Calculating distance between zip codes:
A database case study in consuming web services

Mohammad Dadashzadeh
Oakland University

ABSTRACT

Travel distance is a variable in predicting consumer behavior in choosing a service location. In predictive modeling applications in medicine, there is often a need to calculate the distance of a patient from one or more healthcare facilities. With de-identified patient data, a patient’s zip code becomes the basis for calculating distance. Therefore, the problem of calculating distance between zip codes is a recurring data pre-processing step in such applications. This paper presents the development of a case study for the database course to automate a solution to the problem while providing teaching opportunities in Structured Query Language (SQL), Microsoft Access Visual Basic for Applications (VBA) programming, and consuming web services.

Keywords: IS Curriculum, Database Course, Location Analytics, Travel Distance Calculation, Geocoding API, Map Web Services

Copyright statement: Authors retain the copyright to the manuscripts published in AABRI journals. Please see the AABRI Copyright Policy at http://www.aabri.com/copyright.html
INTRODUCTION

Travel distance is a variable in formulating an understanding of why a specific service location amongst several competing ones is chosen by a consumer. The need to calculate geographic distance occurs in various contexts including tourism research (Nyaupane, Graefe, & Burns, 2003), emergency medical services (EMS) system planning (Hsia et al., 2017), and healthcare provision (Tsai, Orav, & Jha, 2015).

To fix ideas, let us consider an example of calculating distance between patients and hospitals. The data structure in Microsoft Access consists of two tables: tbl_Hospitals (Figure 1) that records the zip code of 14 service locations of interest, and tbl_Zipcodes (Figure 2) that captures the 1,505 distinct patient zip codes for which the distance to each service location needs to be calculated.

**Figure 1.** Microsoft Access Table Listing Various Service Locations

<table>
<thead>
<tr>
<th>Hospital_ID</th>
<th>Hospital</th>
<th>Zipcode</th>
<th>Latitude</th>
<th>Longitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UPMC McKeensport</td>
<td>15132</td>
<td>40.341919</td>
<td>-75.84791</td>
</tr>
<tr>
<td>2</td>
<td>UPMC East</td>
<td>15146</td>
<td>40.431034</td>
<td>-75.76526</td>
</tr>
<tr>
<td>3</td>
<td>UPMC Pittsburgh</td>
<td>15213</td>
<td>40.443269</td>
<td>-75.95467</td>
</tr>
<tr>
<td>4</td>
<td>UPMC St. Margaret</td>
<td>15215</td>
<td>40.497701</td>
<td>-75.9157</td>
</tr>
<tr>
<td>5</td>
<td>UPMC Passavant</td>
<td>15237</td>
<td>40.552768</td>
<td>-80.03227</td>
</tr>
<tr>
<td>6</td>
<td>UPMC Bedford</td>
<td>15537</td>
<td>40.003997</td>
<td>-78.36456</td>
</tr>
<tr>
<td>7</td>
<td>UPMC Passavant - Cranberry</td>
<td>16066</td>
<td>40.700423</td>
<td>-80.11374</td>
</tr>
<tr>
<td>8</td>
<td>UPMC Jameson</td>
<td>16105</td>
<td>41.038205</td>
<td>-80.34539</td>
</tr>
<tr>
<td>9</td>
<td>UPMC Horizon - Shenango Valley</td>
<td>16121</td>
<td>41.211606</td>
<td>-80.49404</td>
</tr>
<tr>
<td>10</td>
<td>UPMC Horizon - Greenville</td>
<td>16125</td>
<td>41.403462</td>
<td>-80.37424</td>
</tr>
<tr>
<td>11</td>
<td>UPMC Northwest</td>
<td>16346</td>
<td>41.387236</td>
<td>-79.6884</td>
</tr>
<tr>
<td>12</td>
<td>UPMC Hamot</td>
<td>16550</td>
<td>42.182748</td>
<td>-80.64915</td>
</tr>
<tr>
<td>13</td>
<td>UPMC Altoona</td>
<td>16601</td>
<td>40.526319</td>
<td>-78.40082</td>
</tr>
<tr>
<td>14</td>
<td>UPMC Susquehanna Williamsport</td>
<td>17701</td>
<td>41.256736</td>
<td>-77.0108</td>
</tr>
</tbody>
</table>

**Figure 2.** Microsoft Access Table Showing 18 of 1,505 Distinct Patient Zip Codes
Given a user-defined function named `fnDistanceBetweenZipcodes` (Figure 3) to calculate and return the distance between two zip codes, the following SQL statement would create the table `tbl_ZipcodeDistances` (Figure 4) consisting of 21,070 (i.e., 1505 x 14) rows showing the desired distances:

```sql
SELECT ID, tbl_Zipcodes.Zipcode AS Zip1, tbl_Hospitals.Hospital, 
tbl_Hospitals.Zipcode AS Zip2, 
fnDistanceBetweenZipcodes([Zip1], [Zip2]) AS Distance INTO 
tbl_ZipcodeDistances 
FROM tbl_Zipcodes, tbl_Hospitals
```

![Figure 3. The Structure of VBA Function to Return Calculated Distance Between Zip Codes](image)

![Figure 4. Sample of the Output Table tbl_ZipcodeDistances Showing Returned Distances](image)
The final solution step would be to create a cross tabulation of data in the table, \textit{tbl\_ZipcodeDistances}, that more clearly shows the distance between each distinct patient zip code and the 14 service location zip codes (Figure 5). This can be accomplished using the Crosstab Query Wizard in Microsoft Access as:

\begin{verbatim}
TRANSFORM First(tbl_ZipcodeDistances.[Distance]) AS FirstOfDistance
SELECT tbl_ZipcodeDistances.[Zip1]
FROM tbl_ZipcodeDistances
GROUP BY tbl_ZipcodeDistances.[Zip1]
PIVOT tbl_ZipcodeDistances.[Zip2]
\end{verbatim}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure5.png}
\caption{Layout of the Desired Final Output}
\end{figure}

As it has been outlined above, the software architecture of a solution to the problem builds upon concepts, namely SQL and VBA programming, taught in a typical database course in the MIS curriculum using Microsoft Access. However, the important new teaching opportunity that emerges is in completing the function \textit{fnDistanceBetweenZipcodes} to return the correct distance by utilizing existing web services.

**DISTANCE CALCULATION USING ZIP CODE WEB SERVICES**

Zip-Codes.com (2019) is a supplier of US and Canadian zip code database/directory. It also provides a subscription-based Application Programming Interface (API) to its web services for functions including obtaining the zip code of an address, distance calculations, and radius searching. The Zip-Codes.com’s API is a RESTful service (Richardson & Ruby, 2007) in that it accepts HTTP requests and provides responses in XML or JavaScript Object Notation (JSON) text formats. For example, the request to calculate the distance between zip codes \texttt{32504} and \texttt{90210} and receive the response in XML format can be sent via the following URL:

\begin{verbatim}
https://api.zip-codes.com/ZipCodesAPI.svc/1.0/XML/CalculateDistance/ByZip
?fromzipcode=32504&tozipcode=90210&key=DEMOAPIKEY
\end{verbatim}

where, the API subscription key of DEMOAPIKEY is used. The XML response from the web service would include text:

\begin{verbatim}
<DistanceInMiles>1835.096876886239</DistanceInMiles>
\end{verbatim}

that can be parsed to obtain the calculated \textit{straight-line} distance of 1835 miles.
The Appendix gives the completed code for the Microsoft Access user-defined function, \texttt{fnDistanceBetweenZipcodes}, that utilizes the Zip-Codes.com’s API to calculate the needed distances.

**DISTANCE CALCULATION USING HAVERSINE FORMULA**

The straight-line distance between two locations ignores the fact that there are no straight lines on a sphere such as Earth. The haversine formula (Wikipedia, 2019) can be used to calculate the shortest spherical distance between two points on the surface of a sphere, measured along the surface. Although the formula would not be completely accurate since the Earth is not a perfect sphere, it provides a good approximation for most applications. The basic formula utilizes latitude and longitude coordinates of the locations and can be expressed as a VBA function in Microsoft Access as shown in the Appendix.

This approach to calculating distances between zip codes requires that we obtain latitude and longitude coordinate values associated with each of our 1,505 distinct zip codes. Geocoding web services that take an address and return an actual or calculated latitude/longitude coordinate can provide that functionality. However, several sites including UnitedStatesZipCodes.org (2019) provide a “personal” version of their zip code database at no cost. Figure 6 shows the data structure of the zip code database as imported into Microsoft Access that supplies the needed latitude and longitude coordinates for distance calculation.

![Zip Code Database](image)

**Figure 6.** Zip Code Database Supplying Latitude and Longitude Coordinates

**DISTANCE CALCULATION USING ROUTING WEB SERVICES**

For applications requiring a more accurate calculation of travel distance between two locations, the solution may be found by using one of the many routing web services available including: Google Maps Platform’s Directions API (Google, 2019), Bing Maps REST Services (Microsoft, 2019), ESRI’s ArcGIS REST API (ESRI, 2019), and HERE Technologies’ Routing API (HERE Technologies, 2019).
Using HERE Technologies’ routing web service, the request to calculate the fastest travel route by car between zip codes 32504 (30.48°N, -87.19°E) and 90210 (34.10°N, -118.41°E) taking traffic conditions into account and departing immediately can be sent via the following URL:

https://route.api.here.com/routing/7.2/calculateroute.json
?waypoint0=30.48, -87.19
&waypoint1=34.10, -118.41
&mode=fastest; car; traffic:enabled
&app_id=devportal-demo-20180625
&app_code=9v2BkviRwi9Ot26kp2IysQ
&departure=now

where, the values specified for request parameters app_id and app_code are developer assigned subscription keys. The JSON response from the web service would include text:

"The trip takes <span class="length">3369 km</span>"

that can be parsed to obtain the calculated routing distance of 3369 kilometers (or 2093 miles).

The Appendix gives the completed code for the Microsoft Access function, fnGetRoutingDistanceHERE_API, that utilizes HERE.com’s API to calculate the needed distances.

SUMMARY AND CONCLUSIONS

The IS 2010 Curriculum Guidelines for Undergraduate Degree Programs in Information Systems (ACM/AIS, 2010) recognizes the “emergence of a new architectural paradigm – service-oriented architecture, web services, software-as-a-service, and cloud computing” as a motivating factor for IS curriculum revision. As such, core skill sets for modern application development must include AJAX, XML, and web services in addition to the basic programming, web development, and database skills that have been taught for many years. Dadashzadeh (2010) describes a case study to expose IS students to web services as soon as they are introduced to the basics of HTML and programming. In this paper, the challenge of solving a real-world problem in data preparation for predictive modeling applications involving location analytics, that is, the problem of calculating distance between zip codes, has been presented as a pedagogical opportunity for the database course instructor to introduce students to modern web services and API’s in a hands-on manner.
REFERENCES


APPENDIX

This appendix provides Microsoft Access VBA code referred to in the paper. This partial code for implementing the solution of calculating distance between zip codes is taught to, and shared with, the students along with the case study. A copy of the example database and the entire solution code is available from the author upon request.

A. VBA function to calculate distance between zip codes using Zip-Codes.com API

Function fnDistanceBetweenZipcodes (Zipcode1 As String, Zipcode2 As String) As Single

'Function to calculate distance between zip codes using Zip-Codes.com API ...
Dim strURL As String, BodyTxt As String
Dim Temp As Variant 'Return value ...

Dim strSearch As String
strSearch = "<DistanceInMiles>" 'Search pattern to look for in response ...

strURL = "http://api.zip-codes.com/ZipCodesAPI.svc/1.0/xml/CalculateDistance/ByZip?" 
strURL = strURL & "fromzipcode=" & Zipcode1 & 
    "&tozipcode=" & Zipcode2 
    "&key=DEMOAPIKEY"

'Send the web service request ...
BodyTxt = getResponse(strURL)

'Parse the returned text for <DistanceInMiles> ...
If InStr(1, BodyTxt, strSearch, vbTextCompare) = 0 Then
    Temp = -1# 'DistanceInMiles is absent in the response ... Return -1 ...
Else
    Temp = Mid(BodyTxt, InStr(1, BodyTxt, strSearch) + Len(strSearch))
    Temp = Mid(Temp, 1, InStr(1, Temp, 
                    
    Temp = Val(Temp) 'Convert from string to numeric ...
End If

'Return Temp ...
fnDistanceBetweenZipcodes = Temp

End Function

Function getResponse (strURL As String) As String

'Submits the URL request and returns the response ...
Dim oXH As Object
Set oXH = CreateObject("msxml2.xmlhttp")

With oXH
B. VBA function to calculate distance between geocoded locations using haversine formula

Function HaversineFormulaDistance(Latitude1 As Single, Longitude1 As Single, Latitude2 As Single, Longitude2 As Single) As Single

Const EarthRadius = 3958 'in miles ...

Dim LatitudeDelta As Single, LongitudeDelta As Single
Dim A As Single, C As Single

'Convert from degrees to radians ...
LatitudeDelta = (Latitude2 * PI / 180) - (Latitude1 * PI / 180)
LongitudeDelta = (Longitude2 * PI / 180) - (Longitude1 * PI / 180)

A = ((Sin(LatitudeDelta / 2))^2) + Cos(Latitude1 * PI / 180) * Cos(Latitude2 * PI / 180) * ((Sin(LongitudeDelta / 2))^2)

C = 2 * ArcSin(Sqr(A))

HaversineFormulaDistance = EarthRadius * C

End Function

C. VBA function to calculate routing distance between geocoded locations using HERE.com

Function fnGetRoutingDistanceHERE_API(Latitude1 As Single, Longitude1 As Single, Latitude2 As Single, Longitude2 As Single) As Single

'Function to calculate routing distance between two geocodes using HERE.com API
Dim strURL As String, BodyTxt As String
Dim Temp As Variant 'Return value ...'The trip takes <span class="length">  strSearch = "The trip takes " & Chr(60) & "span class="length"" & Chr(62)

strSearch = "The trip takes " & Chr(60) & "span class="length"" & Chr(62)
strURL = "https://route.api.here.com/routing/7.2/calculateroute.json?"
strURL = strURL & "waypoint0=" & Trim(Str(Latitude1)) & "%2C" & Trim(Str(Longitude1))
strURL = strURL & "waypoint1=" & Trim(Str(Latitude2)) & "%2C" & Trim(Str(Longitude2))
strURL = strURL & "mode=fastest%3Bcar%3Btraffic%3Aencoded"
strURL = strURL & "app_id=devportal-demo-0180625&app_code=9v2BkviRwi9Ot26kp2IysQ"
strURL = strURL & "departure=now"

'Send the web service request ...
BodyTxt = getResponse(strURL)

'Parse the returned text for strSearch value ...
If InStr(1, BodyTxt, strSearch, vbTextCompare) = 0 Then
    Temp = -1# 'strSearch is absent in the response ... Return -1 ...
Else
    Temp = Mid(BodyTxt, InStr(1, BodyTxt, strSearch) + Len(strSearch))
    Temp = Mid(Temp, 1, InStr(1, Temp, "<") - 1)

    'See if units is meters or kilometers ...
    If InStr(Temp, "km") <> 0 Then
        Temp = Val(Temp) * 0.00062137 * 1000 'Convert km to mile ...
    Else
        Temp = Val(Temp) * 0.00062137 'Convert m to mile ...
    End If
End If

'Return Temp ...
fnGetRoutingDistanceHERE_API = Temp

End Function