Implementing a Natural Language Processing Framework to Perform Spatial Searches of OpenStreetMap Features in ArcGIS

April 24, 2013

Gary R. Huffman
grh145@psu.edu

MGIS Capstone Project
The Pennsylvania State University

Faculty Advisors:
Dr. Alex Klippel
Dr. Jan Wallgrün

I. Abstract

This paper describes the development of a framework for an ArcGIS Desktop application that uses natural language processing tools and full-text indexing libraries to search for basic topological relationships within OpenStreetMap data. Tools such as named entity recognizers and part-of-speech taggers can be run directly within the GIS application enabling natural language search functions typically available only through web-based mapping applications like Bing and Google. Many of these web-based mapping tools do not identify topological relationships described by spatial prepositions such as in and on, but typically return results based upon bounding boxes or an address geo-coding attribute such as place name. Challenges and complexities attributed to natural language processing make this a difficult task, as certain spatial relationships defined by prepositions can be vague and ambiguous. While humans can understand the grammar and syntax in language, computers have difficulty manipulating the concepts and implied meanings within in a spatial search. This proof-of-concept ArcGIS application processes basic natural language searches and generates spatial queries that can identify the intended spatial prepositions such as contains and intersects. This provides a more natural search interface for the GIS user while also expanding the role of OpenStreetMap within ArcGIS beyond that of a base map option.

II. Problem Description

Information retrieval within the context of a Geographic Information System (GIS) typically requires the use of Structured Query Language (SQL) and a basic understanding about the data structure. While well suited for standard GIS use, the limitation of SQL and the nature by which users interact with GIS precludes advanced searches much beyond simple term matching as in [Points_of_Interest_Name]= "Starbucks" or [Feature_Type] = 'Café'. A slightly more advanced version might allow for the use of wildcards as in [Points_of_Interest_Name] like '*Starbucks*'. Spatial queries frequently require the use of units of measure when searching for features having a certain area or length or multiple features within a certain distance of one another. Furthermore, the type of spatial relationship must be explicitly identified. Topological relationships - namely those that do not rely upon numeric distances - are more complex and involve the use of advanced geoprocessing tools and SQL statements. Most users, especially those who are accustomed to interacting with on-line search engines like Bing and Google are comfortable searching with natural language. For example, (1) "Starbucks in Vienna, VA" and (2) "Parks near Vienna" are ill suited for a typical GIS SQL query especially if the user is oblivious to the way in which the data are stored. However, these are very common methods to search the Internet and a more natural way for a user to interact with a GIS.

Far less obvious are the complexities surrounding the use of spatial prepositions in the two examples. Hall (2008) describes the "vagueness of spatial relations" and attempts to quantify these in terms of spatial prepositions. The spatial operations implied by 'near' and 'in' are ambiguous. The preposition term 'in' from example (1) suggests the topological relationship "containment" as in "Starbucks represented as one or more point objects that fall within the spatial boundary defined by the polygonal area of Vienna, VA." Of course, the notion of containment for Starbucks within Vienna, VA becomes very difficult to conceptualize if a point feature derived from a gazetteer represents Vienna, VA.

The term 'near' is equally ambiguous in example (2) given above. Without a notion of a distance, reference, or scale, the term 'near' can define anything from 1 mile to 50 miles, in all directions or a specific direction. In the linguistic sense of spatial prepositions, Coventry and Garrod (2004) describe 'near' as a *projective* term (along with 'far') that "…give information about the distance between objects." Further complicating the use of projective prepositions is the fact that they can fall into different categories of reference: intrinsic (object-centric); relative (viewer-centric); and absolute (environment-centric) (ibid). In example (2), the type of spatial relationship suggested by 'near' is unclear and could imply parks: 1) completely within the geographic boundary of Vienna; 2) within the geographic boundary yet some portion of the park intersects the boundary; or 3) external to the geographic boundary of Vienna but within a certain distance.

Natural Language Processing (NLP) is a computer science discipline that addresses the understanding of concepts and intended meanings in human language. While humans are fairly adept at understanding the linguistic and grammatical syntax as well as the implied spatial relationships in the previous examples, computers have great difficulty processing natural language queries. NLP tools and software application programming interfaces (APIs) have evolved to enable computers to extract named entities from text (e.g., recognize Starbucks as an organization), translate information between languages and, as described by Kordjamshidi, Hois, van Otterlo and Moens (2011), begin to interpret spatial natural language. Examples of these systems and APIs include the Natural Language Toolkit (NLTK, 2013), Stanford University's coreNLP (Stanford, 2013) and the University of Sheffield's General Architecture for Text Engineering (GATE, 2013).

Unfortunately, these tools have not yet made their way into the ArcGIS desktop suite of applications. Presently a user must conduct searches through a controlled interface that constructs a SQL statement or by using a text search capability that allows for a more generic term search (obfuscating some of the complexity that exists when using SQL). Furthermore, ArcGIS does not support using natural language as a search mechanism to identify topological relationships. These must be implemented using yet another specialized selection tool that constructs spatial queries based upon specific choices regarding the desired spatial relationships.

OpenStreetMap (OSM) is the Wikipedia of spatial information and is composed primarily of user contributions and some donated data that provides near-worldwide coverage of GIS data. Individuals can create and update the OSM database through the use of a GPS receiver or through heads-up digitizing with one of the popular OSM client applications. This makes OSM unique in that it contains features that do not exist in any other GIS data set. Even better, the entire database, comprised of 21 gigabytes of compressed XML data, is free for anyone to use under the Open Database License (ODbL) share-alike license (OpenStreetMap, 2013)

This is likely the reason the number of commercial applications utilizing OSM has grown. In early 2012, the social networking company FourSquare migrated from the Google Maps base map interface to the OSM base map due to increased costs and licensing issues (FourSquare Blog, 2012). More recently, and perhaps more significantly, Apple adopted portions of OSM to support a new release of its iPhoto application used to geotag digital pictures (Bennett, 2012). OSM is also available as a base map option within ArcGIS Desktop. In this form, OSM tiles are included as a read-only base map layer upon which other geographic features can be displayed.

While OSM is not the only base map option available within ArcGIS, in many instances it contains the highest level of detail of all the vector-based base maps as demonstrated by the figures below. One figure depicts the Bing base map covering a portion of the Penn State Campus in State College, PA while the other uses the

OSM base map. Finally, a portion of extracted building features from the OSM database is layered upon the Bing base map.

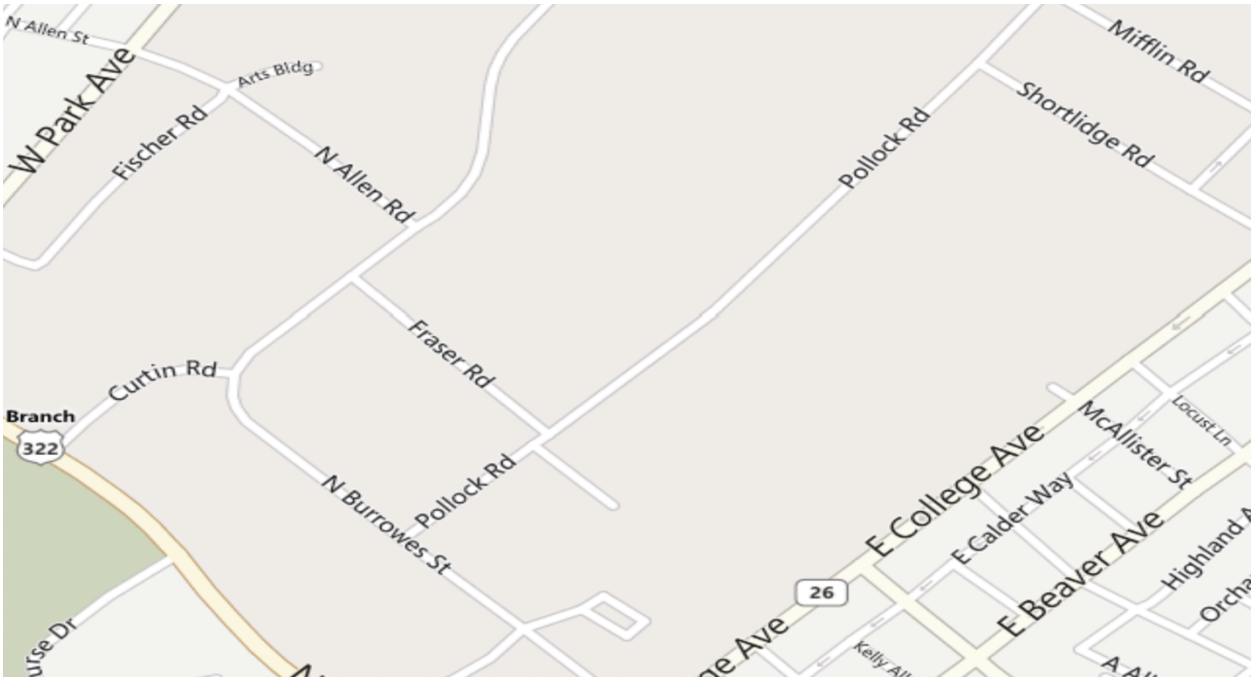


Figure 1 - ArcGIS Map over the Penn State Campus using the Bing base map

Figure 2 - ArcGIS Map over the Penn State Campus using the OSM base map

Figure 3 - ArcGIS Map over the Penn State Campus using the Bing base map layered with extracted OSM Features

Figure 3 illustrates the detail contained within the OSM database and base map relative to the other base map options within ArcGIS. Users hoping to locate a campus building using the Bing base map will not have the level of detail provided by OSM. However, the OSM database is not exposed to the “out of the box” search tools provided by ArcGIS desktop. In order to search OSM data, the user must first extract the features into a GIS database or shapefile.

Given the lack of robust search tools that support natural language queries within ArcGIS, users are required to develop complicated SQL statements to perform basic feature selections. Furthermore, the complexity coupled with the number of steps required to perform a spatial query in ArcGIS using out-of-the-box tools make it too difficult for a casual user. Features must first be selected from a source and target layer before the ‘Select by location’ tool can be used. This makes it impractical for identifying basic topological relationships between features.

III. Developed Solution

To address these problems, I have integrated natural language processing functionality into ArcGIS as a Microsoft .Net application. A user can execute a natural language query using the NLP application to search a GIS database constructed from OSM data. The application identifies parts of speech (POS) including nouns, noun phrases and prepositions within a search query string. Named entities such as locations and organizations are also identified. The

prototype supports both attribute and spatial searches based upon the query string and the information extracted from the text. In the case of a spatial search, the prototype limits the the initial number of allowable prepositions to those Garrod and Coventry (2004) describe as "simple" topological terms: **IN** and **ON** (the third preposition '**UNDER'** is applicable in 3D scenarios and, therefore, will not be considered). Figure 4 describes a preposition taxonomy as presented by Garrod and Coventry.

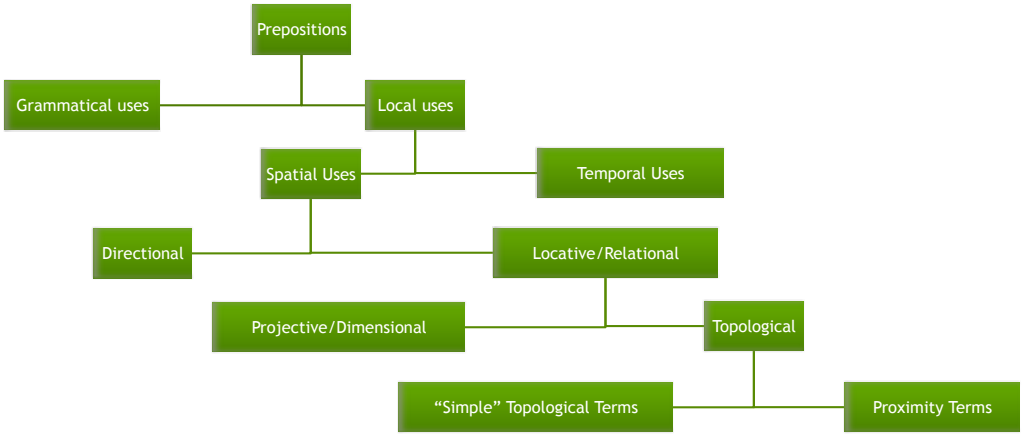


Figure 4 - Preposition Taxonomy as Defined in Garrod and Coventry (2004)

All other cases will be considered as attribute searches (those that do not contain one of the previously mentioned prepositions). The initial capabilities of the prototype are modest due to the complexities involved with the linguistic variations possible from prepositions and other parts of speech. In addition, I have limited the spatial relations to those involving only features in a point/polygon configuration.

My efforts follow the work done for the GeoCAM project (Jaiswal et al, 2011) and SensePlace2 project (MacEachren et al, 2011) that incorporate NLP tools to extract entities from unstructured text. This solution used similar well-established NLP information extraction methods but focused more on the ability to derive topological relationships from search queries.

Users input a query string and the NLP application returns a list of features that match the search parameters. The user can select an individual feature and visualize the selection in the context of OpenStreetMap data in ArcGIS. When the user inputs a query that tries to identify a topological relationship between two groups of features, the NLP application performs an ArcGIS spatial selection using the feature geometries and the spatial relationship that is derived from the spatial preposition located in the search string.

My prototype system consists of two main components: the **OSM Database Import Tool** and the **Query Processing Pipeline**.

A. OSM Database Import Tool

This module functions as a stand-alone tool that imports an OpenStreetMap XML file and creates an ArcGIS feature class for points, lines and polygons respectively. These feature classes represent spatial data as defined in OSM by nodes, ways (lines and polygons) and relations (lines and polygons) and are created in an ArcGIS file geodatabase. The Database Import tool also populates a Lucene index that includes the ObjectID for each feature written to the geodatabase. This allows feature selection using Lucene without the need to directly query the geodatabase.

MongoDB is used as a staging database while reading the OSM XML file (MongoDB, 2013). This is primarily due to the memory requirements necessary to read and process a single OSM file as they can easily exceed 1 GB in size. MongoDB is a popular NoSQL document database that is ideally suited to manage OSM features due to its flexible data model. Where a RDBMS requires a predefined schema at the table level, MongoDB allows a document - the conceptual equivalent to a table row - to have a schema independent of any other document in a collection (the conceptual equivalent of a database table).

A MongoDB database is created for the imported OSM file and separate collections created for nodes, ways and relations. The nodes collection contains a document for each node read from the OSM file including geometry definitions for every vertex used to create feature geometry in the geodatabase. The way collection contains both line and polygon features and a type value is determined based upon the vertices that make up the geometry. The relation collection is similar in that it contains both line and polygon features; however, relations are defined by existing lines and polygon features as opposed to being defined by a set of vertices. Figure 5 depicts the flow of data into the MongoDB collections for nodes, ways and relations.

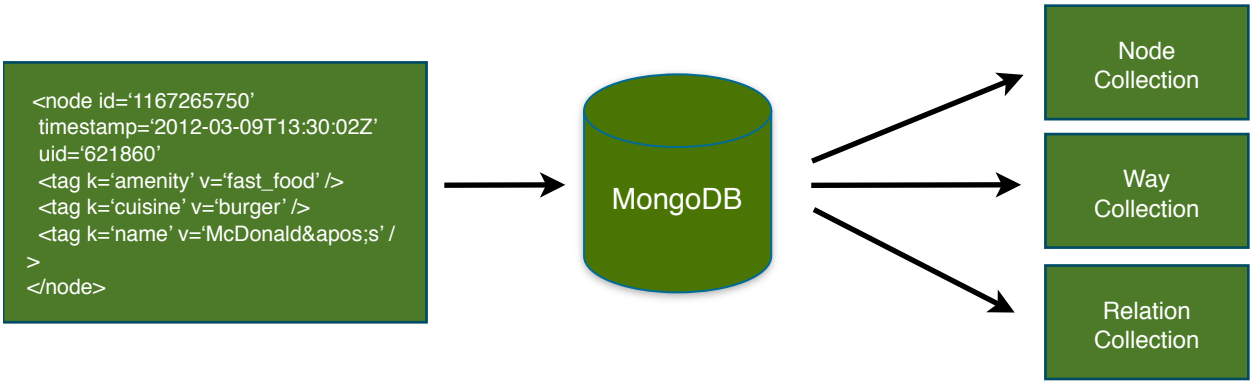
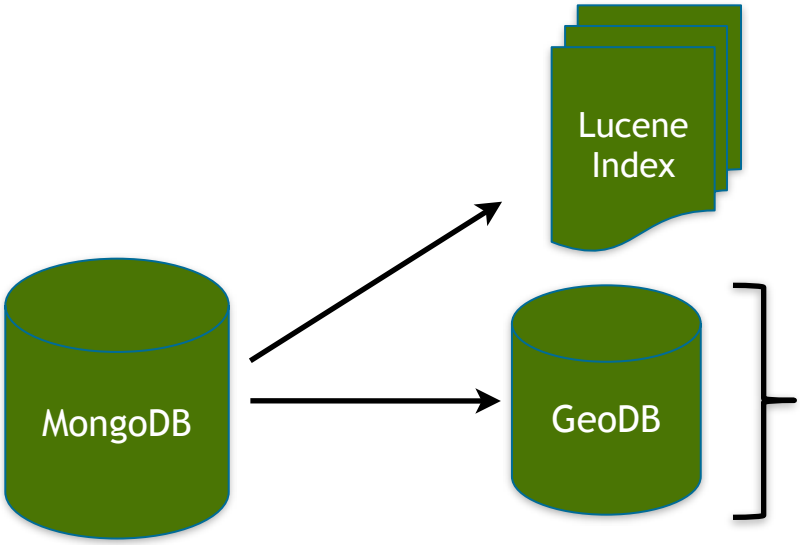


Figure 5 - OSM XML File Imported into MongoDB Collections

Nodes that are represented as features within the Point feature class are distinguished from nodes that are functionally part of other features if they have at least one tag definition that does not specify the user id or creation date of the record. This ensures that the point feature class consists of geometries that are stand-alone features (e.g., the way that represents a Starbucks café) as opposed to a way used as a vertex that only serves to define another feature (e.g., a polygon representing a building).

The geodatabase schema has a simplified structure due to the enormous flexibility OSM affords users when creating feature attribution to any given geometry. For example, a way feature can have multiple name values – each in a different language – as well as other tag values such as "place," "is_in," "tourism" and "amenity." Instead of creating separate fields to accommodate the diverse set of tags, each unique tag is concatenated into a delimited field named 'generic." In instances where the software can identify the "name" tag for a given geometry, the name field will be populated. However, the "generic" field contains all feature tags imported into the geodatabase. The format for the key/value pair will be colon separated as in "key:value" and each pair will be further delimited within the generic field using the pipe 'I' character.

During the process by which the OSM data is being loaded into the geodatabase, the OSM Database Import Tool indexes the POINT, LINE and POLYGON features using the Lucene search libraries. Lucene allows for full text indexing of the database tables and enables the searchable index to be utilized outside the geodatabase. The NLP Search application searches the Lucene index for features that match the query terms. Figure 6 depicts the process of loading the geodatabase and populating the Lucene index.



Field Name	Data Type
OBJECTID	Object ID
Shape	Geometry
OSMNODEID	Long Integer
MongoID	String
Name	String
Type	String
Generic	String

Figure 6 - Loading and Indexing the Geodatabase with Lucene

Lucene .Net is a set of libraries that support full-text indexing and search functionality to support a variety of information retrieval tasks (Apache, 2013). It does this by exposing a number of different algorithms that allow a .Net developer to tokenize and index textual information for advanced search capabilities. The index created by Lucene is often referred to as an inverted index and is depicted in Figure 7.

ID	...	Generic
1		name=Starbucks I amenity=cafe
2		name=Dunkin Donuts amenity=cafe
3		tourism=hotel I name=Homestead
4		amenity=fast_food I name=Dominos

Index with Lucene →

Token	Documents
name=Starbucks	1
amenity=cafe	1,2
tourism=hotel	3
name=Dunkin Donuts	2

Geodatabase Table

Inverted Index

Figure 7 - Example of Lucene Inverted Index For the Generic Field

Lucene creates its index by processing a string of text and applying a tokenization scheme through which individual tokens (typically individual words that appear in the source text) are identified and written to a table along with an identifier that denotes the document containing the token. When a feature is

created within the geodatabase, the text of each field from the feature class attribute table - excluding the binary feature geometry - is processed and written to the Lucene index. For simplicity and portability, the Lucene index is stored in the same folder as the ESRI geodatabase; however, the Lucene index can be stored and accessed separately from the geodatabase. This was an important design consideration as the Lucene index will return individual feature ObjectIDs from a search request without ever touching the geodatabase tables.

The OSM Database Import tool is launched using a button located on the main toolbar of ArcGIS. A dialog window displays options for the user to select and load an OSM file.

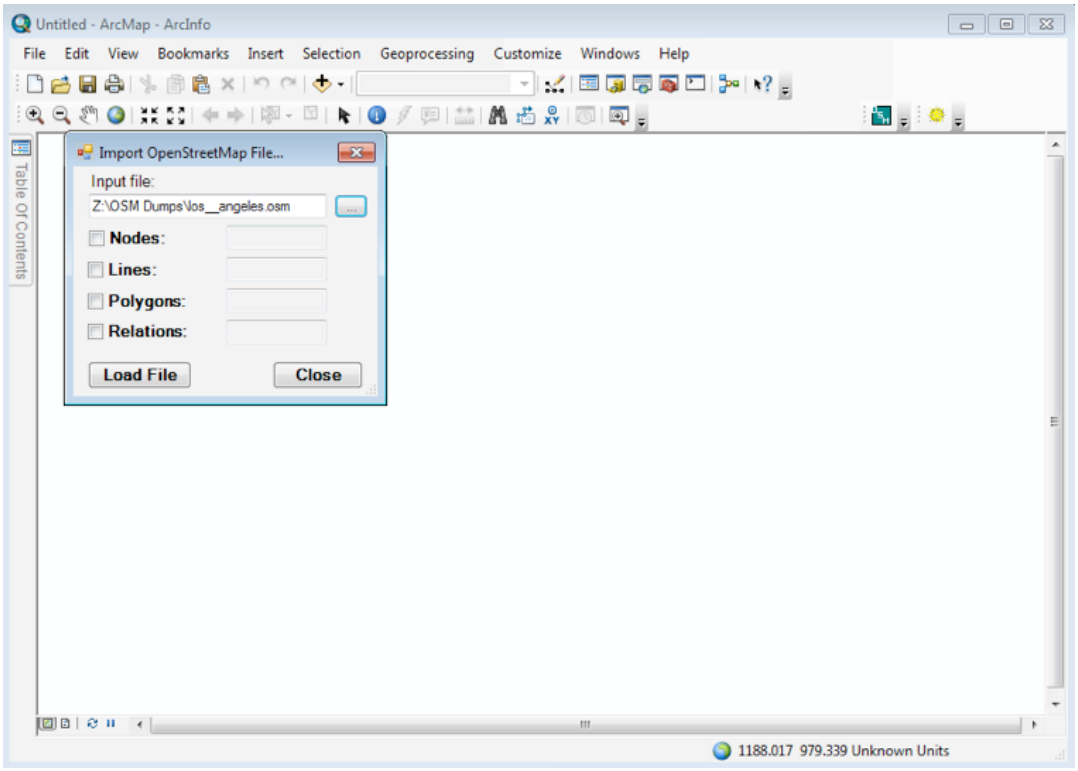


Figure 8 - OSM Database Import Tool with a file selected for processing

After clicking on the 'Load Data' button, the application extracts the contents from the OSM XML file and populates the MongoDB database. Once this process is complete, the dialog window displays record counts for each of the data types that have been extracted: Nodes, Lines, Polygons and Relations. Check boxes allow the user to determine which OpenStreetMap features should be loaded into the geodatabase.

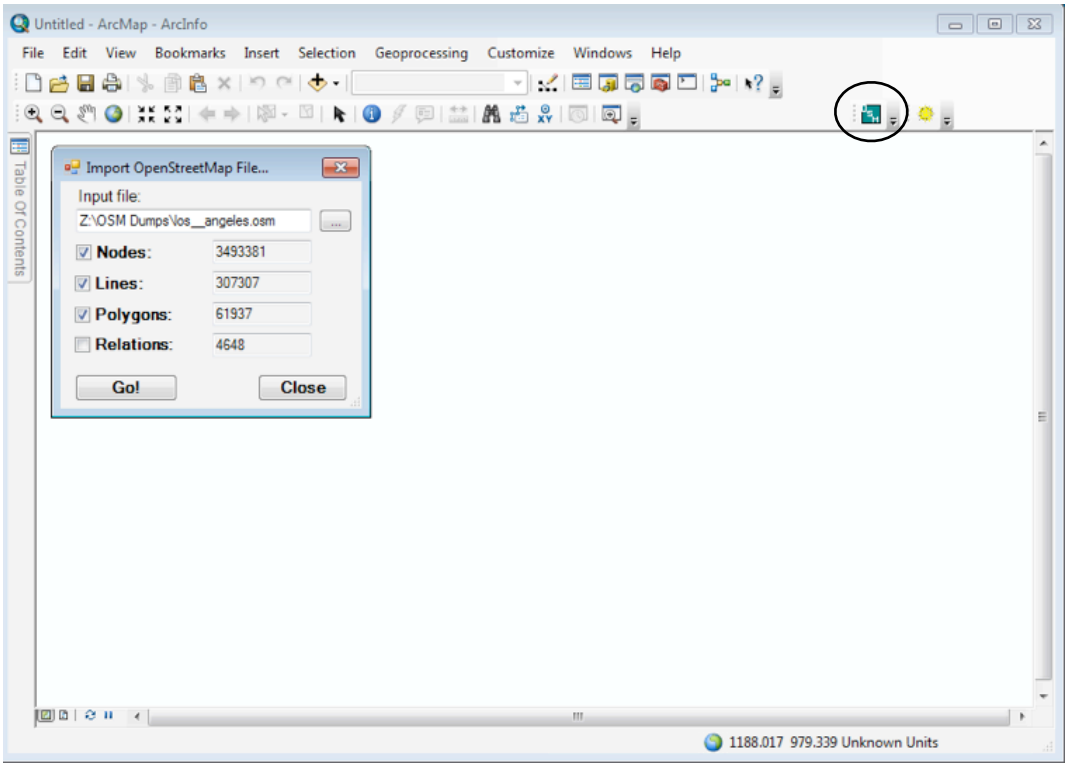


Figure 9 - OSM Database Import Tool with a file selected for processing

Once the appropriate input feature types have been selected, the user clicks the 'Go!' button to begin populating the geodatabase and creating the Lucene index. These steps happen simultaneously and can sometimes take upwards of 15 minutes depending on the amount of data being processed. After the import process is complete, the feature classes are loaded into the ArcGIS table of contents and the map area is refreshed to display the features.

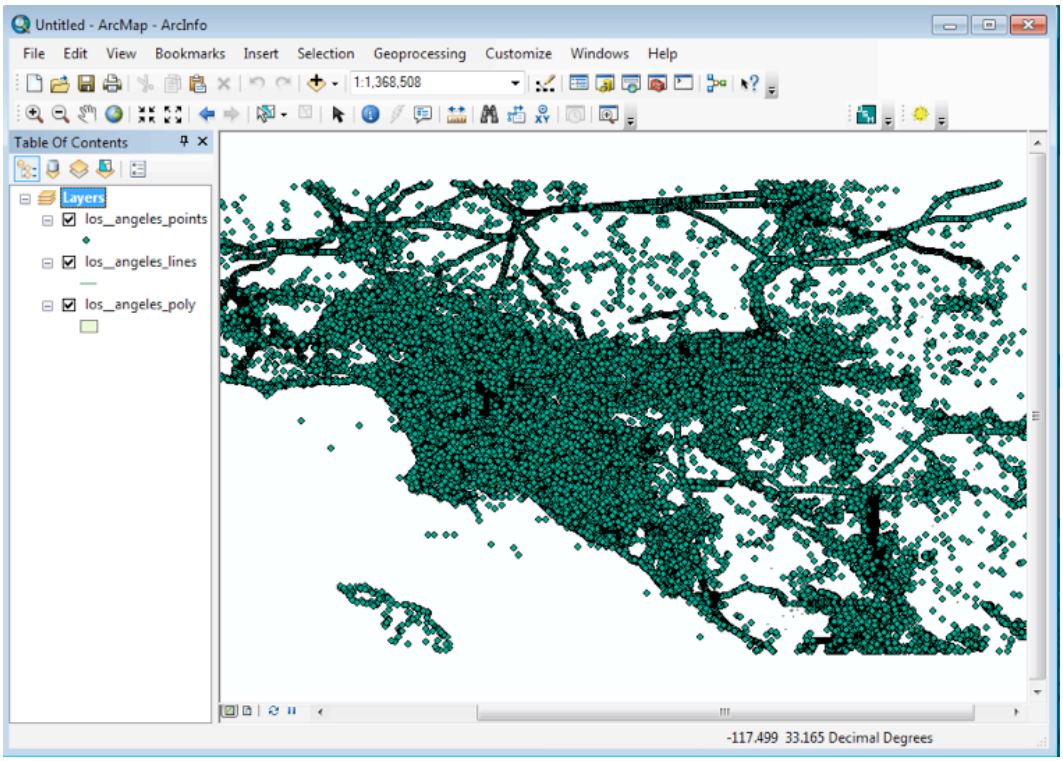


Figure 10 - OpenStreetMap Data Rendered as Geodatabase Features in ArcGIS

B. NLP Search Application

This software functions as a standalone ArcGIS application with the primary purpose of selecting and visually identifying features that have been returned from a user query expressed in natural language. It searches the Lucene index created by the OSM Database Import Tool for any feature classes loaded in the ArcGIS map view. Features that participate in either an attribute or spatial query are identified by ObjectID using the Lucene index. The spatial query seeks to identify topological relationships supported by the two spatial prepositions 'IN' and 'ON' for point/polygon whereas an attribute query tries to identify matching terms in the feature attribute table. The ObjectIDs are used to select records in the feature class attribute table in order to show a visual selection within the map view in ArcGIS.

The application processes each query through a system workflow pipeline, referred to as the 'Query Processing Pipeline,' that applies various NLP tools and queries in order to display results within the map display. Figure 11 describes the different components that make up the Query Processing Pipeline.

Figure 11 - NLP Search Application Query Processing Pipeline

Query Processing Pipeline

The 'Query Processing Pipeline' is a process workflow that performs NLP analysis and search functions on the string of text entered by the user. When the NLP Search Application is initiated, a .Net class is instantiated to store and manage the output from each step in the workflow. The 'phraseObject' class stores the source query, the parts of speech as identified for each word in the query, a spatial preposition and its location in the query string, the source and target phrase clauses, entities that exist in the source and target clause respectively and the fields to be searched. Each step processes the phraseObject and works on the output from previous steps as required.

1) Tokenize Search String

This module strips punctuation from the search string and identifies individual words from a series of characters. A specialized function is used to identify and remove punctuation likely to occur in the search string. Specifically, this removes any commas, periods and question marks. Words are identified and stored in a string array calculated by the C# string split() function that uses the space character as a separator. This enables better processing of individual words and allows the system to assign part of speech and entity tags to each word.

2) POS Tag

The POS Tag module has two primary functions: identify the part of speech for each word in the search string; and, determine if the search string contains any prepositions. The assignment of POS values to each term is performed using the Stanford University Part of Speech tagger software (Toutanova et al, 2003). This software is written in Java and implements a log-linear part of speech tagger (Ibid).

When the application is first instantiated, it loads a trained model for the POS tagger. This model ultimately enables the tagger to statistically predict the part-of-speech for each word. For the prototype, I use a model trained on annotated Wall Street Journal articles. However, it should be noted that any of the provided models can be used. Ultimately, I selected a generic model that performed well in preliminary tests using a sample of test queries.

In order to make POS tagger software function within a .Net application, I incorporated the IKVM software libraries (IKVM, 2013). This allows a Java JAR file to be wrapped within a .Net dynamic linked library (dll) such that calls can be made directly to the JAR file using C# syntax.

The POS tagger identifies any nouns, verbs, adjectives, adverbs, pronouns, prepositions and other parts of speech given an input string of text. Each word is annotated using the Penn Treebank tag set (Marcus, Santorini, and Marcinkiewicz, 1993).

If a preposition is located within the search string, it is stored in the phraseObject class and an index that describes its location within the string array is saved. The presence of a preposition also determines whether the query should be considered an attribute or spatial query.

In the case of a spatial query, the search string is further segmented into source and target phrases using the previously identified preposition as a separator in the C# string split() function.

For example, the search string “Find the Starbucks in Los Angeles” would be segmented into the following two phrases:

Source: **Find the Starbucks**; and
Target: **Los Angeles**

In the case of an attribute query, the search string becomes the source phrase while the target phrase is left undefined.

3) NER Tag

The NER Tag module identifies any location and organization entities that exist within the search string. This is done using the Stanford University Named Entity Recognizer software (Finkel et al, 2005). This software is written in Java and implements a conditional random field classifier (Ibid).

When the application is first instantiated, it loads a trained model for the NER tagger. This model ultimately enables the tagger to statistically predict the type of entity for each word. For the prototype, I use a model trained on annotated US and UK news articles. However, it should be noted that any of the provided models can be used although some of these models identify entity types that are not applicable to this prototype (e.g., monetary amounts and time). Ultimately, I selected a generic model that could identify locations, persons and organizations.

In order to make NER Tag software function within a .Net application, I incorporated the IKVM software module in the same manner as was done for the POS Tag software.

Once the source and target phrases have been identified, the NER Tag module determines whether any entities exist. Entities are tagged as either a PERSON, LOCATION or ORGANIZATION. However, the prototype ignores any words that may be tagged as a PERSON. Note this may inadvertently occur for businesses that are based upon a person's name (e.g., "Arthur Treacher"). Each phrase and any entity it may contain is managed within the 'phraseObject' class.

For example, the search string "Find the Starbucks in Vienna" would yield the following tagged sequence:

Find/O the/O Starbucks/Organization in Vienna/Location

4) Normalize Entities

Once the two Stanford NLP tools have been used to analyze the search string, some normalization is required to clean up any entities. Furthermore, there may be some instances where the NER Tag does not locate any organizations or locations in either the source or target phrase.

The most important part of normalizing entities is identifying a suitable entity value in the event the NER Tagger cannot determine an organization or location. This is done by analyzing the part-of-speech values for each word in the query string. The assumption is made that nouns are likely to be an entity in the search string. Therefore, the appropriate source and/or target phrase is analyzed and any noun identified is designated as the entity.

Additional steps are taken to refine the entity value when the entity type is assigned based upon the word's part of speech. In this instance, a special phrase dictionary is used to determine the best way of identifying features in the OpenStreetMap database. OSM has enabled a search tool called Nominatim (OpenStreetMap Wiki, 2012) that utilizes a list of user contributed Special Phrases that map terms into the OSM schema for key/value attribute pairs. For example, OSM represents airport by the key/value pair 'amenity=airport.' Because plural cases are not managed separately within OSM, a separate Special Phrases entry exists for airports that maps to the key/value pair 'amenity=airport.' Furthermore, certain key words may participate in multiple key/value pairs. This is true for the term 'Bar.' It can be identified by using either "amenity:bar" or "amenity:pub." Any entity that is defined based upon its part of speech will be filtered against a MongoDB collection of the Nominatim special phrases. In cases where a term is located within Nominatim, the OSM key/value term will be used as the entity value

(e.g., an entity term of 'airports' will be replaced by the Nominatim term 'amenity=airport).

There are certain instances in which a location name is comprised of more than 1 word as in Los Angeles. The NER Tagger software does not combine terms and simply tags each term as a location value. This requires further analysis of the string in order to combine the location terms into a single entity value that represents a location. This is done by searching for consecutive words that have the same NER tag value.

For example, the NER Tag module would process the following query string "Where are the Starbucks in Los Angels" as:

Where/O are/O the/O Starbucks/Organization in/O Los/Location Angeles/ Location

However, this module correctly merges the two terms together when defining the location entity as in "Los Angeles."

Any locations that contain a state name or abbreviation are modified to remove the state reference. This modification is made due to the lack of a state value in the OSM database for a populated place polygon (e.g., the polygon representing Vienna, Virginia does not include a tag value that includes the state name of Virginia).

5) Construct Queries

Queries are constructed based upon the source and target phrases and the types of entities identified. In instances where the NER Tag software determines the entity, the likelihood is great that the organization or location are described using the name tag value in the OSM database schema. If the entity is determined using the POS Tag software, the likelihood is great that the entity is a more generic noun or an organization/location that went unrecognized. In the case where the entity is a qualified organization/ location, the 'name' field will be searched. In the case where a noun identified by the POS Tag module is used as the entity, the 'generic' field will be searched. The 'phraseObject' is updated with the appropriate search field information for both the source and target phrases.

6) Identify Features

This module is responsible for identifying features from the geodatabase that will participate in a spatial query. The determination is based upon a query made against the Lucene index using the query fields and entity values computed by previously executed modules. A benefit to using Lucene is that all the features from the source and target groups respectively can be

identified without interacting with the geodatabase and without executing a single SQL statement. The Lucene API includes a QueryParser method that uses a string to to parse a Lucene index. Any documents that are returned by the QueryParser are scored and ranked from highest to lowest. These scores are higher when searching the name field for an entity than if searching the generic field for the same value. This is due, in large part, because the query terms are more closely matched with the contents of the Lucene name field as it does not contain the additional OSM tag values.

A specialized class called 'luceneDoc' is used to store the results in a way that can be processed by the Intersect Selections module. Each result from the Lucene query result contains a feature ID from the geodatabase, the feature type (e.g., POINT, LINE or POLYGON), and the contents from the name and generic fields. Based upon how the index was written, all attribute information is retained by the index and is recalled with the results. These are then stored in an array of 'luceneDoc' objects and saved with the 'phraseObject' class. The output of this module are sets of features that have been identified without any geodatabase interaction.

7) Intersect Selections

This module is responsible for selecting all features within the geodatabase based upon the information contained in the 'phraseObject.' If the query involves a spatial filter, the Intersect Selections module must map the spatial preposition to the allowable topological relationships implemented in the prototype. ArcGIS 10 currently supports a number of spatial selection methods that allow a user to perform a 'select by location' query. A subset of these represents topological relationships. As taken from the ArcGIS Help Documentation, these comprise the underlying spatial query executed using the ArcGIS API (ESRI, 2011). Table 2 lists the 2D spatial relationships and includes a brief description about how they are implemented within ArcGIS.

Geoprocessing Tool	Description
INTERSECT	Target layer(s) features intersect the Source layer feature
WITHIN_A_DISTANCE	Target layer(s) features are within a distance of the Source layer feature
CONTAINS	Target layer(s) features contains the Source layer feature
COMPLETELY_CONTAINS	Target layer(s) features completely contain the Source layer feature
CONTAINS_CLEMENTINI	Target layer(s) features contains (Clementini) the Source layer feature
WITHIN	Target layer(s) features are within the Source layer feature
COMPLETELY_WITHIN	Target layer(s) features are completely within the Source layer feature
WITHIN_CLEMENTINI	Target layer(s) features are within (Clementini) the Source layer feature
ARE_IDENTICAL_TO	Target layer(s) features are identical to the Source layer feature
BOUNDARY_TOUCHES	Target layer(s) features touch the boundary of the Source layer feature
SHARES_A_LINE_SEGMENT_WITH	Target layer(s) features share a line segment with the Source layer feature
CROSSED_BY_THE_OUTLINE_OF	Target layer(s) features are crossed by the outline of the Source layer feature
HAVE_THEIR_CENTER_IN	Target layer(s) features have their centroid in the Source layer feature

Table 1 - ArcGIS 2D Spatial Relationship Geoprocessing Tools

The prototype only supports the following prepositions: IN and ON. Therefore, a C# switch statement is used to select the appropriate filter relationship based upon the preposition. This switch statement can be modified to support additional topological relationships as functionality is expanded. For the time being, IN and ON are mapped to the CONTAINS topological relationship. ESRI exposes these spatial relationship types through the ISpatialFilter interface included as part of the ArcObjects SDK for .Net.

The geometries are first selected in the point layer and polygon layer using only the feature IDs contained within the 'phraseObject' class. If both point and polygon features are selected, the spatialFilter object is constructed using the geometries from the polygon features and intersected with the geometries from the point features. The results return only those point features that are completely contained by the polygon features. The dialog is updated with the information from the returned features and the map display is refreshed to highlight the point and polygon features that satisfied the topological relationship

In the event that the spatialFilter object cannot find any features that satisfy the topological relationship, a notification is presented to the user that indicates no features were found.

IV. Search Example

Point/Polygon Spatial Search

The following highlights how the system processes a spatial search in which the user inputs the following query string: 'Where are the Starbucks in Riverside?'

As previously described, the Query Processing Pipeline tokenizes the query string and processes the words using both the POS Tagger and the NER Tagger. 'Starbucks' is identified as a proper noun singular (NNP), 'in' is identified as a preposition/ subordinating conjunction (IN) and 'Riverside' is identified as a proper noun singular (NNP). The NER Tagger recognizes 'Starbucks' as an Organization and Riverside as a Location. Finally, the query string is identified as a spatial search because the POS Tagger located the preposition 'in.'

Next the string is divided into a source phrase and target phrase respectively. In this example, the source phrase is 'Where are the Starbucks' and the target phrase is 'Riverside.' Because Starbucks is recognized as an organization, the Lucene index for the POINT feature class is searched using the 'name' field. The Lucene index returns 141 records for Starbucks POINT features.

The only valid feature class for the target phrase is POLYGON; therefore and because Riverside was recognized as a Location, the Lucene index for the POLYGON feature class will be searched using the 'name' field. Only 1 hit is returned for the POLYGON feature class. The spatial preposition 'IN' maps to the CONTAINS topological relationship and is used with the ArcGIS spatialFilter class for both the currently selected POINT and POLYGON features.

Figure 12 depicts the OpenStreetMap tiles for Los Angeles, California and the NLP Search Form with the example search string entered.

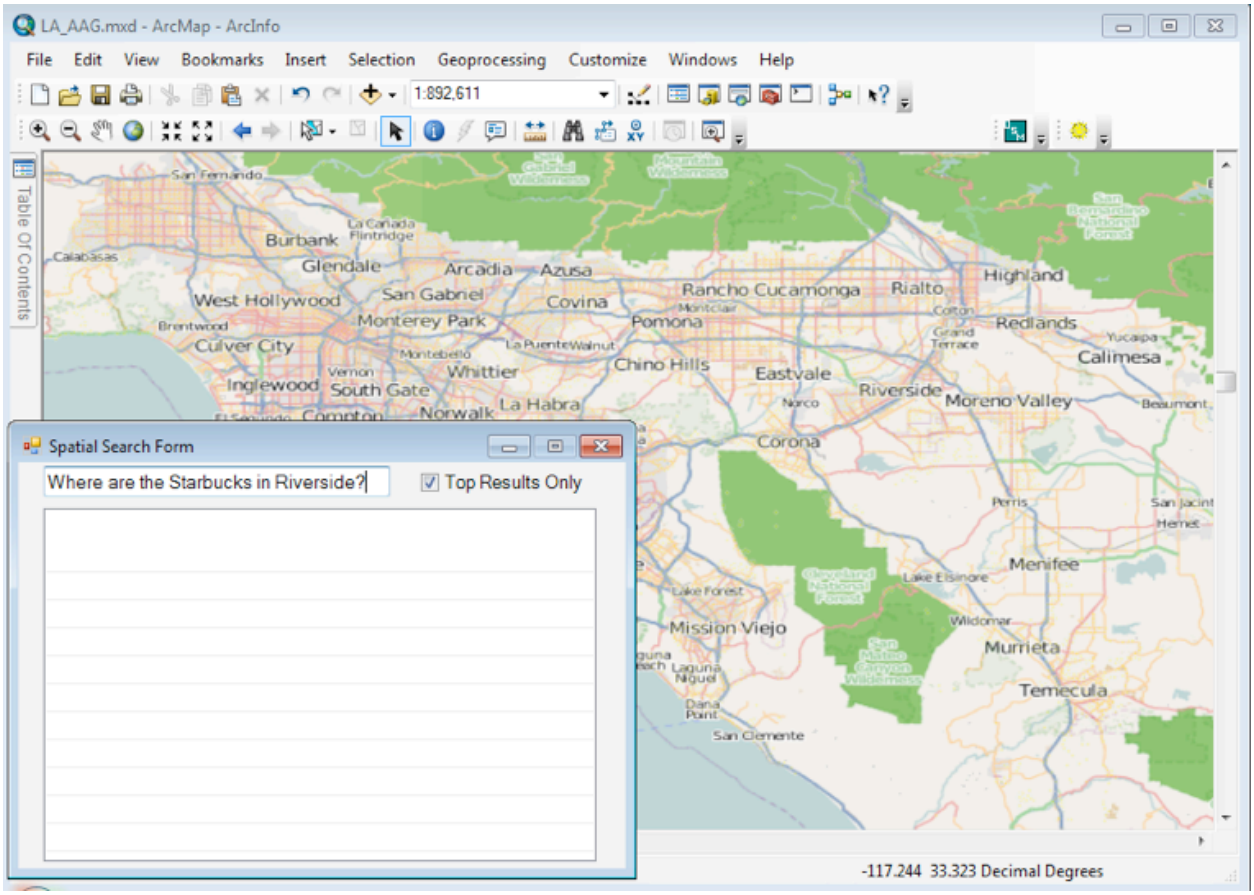


Figure 12 – OpenStreetMap Depicting Los Angeles, CA

When the user hits return after entering the search string, all Starbucks are selected as depicted in Figure 13.

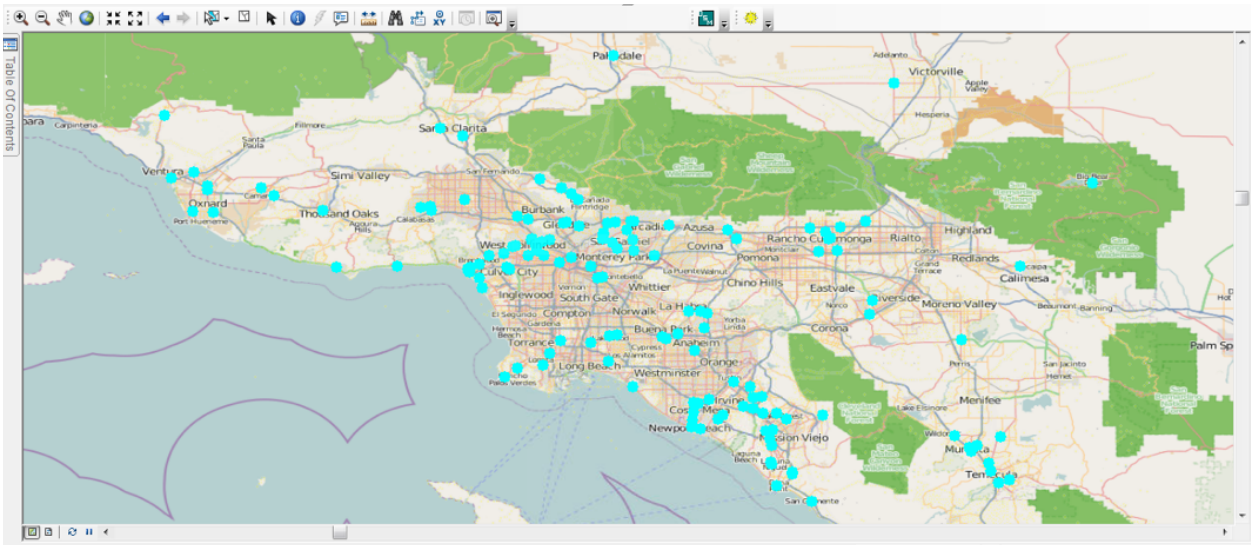


Figure 13 - OSM Point Features with Starbucks Cafes Highlighted

The Target Feature Class is searched using the target selection string as in “name Riverside.” This results in a single polygon feature being selected for the Riverside. Figure 14 depicts the selected Riverside polygon.

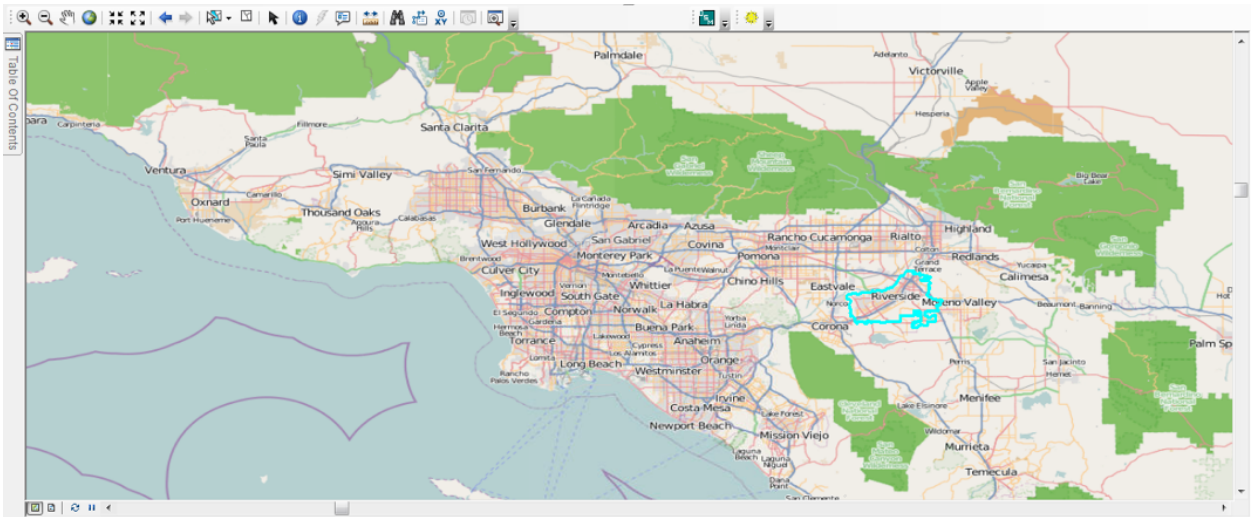


Figure 14 - OSM Polygon Features with the Riverside Boundary Highlighted

With both source and target features selected, the spatialFilter is applied using the 'CONTAINS' topological relationship type. The result of this spatialFilter operation is depicted in Figure 15 that shows the two Starbucks that lie inside the Riverside polygon. The NLP Search Application dialog window is also updated to show the feature attribute information for the two Starbucks highlighted on the map display.

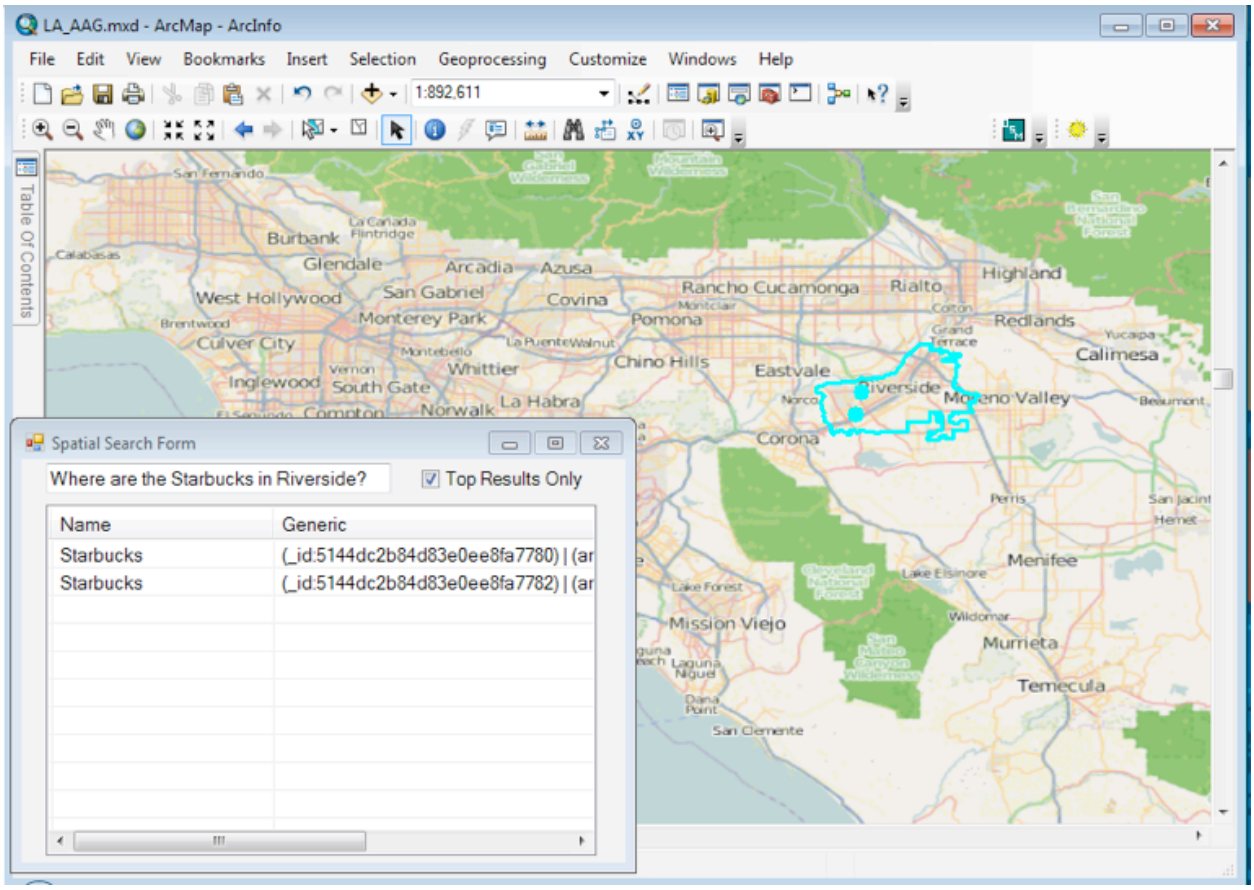


Figure 15 - Two Starbucks Located within the Riverside Boundary Polygon Feature

V. Conclusions

There are a number of conclusions that can be drawn from the development of this prototype framework. Perhaps the most significant is that NLP tools can be integrated into ArcGIS desktop to support GIS queries that are not possible in SQL. This was demonstrated through the simple example that performed a basic spatial search using a natural language query.

Furthermore, the prototype framework has reduced the number of steps required for a basic spatial search from more than 10 to 3. This greatly simplifies a basic spatial filter operation to the point where a casual user could interact with the framework.

Finally, the performance of the Lucene index was faster and more flexible than a generic 'like' SQL search. By exposing the index outside the geodatabase, a more generic interface can be written that does not require the ArcGIS SDK for basic feature searches. This also takes advantage of Lucene's inverted index architecture for fast information retrieval.

VI. Future Work

This paper describes a prototype capability that incorporates Natural Language Processing tools into the ArcGIS Desktop application. Beyond the expanded search capability, the prototype introduces a framework within which NLP functionality can be expanded to handle more complex query strings and a larger set of spatial prepositions.

Based upon Coventry and Garrod's preposition taxonomy, the next logical group of prepositions to support include those described as proximity terms such as near and far. The ambiguity of these prepositions makes it challenging to determine a standard distance when calculating a proximity between multiple features. However, the use of minimum bounding rectangles offer one possible solution in situations where the closeness of a point or line feature to a polygon feature is being tested. Point to point proximity will require a different method to set a distance by which one feature is determined to be near or far from another feature. In this case, some form of user interaction may be necessary.

The number of supported geometry types can also be expanded to include lines and polygons such that POLYGON/POLYGON, POINT/LINE and LINE/POLYGON relationships can be discovered. Determining when one polygon is 'in' another polygon can be implemented in the current framework through the use of temporary feature classes. These can be cloned from the basic OSM feature classes to support feature selection and the spatial filtering process.

Finally, integrating tools that support more robust sentence parsing should allow for fine-grained processing and a greater understanding of the query string when it contains multiple prepositions, descriptors and word dependencies. For example, an expanded application would need to analyze "Find all the Starbucks in Riverside that are near a gas station" and understand it contains two prepositions and that 'near a gas station' refers to 'Starbucks' and not 'Riverside.' Stanford, the NLTK and GATE have tools that can perform this type of sentence parsing. These tools could support more complex natural language searches and greatly enhance the prototype's utility.

VII. References

Hall, M. M., & Jones, C. B. (2008). Quantifying spatial prepositions. *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems GIS 08* (p. 1). ACM. Retrieved from http://dx.doi.org/10.1145/1463434.1463507

Garrod, S., & Coventry, K. (2004). *Saying, Seeing, and Acting: the Psychological Semantics of Spatial Prepositions*. *Essays in Cognitive Psychology series*. Psychology Press.

NLTK (2013). NLTK 2.0 Documentation. Retrieved from http://nltk.org/.

Stanford (2013). The Stanford Natural Language Processing Group CoreNLP. Retrieved from http://www-nlp.stanford.edu/software/corenlp.shtml.

GATE (2013). General Architecture for Text Engineering. Retrieved from http://gate.ac.uk.

OpenStreetMap (2013). Open Database License. Retrieved from http://wiki.openstreetmap.org/wiki/Open_Database_License

Four Square Blog (2012, February 29). Foursquare is joining the OpenStreetMap movement! Say hi to pretty new maps! Retrieved from http://blog.foursquare.com/2012/02/29/foursquare-is-joining-the-openstreetmap-movement-say-hi-to-pretty-new-maps/

Bennett, J. (2012, March 8). Welcome Apple. Retrieved from http://blog.osmfoundation.org/2012/03/08/welcome-apple/

Kordjamshidi, P., Hois, J., van Otterlo, M., Moens, M. Learning to Interpret Spatial Natural Language in Terms of Qualitative Spatial Relations (2011), (book chapter, to appear/submitted).

Jaiswal, A., Pezanowski, S., Mitra, P., Zhang, X., Xu, S., Turton, I., Klippel, A., MacEachren, A. (2011). GeoCAM: A geovisual analytics workspace to contextualize and interpret statements about movement. *Journal of Spatial Information Science*, *3*(3), 65-101. Retrieved from http://josis.org/index.php/josis/article/view/55.

MongoDB (2013). MongoDB. Retrieved from http://www.mongodb.org/.

Apache Lucene (2013). Welcome to Apache Lucene. Retrieved from http://lucene.apache.org/.

MacEachren, A. , Jaiswal A., Robinson A., Pezanowski S., Savelyev A., Mitra P., Zhang X., Blanford J. (2011). SensePlace2: GeoTwitter Analytics Support for Situational Awareness. IEEE Conference on Visual Analytics Science and Technology [Internet]. Retrieved from: http://www.geovista.psu.edu/publications/2011/MacEachren_VAST_2011_reducedsize.pdf

Toutanova, K., Klein, D., Manning, C., and Singer, Y. (2003). Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of HLT-NAACL 2003*, pp. 252-259.

IKVM (2013). IKVM.Net Home Page. Retrieved from http://www.ikvm.net/.

Marcus, M. P., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, *19*(2), 313-330. Retrieved from http://citeseer.ist.psu.edu/marcus04building.html

Finkel, J., Grenager, T, Manning, C. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005),* pp. 363-370.http://nlp.stanford.edu/~manning/papers/gibbscrf3.pdf

OpenStreetMap Wiki (2012). Nominatim. Retrieved from http://wiki.openstreetmap.org/wiki/Nominatim

ESRI ArcGIS 10.0 Help: Select by location- graphic examples (2012). ESRI, Redlands, California. Retrieved from http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//0017000000tp000000