGURE 6. LINES SHAPES INSIDE BUFFER

The last one is the relation. A relation is combination of ways structure, mostly polygons. We first combine directly all the ways in a relation but the combination does not give us polygon structure, as those shapes may have self-intersection or multi-point intersection thus does not exhibit a polygon behavior. We then extend the shape by applying buffer(0) function of shapely library and get all the exterior points of the shape. Those exterior points are the structure of polygon. We then treat the shape as normal polygon. The overlapping area with buffer is found and then area of the overlapped region is calculated. Which is then added to a dictionary.