A Data Mining Framework for Analysing Geospatial-Temporal Data

A Thesis

Presented to

The Faculty of the College of Graduate Studies

Lamar University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Raunak Sarbajna

December 2019

A Data Mining Framework for Analysing Geospatial-Temporal Data

RAUNAK SARBAJNA

Approved:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sujing Wang

Committee Chair

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Stefan Andrei

Committee Member

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lawrence J. Osborne

Committee Member

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Stefan Andrei

Chair, Department of Computer Science

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lynn M. Maurer,

Dean, College of Arts and Sciences

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

William E. Harn

Dean, College of Graduate Studies

©2019 Raunak Sarbajna

No part of this work may be reproduced without permission except as indicated by the “Fair Use” clause of the copyright law. Passages, image, or ideas taken from this work must be properly credited in any written or published materials.

ABSTRACT

A Data Mining Framework for Analysing Geospatial-Temporal Data

by

Raunak Sarbajna

Change analysis and automated storytelling are essential techniques in understanding patterns and trends in multifaceted, time-series geospatial-temporal data. In this thesis, we introduce a novel data mining framework for detecting and analysing how the patterns of features change over time and space in spatio-temporal data and automatically generate interesting stories based on a variety of unique factors. We discuss and develop techniques to generate appropriate spatial polygons from raw georeferenced point data while preserving its meta data and shape. Our framework utilizes appropriate change predicates that are data agnostic and can capture the widest possible polygon movements. We then develop and use new measures of interestingness to aid in generating automated stories based on the change analysis results. Our change monitoring framework then creates a change graph that captures the changes in the spatial clusters and our change summarization framework creates specific change summaries based on the change graph types. We evaluate the effectiveness of our framework through case studies involving 2016 Twitter data with sentiment scores attached and 2017 drought data from North America. The experimental results show that our framework can discover interesting change patterns and useful information from spatial-temporal data.

ACKNOWLEDGEMENTS

I would like to thank Dr Sujing Wang for her patience and dedication as I spent the better part of a year working on this framework. I also very much appreciate the support from Dr. Stefan Andrei throughout my time. Dr. Lawrence J. Osborne provided me with some much-needed perspective during the year. Dr. Christopher Eick and his team at University of Houston gave me guidance at crucial periods when I was stuck. Last, but not the least, my family’s constant support throughout even though are thousands of miles away, kept me working. Thank you, Ma, Baba, Pishi and Thamma!

Table of Contents

List of Tables v

List of Figures vi

Chapter Page

1 Introduction 1

Analytical Methods for Spatio-Temporal Analysis 3

Research Contribution 4

2 Literature Review 5

3 Methodology 11

Change Analysis Predicates 11

Polygon Generation from Point Data Sources 12

Change Analysis for Polygonal Map Data Sources 14

Storytelling Techniques for Geospatial Data 17

4 Case Study 19

Data Sets 19

Dataset Preprocessing 22

Drought Data Sources 27

Emotion Spatial Clusters from Twitter 32

Analysis 34

5 Conclusion and Future Work 36

Bibliography 37

Appendices 42

Appendix A 43

Appendix B 45

Appendix C 48

List of Tables

Table Page

Table 1. Layer Specification for Drought Data 21

Table 2. Layer Specification for Twitter Data 21

List of Figures

Figure Page

Figure 1. Change Analysis framework Architecture 16

Figure 2. Spatial reference for Twitter Data 19

Figure 3. Spatial reference for Drought Data 20

Figure 4. United States Drought Monitor Data 20

Figure 5. Twitter shapefile data inserted into Postgres db 22

Figure 6. Emotion polygons created from June 5 Twitter data 25

Figure 7. Emotion polygons created from June 6 Twitter data 25

Figure 8. Emotion polygons created from June 7 Twitter data 26

Figure 9. Emotion polygons created from June 1 Twitter data

Using Convex Hulls 26

Figure 10. Continental USA original monthly drought polygon

distribution for 2017 27

Figure 11. Increase in Drought Affected Regions in Texas

following Hurricane Harvey 29

Figure 12. Decrease in Drought Affected Regions in Texas

following Hurricane Harvey 30

Figure 13. Shifting areas of drought affected regions in California

following the 2017 wildfires 32

Figure 14. Change in Twitter emotion polygons from June 5 to June 6 33

Figure 15. Change in Twitter emotion polygons from June 7 to June 7 34

Chapter 1

Introduction

Analyzing change in spatial-temporal data is critical for many applications including developing early warning systems that monitor environmental conditions, detecting political unrest and crime monitoring. Change analysis models are essential in understanding larger patterns and trends in multifaceted, time-series geographic data. The purpose of this study is to detect spatio-temporal changes within sequential (time-series) geospatial-temporal data.

Generating a knowledge base from the vast quantities of geospatial data requires change detection capabilities that are beyond what a human can grasp (Allerin 2017). The minuscule details needed to distinguish and detect differences in real world structures as different as, say, number of parked cars at retail outlets, or reconstructions on buildings, or small changes in the angle of incidence of a satellite dish, are practically impossible for the human eye to detect. Even when the changes are massive or notable enough to be observed by the human eye, given the immense volume of geospatial it is neither cost effective nor labour efficient to have humans pore over LiDAR or satellite imagery to constantly keep looking. Without change detection software that is both smart and efficient, most of this data is going to waste.

A common understanding is that most big data available today is either archival, media or web scrapes. However, a large source of that data is from Geographic Information Systems (GIS), and the tools available to interpret them easily are lacking (Geomenke 2014). The multiplicity of APIs has standardized access and structuring, but they limit much of the meta-data associated with them. Most publicly available (i.e. non-governmental) 'big data' sources with spatial components revolve around data scraped from mobile software platforms, including twitter, Instagram, Snapchat and reviews on mapping apps. The recent API changes by Google Maps, for instance, aptly show that data content, and thus its meaning, is subject to regulation that is outside the control of researchers (Thatcher 2014). Thus, the goal of this research project is to detect and analyze how the patterns of features change over time and space in spatio-temporal data and then automatically generate stories based on the sequential spatio-temporal data.

*Analytical Methods for Spatio-Temporal Analysis*

Analytical methods for spatial and spatio-temporal analysis have developed considerable recently. Most of the newer techniques capture only spatial correlation or covariance structures over polygonal lattice data, and there are few methods involving polygon, point and line spatial data. For most remotely sensed imagery, the phenomena are represented as vector points, lines, polygonal models or raster surfaces. Polygonal models classify objects, while rasters classify fields. However, many situations cannot be classified and mapped using such categories, because they contain elements of both, such as, wildfires, hurricanes, droughts or ecological hotspots. Methods for characterizing space-time change in such phenomena are currently poorly developed (Jin et al. 2013).

Spatial relationships between multiple polygons are not very well defined. The most widely used models of spatial relations is the point set topology based 9-intersection model by Egenhofer and Herring. The model has been further refined by Chen et al. using Voronoi diagrams of spatial objects. The modern approach combines the 9-intersection model and metric parameters, as proposed by Liu and Deng. However, the complexity of spatial relationships between polygons makes characterizing them using topological relations defined by point-set topology a very challenging analytical and computational task.

The basic metrics of spatial relations between moving polygons are useful to describe changes in edges or centroids. Over a larger time period, changes in distance (related to velocity) and direction combine to describe the underlying phenomena with some degree of certainty. However, an issue arises in quantifying distance and directional relationships where polygons both shift and change in size (Robertson et al. 2007). This is where polygons as objects and rasters as fields are combined to get finer metrics for both. In this case, most real-world polygons are attached metadata that serve the same purpose.

The digital algorithms for change detection work by comparing unclassified imagery on a pixel-by-pixel or patch-by-patch basis. For classified images we can only observe change in specific classes over time. However, even if we do use an automated algorithm, we usually start by manually comparing /co-referenced images from the two dates. Any image processing tool will have facilities to swipe one image over the other, flicker between images, and view images side-by-side and does not require specialized georeferencing software. While digitizing the images into chloropleth maps is our end goal to identify and classify the change, manual visual inspection is always needed to select the most appropriate automated change detection technique (Kumar et al. 2008).

*Research Contribution*

Our approach provides a change monitoring framework which creates a change graph that captures the changes in spatial land uses clusters and a change summarization framework that creates specific change summaries based on the change graph based on the change story types.

Our research contributions are summarized as follows:

1. A novel change analysis framework that works on vectored polygonal datasets
2. New change predicates that are data agnostic and can work on a large spectrum of data
3. A new measure of interestingness to aid in generating automated storytelling based the change analysis results

The rest if the thesis is structured as follows. Chapter 2 reviews previous literature on the subject and discusses related work. Chapter 3 introduces our data mining framework and lays out the methodology in detail. Chapter 4 evaluates the framework with case studies on drought datasets pre and post Harvey, pre and post California wildfires and twitter emotion polygons. Chapter 5 provides a conclusion and discusses potential future expansions to the framework.

Chapter 2

Literature Review

Liu and Deng have created the model for modern metric based spatial relationships between polygons (Liu et al. 2012). They updated the original 9-intersection model by Egenhofer and Herring (Egenhofer and Herring 2019), eliminating limitations. Liu and Deng’s metric spaces avoid problems with the immensity of the spatial object complement operation, restricts sensitivity of boundary data errors and makes representing dynamic topological relations easier. They introduce geometrical metric parameters of point sets, such as their distance, area and the circumference. This is divided into segmentation parameters and nearness parameters. The finals steps are (1) to find and locate spatial objects, (2) to make a approximate classification for spatial relations between a selected reference object and a target one using the 9-intersection model, and (3) to identify the spatial relations in more detail by applying metric parameters.

Robertson et al. coined the term moving polygons for spatio temporal polygon data (STAMP) (Robertson et al. 2007). They developed the original work by Sadahiro and Umemura and improved upon the original event-based framework. They first grouped polygons based on factors such as distance thresholds and intersection points. These groups are then evaluated according to three event types and in four hierarchical levels. The typology of events to describe these geometric changes include Generation, Disappearance, Expansion and Contraction. These are modified as needed according to the dataset, with movement events such as Displacement, Convergence, Fragmentation and Divergence.

A survey of the classical change detection algorithms can be found in the Lu et al. paper (Lu, Brondízio, and Moran 2004) and tells us that the integrated GIS and remote sensing approaches yield the best results. However, they are very sensitive to registration accuracies between images. Thus, images must be properly orthorectified and georeferenced, especially because the changes in the emotion polygons are so subtle. This assumes the emotions are to be treated as just another feature in the map, like any other category.

The ability to develop spatially distributed models of topographic change is presenting new capabilities in geomorphic research, as seen in James et al. High-resolution maps of elevation change indicate locations, processes, and rates of geomorphic change, and provide a means of calibrating temporal simulation models. Methods of geomorphic change detection (GCD), based on gridded models, may be applied to a wide range of time periods by utilizing cartometric, remote sensing, or ground-based topographic survey data to measure volumetric change. Advantages and limitations of historical DEM reconstruction methods are reviewed with a focus on coupling them with subsequent DEMs to construct DEMs of difference (DoD), which can be created by subtracting one elevation model from another, to map erosion, deposition, and volumetric change.

Since our data is primarily in an urban environment, with all the grid like rigidity that entails, it is a good idea to look at change detection algorithms optimized for urban environments. One of the hardest aspects to measure is to distinguish between change and no-change, as well as different kinds of change. Comparing image differencing, image regression, tasselled-cap transformation and chi square transformation, Ridd and Liu found image differencing to be the most consistent, with a sustained overall accuracy of >80% (Ridd and Liu 1998).

It is useful to have a programming-oriented study comparing several of the change detection algorithms using MATLAB, rather than pure application-oriented comparison, in order to have a benchmark. Minu and Shetty analyzed image differencing, image ratioing, change vector analysis, tasseled cap transformation and principal component analysis for efficiency and effectiveness. Although their area of study was not urban but a variety of land use/ land cover, change vector analysis gave the best overall accuracy (Minu and Shetty 2015).

We also studied two novel methods that are recent developments and are showing promising results: Neighborhood Correlation Image and Comprehensive Change Detection Method, both of which are optimized for remote sensing imagery but can be adapted to vectorized maps without loss of generality.

The change detection model using Neighborhood Correlation Image (NCI) logic works because of the obvious fact that the same geographic area (e.g., a 3x3 pixel window) on two dates of imagery will tend to be highly correlated if little change has occurred, and uncorrelated when change occurs. Computing the piecewise correlation between two data sets demonstrates that NCIs contain change information and that NCIs may be powerful tools for change detection (Im and Jensen 2005).

A high-performance remote sensing method called Comprehensive Change Detection Method (CCDM) integrates spectral-based change detection algorithms and a novel change model called Zone, which extracts change information from two Landsat image pairs. This can be easily modified to work on the Twitter-based emotional grading maps. This method is simple, easy to operate, widely applicable, and capable of capturing anthropogenic changes like our area of interest (Jin et al. 2013).

Storytelling techniques are an effective summarization method to succinctly organize extensive information. Traditional storytelling has been mostly successful on news articles, blogs, as well as structured databases. However, traditional storytelling techniques tend to perform poorly on social media content, such as Twitter, where text lacks proper form and function. Moreover, the ability to support dynamic storylines as they evolve is critical to modelling fast moving social media streams such as Twitter. Dos Santos et al. introduced a set of methods to automatically derive stories over linked entities in tweets. They model a story as a graph of entities propagating through spatial regions in a temporal sequence, and controls search space complexity by suggesting regions of exploration. They developed algorithms to conduct storytelling to model tweets over space and time, reasoning over spatio-temporal features, and devise spatio-temporal storylines based on connectivity strength (Dos Santos et al. 2015).

Kumar et al. formulated storytelling as a generalization of redescription mining. Stories are defined as chains of redescriptions. They proposed an efficient storytelling algorithm as A\* search around the outputs of a CARTwhells redescription mining algorithm (Ramakrishnan et al. 2004). The efficiency and scalability of the proposed algorithm were evaluated by three application case studies: word overlaps in large English dictionaries, exploring connections between gene sets in a bioinformatics data set, and relating publications in the PubMed index of abstracts. They proposed an efficient storytelling implementation that embeds the Cartwheels redescription mining algorithm which utilizes induced classification trees to model redescriptions in an A\* search procedure, using the CARTwheels to supply next move operators on search branches to the A\* search procedure (Kumar et al. 2008). Vocht et al. proposed the implementation of an optimized algorithm controlling the pathfinding process to obtain more homogeneous search domain and retrieve more links between adjacent hops in each path to improve the semantic relatedness of concepts mentioned in a story by increasing the relevance of links between nodes through additional domain delineation and refinement steps (Vocht et al. 2015).

Chen et al. proposed a multimodal imitation learning via generative adversarial networks (MIL-GAN) method (Chen et al. 2017) to directly model users' interests as reflected by various data by imitating users' demonstrated storylines. MIL-GAN model is designed to learn the reward patterns given user-provided storylines and then applies the learned policy to unseen data.

Santos et al. introduced three methods of association analysis, Distance-based Bayesian Inference, Spatial Association Index, and Spatio-logical inference, to capture relatedness among real-world events in high data volumes, and to model similar events that are described disparately under high data variability. It takes as input a set of geotemporally-encoded text streams about violent events called “storylines”. This study demonstrated that spatio-temporal storytelling is able to capture important associations among violent events reported in social media and traditional datasets (Santos et al. 2014). They combined storytelling and Spatio-logical Inference (SLI) to generate rules of interaction among entities and measure how well they forecast a real-world event.

Hossain et al. introduced Google Fusion Tables (GFT) that offers collaborative data management in the cloud for data scientists to enable the integration of increasingly complex geospatial data to support storytelling. The paper focused on introduction of overview of map processing in GFT, the architecture overview of GFT, and how to scale to large datasets, massive and complex polygon datasets. GFT provides a useful tool for storytelling through interactive maps. They proposed an approach to automatically construct stories between entities in large document collections that can help from directed chains of relationships, with support for co-referencing, evidence marshaling, and imposing syntactic constrains on the story generation process. A new optimization technique based on concept lattice mining is used to rapidly construct stories on massive datasets (Hossain et al. 2012).

Chapter 3

Methodology

*Change Analysis Predicates*

We begin by creating our change predicates. There are three change functions that are necessary for complete change predicates, we define them as follows:

Here, is a spatial cluster polygon from the batch timed and is another spatial cluster polygon from a batch timed , with indicating a time increase. We denote cluster intersection as and union as . denotes the area covered by the polygon , upto a desired level of accuracy.

The change function is a measure of similarity. measures whether individual polygons coincide. To measure concurrence between one polygon and a set of polygons, we use the functions. Here are spatial clusters from a batch , and is a cluster from a different batch . Other basic functions include , which indicates the Centroid of a polygon as defined as:

…(1)

… (2)

where and A = the area of the polygon. We also use a simple Manhattan distance function which is tuned to the required georeferenced accuracy. We use all these functions to define these seven change predicates:

Implementations of these change predicates can be found in the Chapter 4.

*Polygon Generation from Point Data Sources*

We improve upon the change analysis framework called Aconcagua (Elgarroussi et al. 2018). The system expects an input of emotion polygons annotated with emotion assessment scores, with +1 representing a very high positive emotion and -1 representing a very high negative emotion. While this method does lead to inconsistencies in locations due to georeferencing inaccuracies, we find that the 8 ~ 10m precision works well within city limits.

There are several ways of using the numerous point data we have obtained from this step into an actual polygonal map:

1. Creating closed contour lines for contour lines that lie on the boundary of the observation area.
2. Creating a convex hull from points with similar scores.
3. Creating alpha-shapes or general concave hulls by selecting the minimal Delauney triangulation of the points.

We now elaborate on creating these polygons.

For contour lines, we start by selecting all point objects belonging to the same emotion group. We pick a random point within a group and connect all neighboring spots by straight lines, where ‘neighbor’ is defined somewhat arbitrarily according to close distances. We then move on to another point and restart the process. The results after all points are connected is a triangulated irregular network (TIN). All lines are then marked using an equal interval classification scheme, where the contour interval is chosen depending on the size of the polygon under consideration. Then, selecting an arbitrary point in the TIN, contour lines are threaded through ticks of equal value.

A convex hull can be though of as the smallest convex geometry that encloses all the points considered. If the group of points forms a line or is only a single point then the convex hull is considered a degenerate and is ignored. Otherwise convex hulls are very fast to calculate using the Graham Scan algorithm, as the points are already sorted by latitude and longitude. The algorithm stars at the South Western most point and moves Eastward. For every coordinate triplet, the procedure checks whether or not it’s a concave corner. If it is, then the middle point cannot lie on the convex hull. This cross product calculation is also very fast. Convex hulls are widely used and are implemented in most common GIS systems. The shapes formed by a convex hull are however necessarily blocky and do not reflect natural distributions.

For a generalization of the convex hull, the concave hull is used. The most common method of concave hulls is Alpha Shapes. Alpha shapes usually look like generalized and more optimal versions of contour interpolations. Alpha shapes are very strong at formalizing the shape of a spatial point data set, unlike convex hulls. They area subgraph of the Delauney triangulation of the entire geometry. Given a set of points, the Delauney triangulation can generate a large family of shapes. We add the parameter ‘alpha’ to control the level of detail. The alpha shape degenerates to the point set for alpha tending to zero, and a large value of alpha begins to approximate the convex hull. There exists a generalization of the alpha shape called weighted alpha shape, where we replace the Euclidean distance between points by the power to weighted points.

*Change Analysis for Polygonal Map Data Sources*

Our approach uses three primary set operations: union, intersection and erase. We calculate the area of each individual polygon within each map layer. We then execute a union operation and calculate area. The union layer now contains the original areas of both layers and the areas of the overlapping polygons - we now need to query them properly to prepare for calculating the change percentage and tabulating intersection.

To outline the polygon, we examine several different methods:

1. We find features common to either of the layers but not both, essentially performing a symmetrical difference
2. We erase the larger of the polygons from the smaller, thus retaining only the growth, and do vice-verse for shrinking
3. We perform simple intersection and then invert selection to get changed regions.

We then use these techniques:

1. **Data Pre-Processing**: This involves curation of datasets with obvious georeferencing errors. This would preferable be done by minimizing the root mean square error.
2. **Parametrization of polygons**: Calculate shape and area parameters for each individual polygon with each map layer.
3. **Analysis through Symmetrical Difference**: Extract features common to either of the layers.
4. **Polygon Union Computation**: Union sequential layers to contain the original areas of both layers and the areas of overlapping polygons.
5. **Polygon Erase Operation**: Erase larger polygons from smaller (or vice-versa) for detection of growth/shrinking.
6. **Polygon Intersection/Invert**: Selecting and then labelling the changed regions.

The framework can be found in Figure 1.

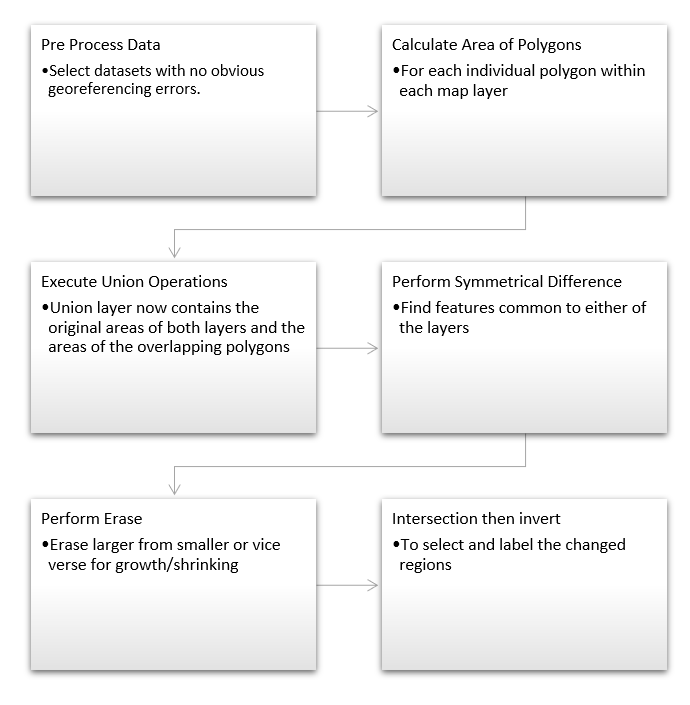


Figure 1: Change Analysis framework Architecture

*Storytelling Techniques for Geospatial Data*

In order to tell a coherent spatio-temporal data story from the change analysis output, we propose using an interestingness function for this task (Madhavan et al. 2013).

We choose an area of interest which is a rectangle bounded by coordinates and . The length of the diagonal is . We choose a set of polygons SCP, whose centroid lies within .

The Thinness ratio measure takes a maximum value of 1 for a circle. Objects of regular shape have a higher thinness ratio than similar irregular ones. (Wu, Merchant, and Castleman 2010, 201). We define the thinness ratio of a polygon as:

… (3)

It not only contains those polygons but also their associated characteristics. For example, a SCP could contain a set of spatial clusters represented by polygons, their average drought score, total area of each polygon, centroid coordinates of each polygon and other summaries for each spatial cluster (polygon). We define the function as:

… (4)

We define a threshold , which ensures that a narrative will only be generated an object such that . The parameters for are:

The parameters and are special cases. is to be used for those SCP whose primary change was growth/shrinking, henceforth referred to as . is to be used for those SCP whose primary change was shifting, henceforth referred to as . The threshold parameters need to be finely tuned so as to not exclude those polygons who fall through exceptions. Once we have a suitable selection of polygons and chose a threshold value, we can create a summary narrative based on that. (Huang and An and Cercone 2002)

Thus, our Interestingness function is hence calculated as:

… (5)

… (6)

Chapter 4

Case Study

*Data Sets*

We use two datasets for our case study:

1. Drought dataset from the North American Drought Portal.
2. Twitter posts from the United states.

The Twitter data contains the timestamp, latitude, longitude and the text of each tweet, which are then processed and tokenized (Banerjee et al. 2019). The spatial reference can be seen in Figure 2 and the layer specification in Table 2 (Person 2016).

The layer specification for the Drought shapefiles is summarized in Table 1. the spatial reference for the Drought shapefiles can be seen in Figure 3. The original structure of the drought monitor data can be seen in figure 4 (NOAA 2019).

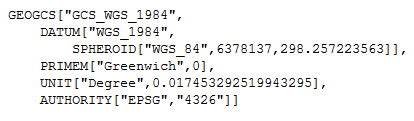


Figure 2: Spatial reference for Twitter Data

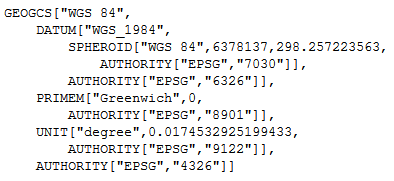


Figure 3: Spatial reference for Drought Data

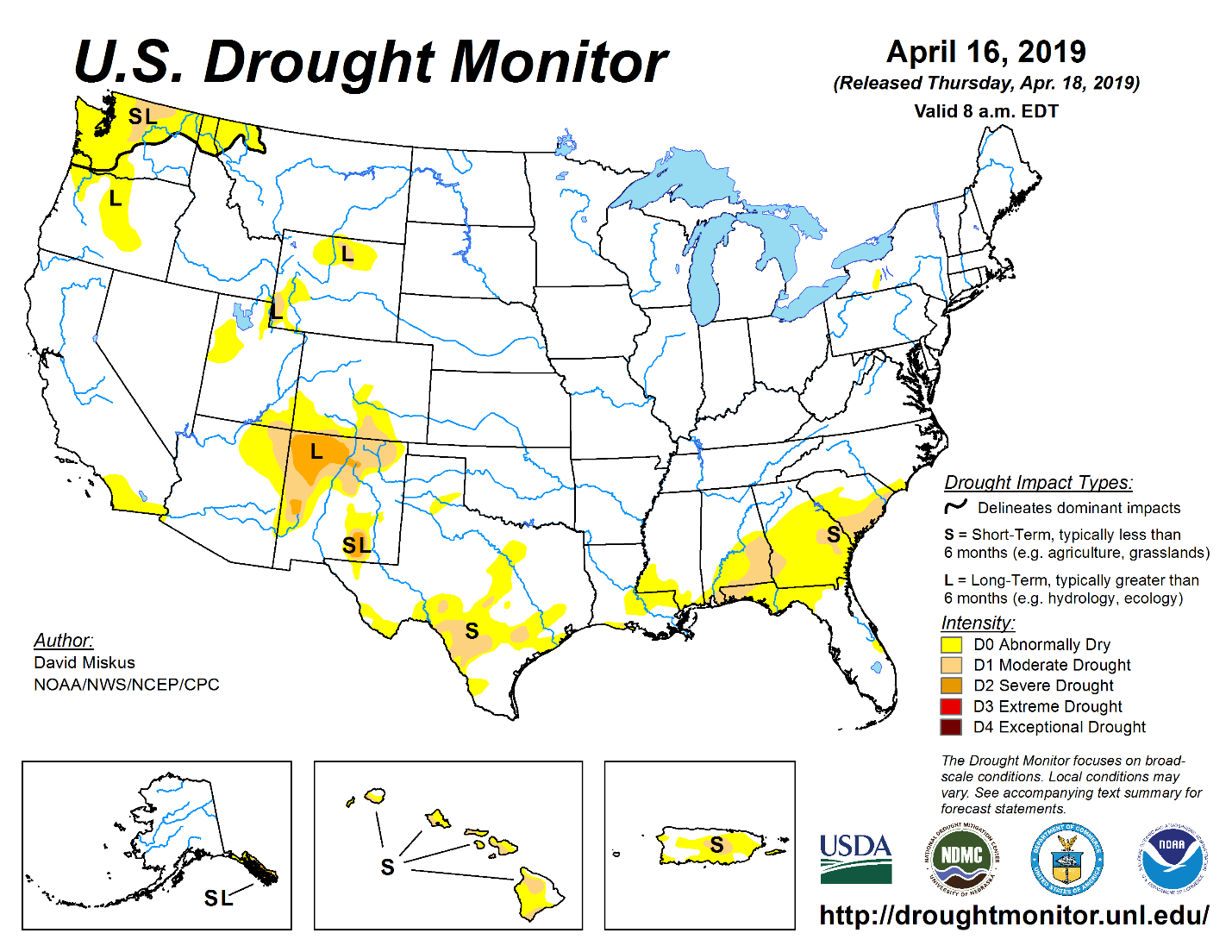


Figure 4: United States Drought Monitor Data (reproduced with permission)

Table 1: Layer Specification for Drought Data

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Width | Precision |
| Long | Real | 24 | 15 |
| Lat | Real | 24 | 15 |
| Id | Integer | 9 | 0 |
| dnstyTh | Real | 24 | 15 |
| avgScor | Real | 24 | 15 |
| numTwts | Integer | 9 | 0 |
| stdDev | Real | 24 | 15 |
| batchNm | Integer | 9 | 0 |
| geodata | String | 80 | 0 |

Table 2: Layer Specification for Twitter Data

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Width | Precision |
| FIPS\_ADMIN | String | 4 | 0 |
| GMI\_ADMIN | String | 7 | 0 |
| ADMIN\_NAME | String | 42 | 0 |
| FIPS\_CNTRY | String | 2 | 0 |
| GMI\_CNTRY | String | 3 | 0 |
| CNTRY\_NAME | String | 40 | 0 |
| POP\_ADMIN | Integer | 9 | 0 |
| TYPE\_ENG | String | 26 | 0 |
| TYPE\_LOC | String | 50 | 0 |
| SQKM | Real | 16 | 2 |
| SQMI | Real | 16 | 2 |
| COLOR\_MAP | String | 2 | 0 |

*Dataset Preprocessing*

Our initial approach to this problem was to store all shapefiles of the Twitter dataset in a postgres database with a GIS addon and perform operations in python. We used psycopg2 and osgeo libraries to import, process and visualize maps. However, this led to many problems with interconversions between georeferencing schemes, while converting from WKT geometry to PostGIS geography.

We start with basic point data, which contains latitude/longitude, along with metadata identifying value of interest, whether that is drought level or emotion value. We insert the contents of the shapefile into a PostGreSQL database using the shp2pgsql toolkit that comes along with the PostGIS extension. The results can be seen in Figure 5.

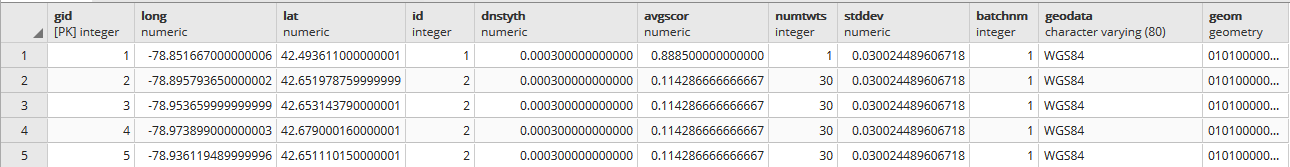


Figure 5: Twitter shapefile data inserted into Postgres db

As our framework relies on using polygonal data, we cannot use this. So, we begin by creating a convex hull and alpha shapes of points and inserting it into a new table, while preserving related meta-data. Due to engine limitations, we need to be careful and insert only those convex hulls or alpha shapes that are polygons specifically, not points or lines.

Our query to generate concave hulls were:

INSERT INTO public.june1poly (avgscor, numtwts, geodata, id, batchnm, geom)

(SELECT d.avgscor, d.numtwts, d.geodata, d.id, d.batchnm, ST\_ConvexHull (ST\_Collect(d.geom))

FROM public."2014-06-01 " AS d

GROUP BY (d.id, d.avgscor, d.numtwts,d.geodata, d.batchnm)

HAVING ST\_GeometryType(ST\_ConvexHull (ST\_Collect(d.geom))) = 'ST\_Polygon')

For concave hulls, we needed to select a shrink factor target percentage between 0.5 and 0.99 depending upon the geometry in question. We do not allow for polygons with holes due to problematic edge cases. Our query for this is as follows:

INSERT INTO public.june1poly (avgscor, numtwts, geodata, id, batchnm, geom)

(SELECT d.avgscor, d.numtwts, d.geodata, d.id, d.batchnm, ST\_ConcaveHull (ST\_Collect(d.geom), 0.75)

FROM public."2014-06-01 " AS d

GROUP BY (d.id, d.avgscor, d.numtwts,d.geodata, d.batchnm)

HAVING ST\_GeometryType(ST\_ConcaveHull (ST\_Collect(d.geom), 0.75)) = 'ST\_Polygon')

Following this, we need to add some meta information into the shapefiles, such as centroid, perimeter lengths and area. The query for each of them are as follows:

1. Centroid:

UPDATE public.june1poly

SET centroid=ST\_Centroid(geom)

1. Area:

ALTER TABLE public.june1poly ADD COLUMN IF NOT EXISTS area double precision

UPDATE public.june1poly SET area=ROUND((ST\_Area(geom::geography))::numeric,2)

1. Perimeter

SELECT ST\_Perimeter(j2s.geom, true) FROM public.june1poly j2s

We repeat variations of this process for every shapefile. For the exact code to run each query, see Appendix A. Figures 6, 7 and 8 show the results of our polygon generation by using concave shapes from what was previously point data. Figure 7, especially, shows interesting patterns in how much more naturalistic the polygon is, as compared to Figure 9, which was generated without using alpha shapes.

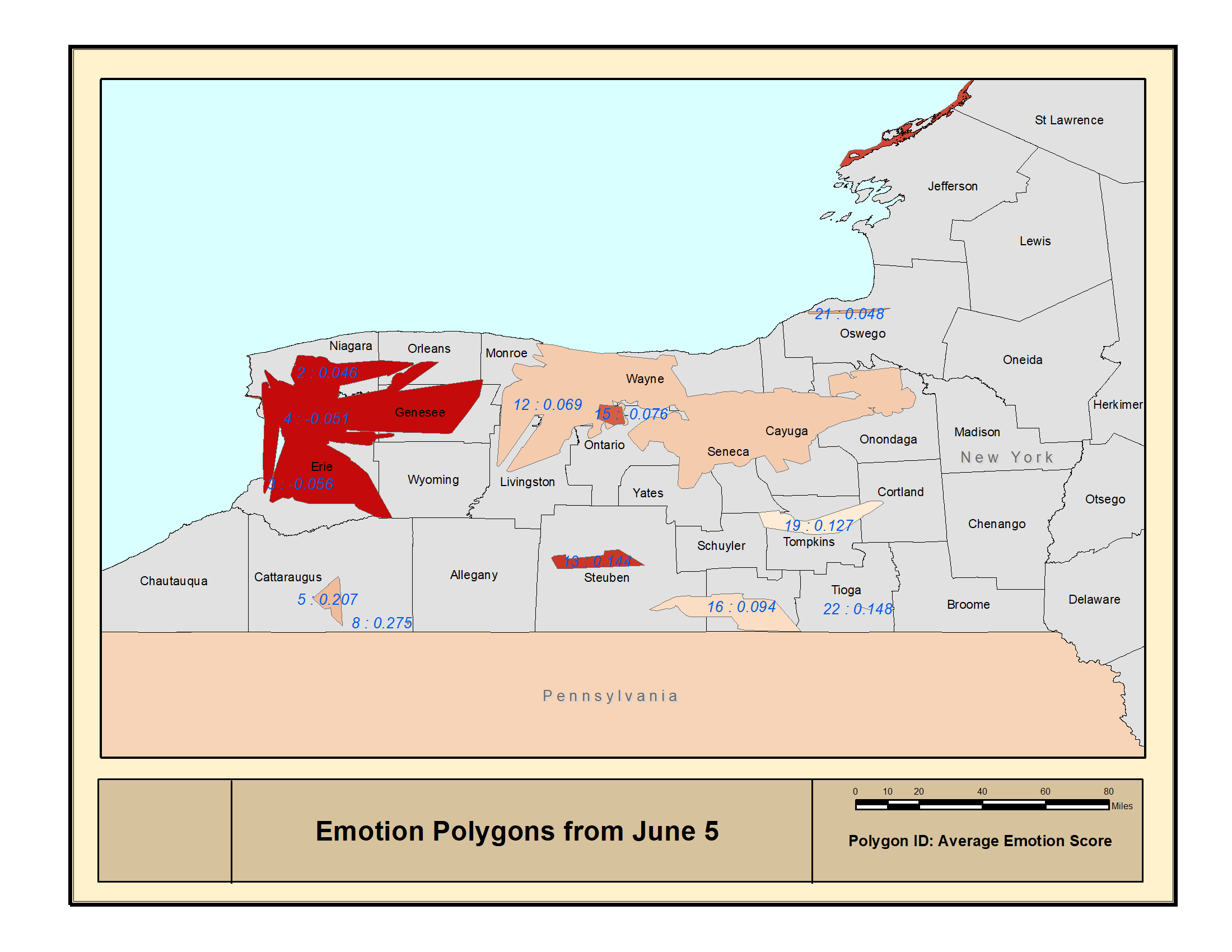


Figure 6: Emotion polygons created from June 5 Twitter data

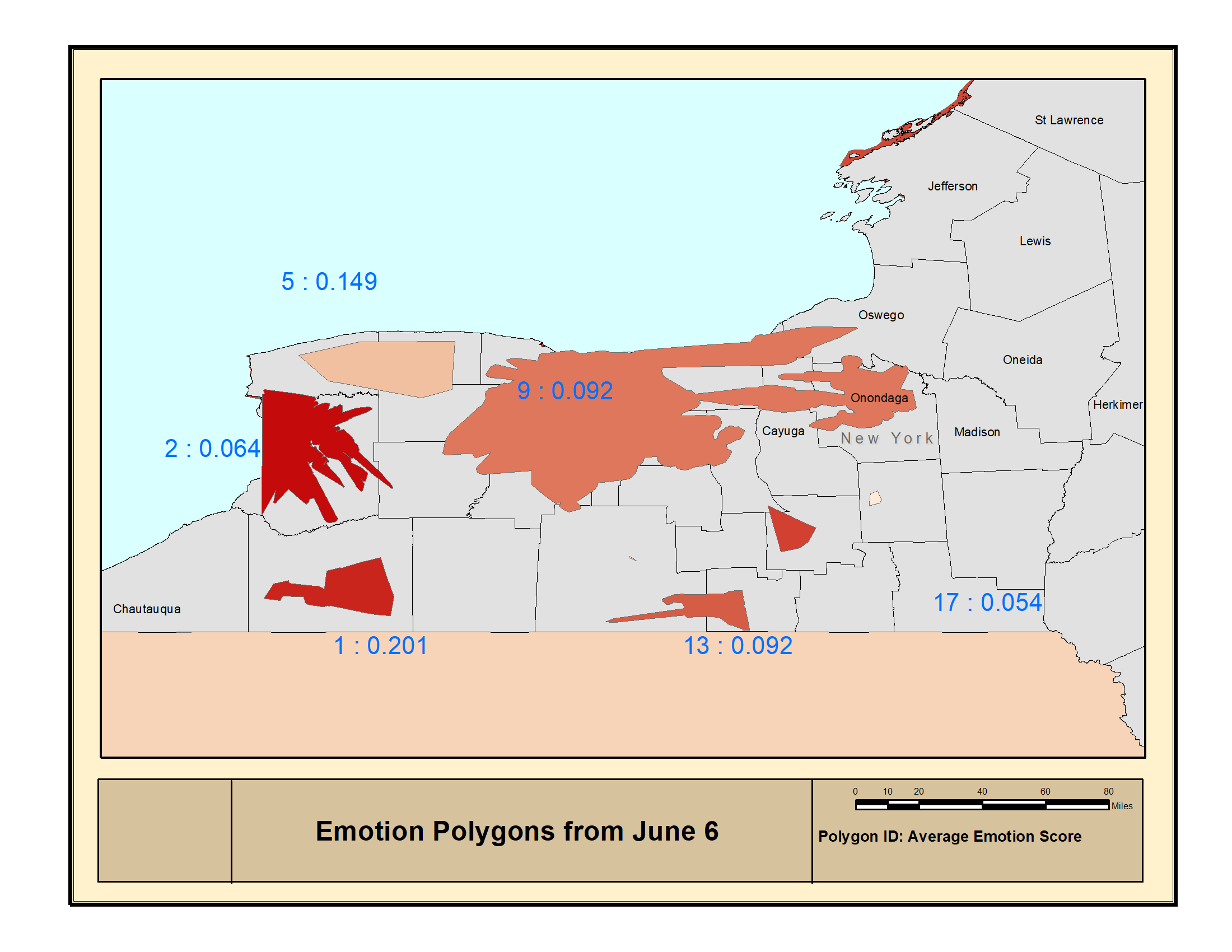


Figure 7: Emotion polygons created from June 6 Twitter data

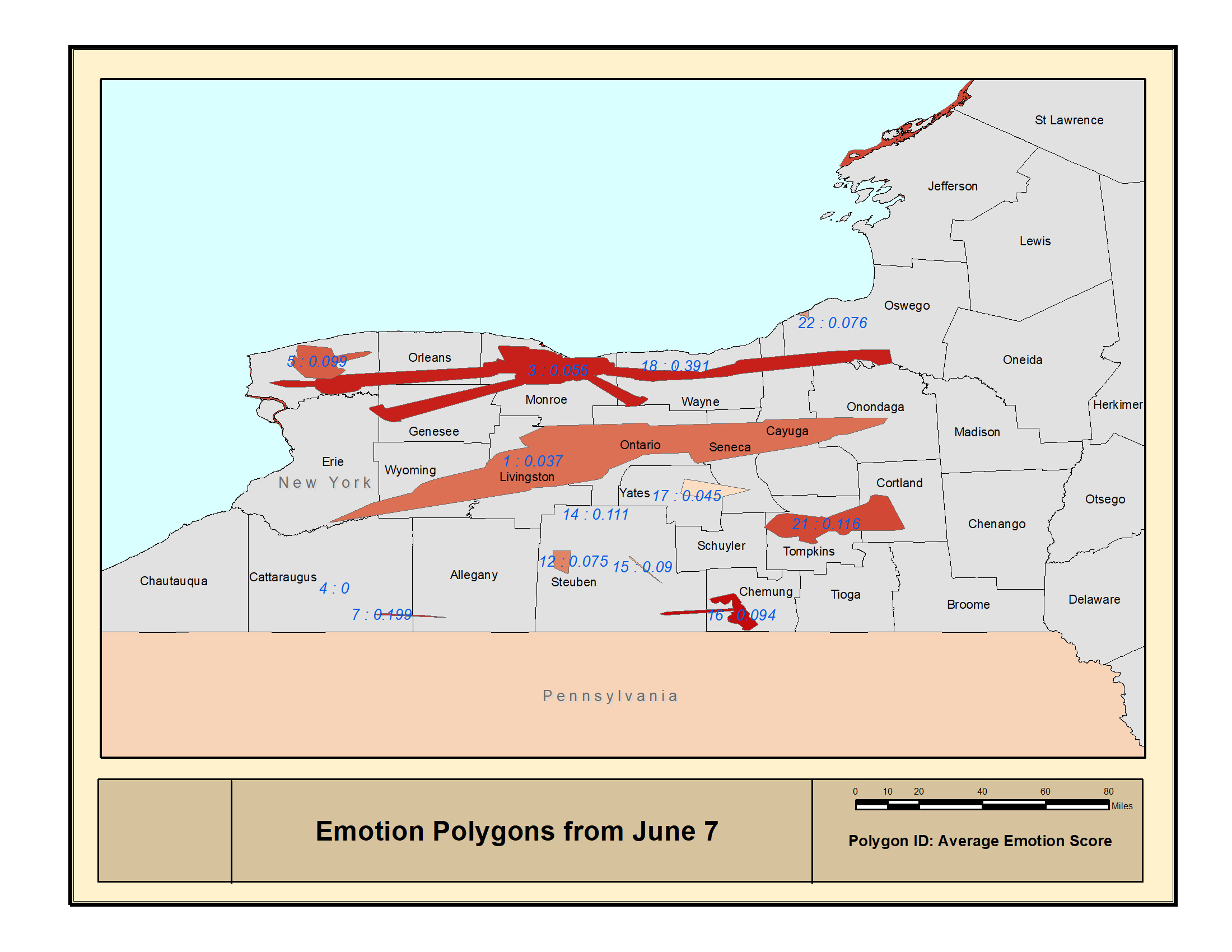


Figure 8: Emotion polygons created from June 7 Twitter data

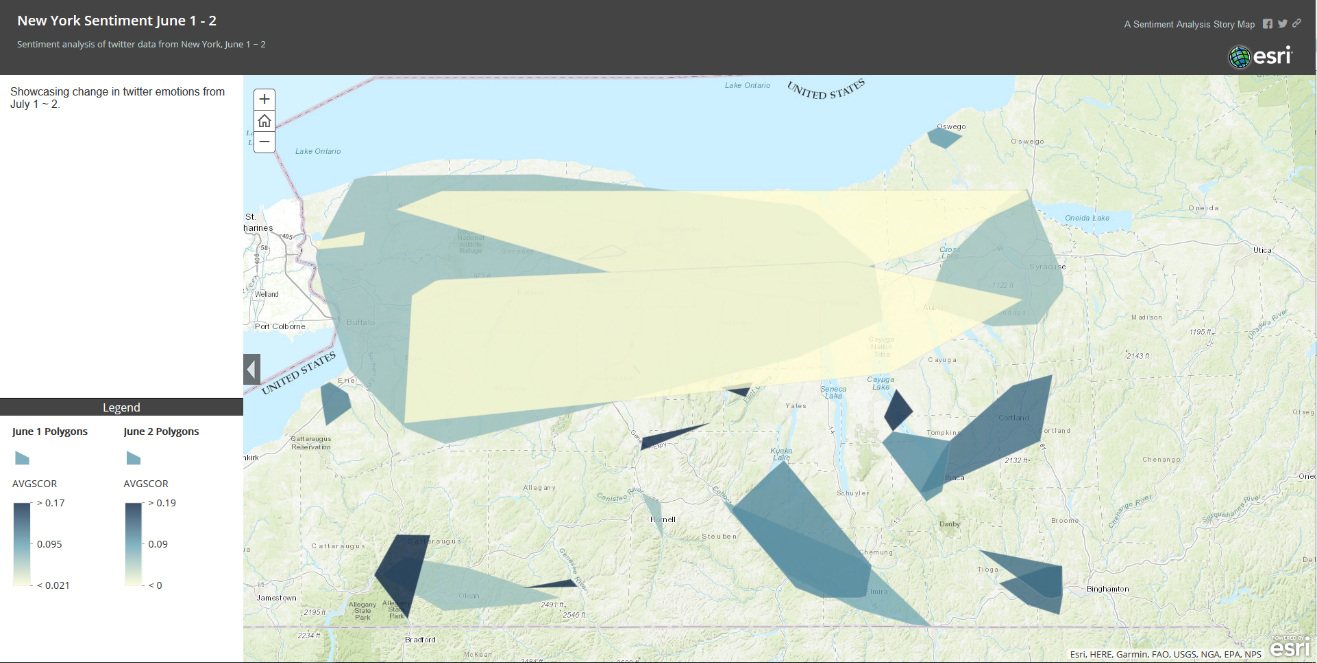


Figure 9: Emotion polygons created from June 1 Twitter data Using Convex Hulls

*Drought Data Sources*

The original polygons for the drought datasets, before our analysis is in Figure 10.

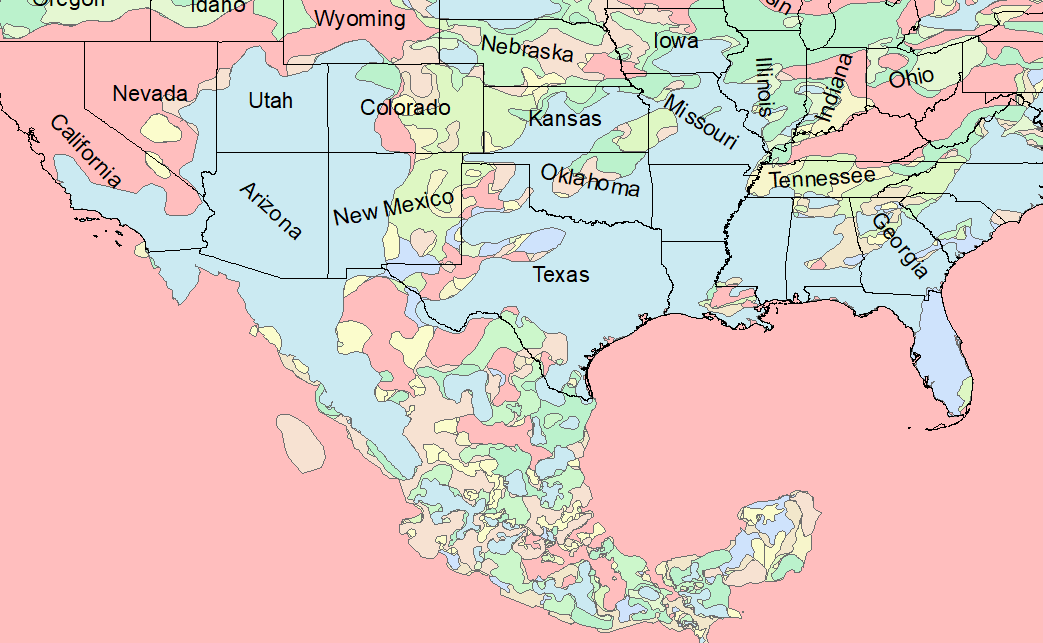


Figure 10: Continental USA original monthly drought polygon distribution for 2017

Now, we demonstrate the three change predicates as discussed in Chapter 3.

1. To detect polygons that are increasing in size, we check for similar IDs, intersection and then the rate of overlap. We initially check whether the polygons intersect at all before querying for amount of overlap. This leads to faster processing as it discards the many combinations where the polygons don't touch each other. Our query is structured as:

SELECT DISTINCT j2.\*

FROM public.june1poly j1, public.june2poly j2

WHERE ST\_INTERSECTS(j1.geom, j2.geom)

AND

(ST\_AREA(ST\_INTERSECTION(j2.geom, j1)) /st\_area(j2.geom)) > .85

A result of this operation performed on the drought dataset on September ~ October, 2017 can be seen in Figure 11.

1. To detect polygons that are shrinking in size, we check for similar IDs, and lower rates of overlap. This can be modified based on need.

SELECT DISTINCT j2.\*

FROM public.june1poly j1, public.june2poly j2

WHERE ST\_INTERSECTS(j1.geom, j2.geom)

AND

(ST\_AREA(ST\_INTERSECTION(j2.geom, j1)) /st\_area(j2.geom)) < .25

A demonstration of the output of this operation performed on the drought dataset on September ~ October, 2017 is found in Figure 12.

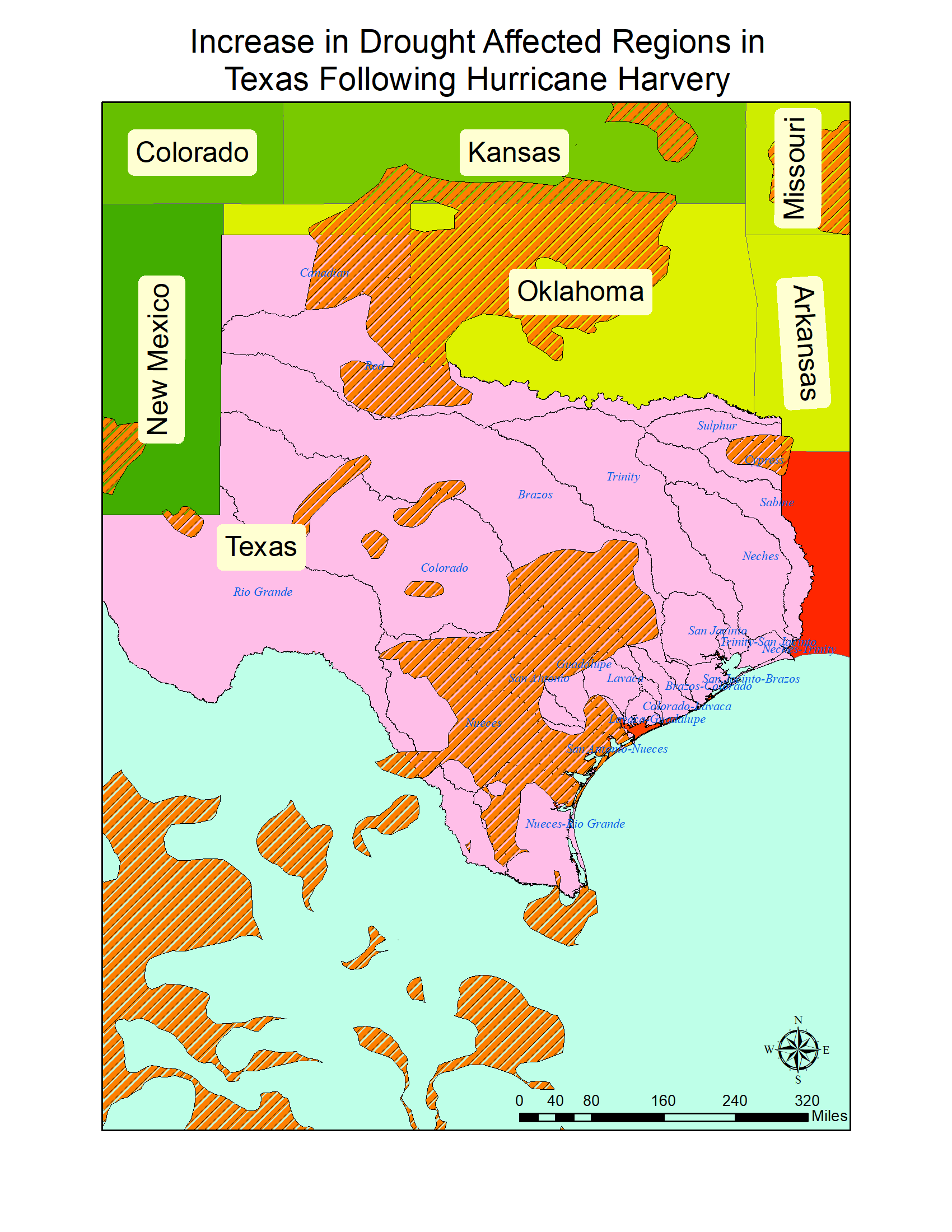


Figure 11: Increase in Drought Affected Regions in Texas following Hurricane Harvey

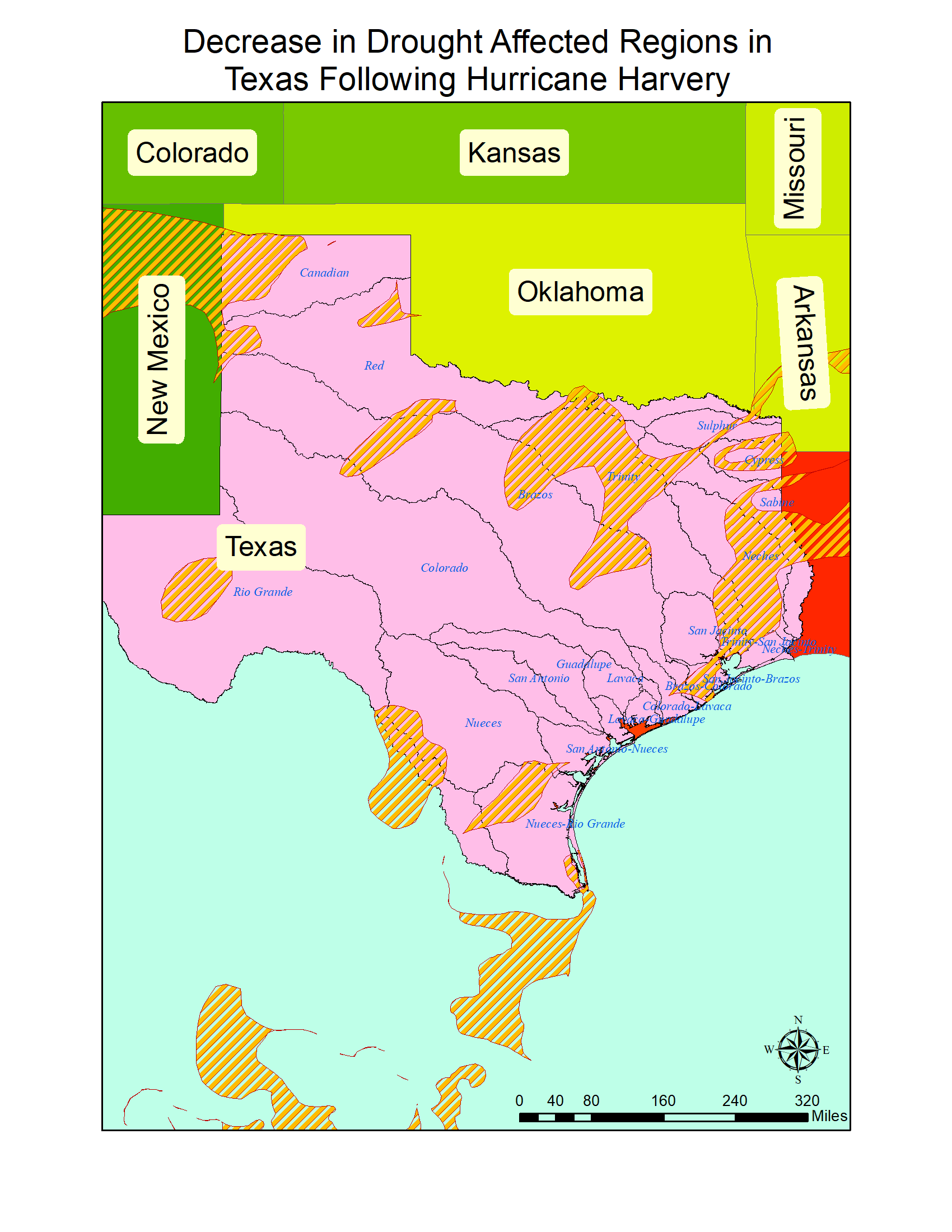


Figure 12: Decrease in Drought Affected Regions in Texas following Hurricane Harvey

1. To detect polygons that have shifted we compare their centroids and check if they have moved over 75km. Remember these polygons are created through a convex hull of points, which cannot ensure the centroid will lie within the polygon itself. Which is why we are taking a sufficiently large bounding value for the polygon.

SELECT ST\_Distance\_Spheroid

(j1.centroid, j2.centroid, 'SPHEROID["WGS 84",6378137,298.25]'), j1.id

FROM public.june1poly j1, public.june2poly j2

WHERE j1.id = j2.id AND ST\_Distance\_Spheroid(j1.centroid, j2.centroid,

'SPHEROID["WGS 84",6378137,298.25]')>75000;

An example of polygons shifting around can be found in Figure 13, highlighting changes in the California area following the 2017 wildfires.

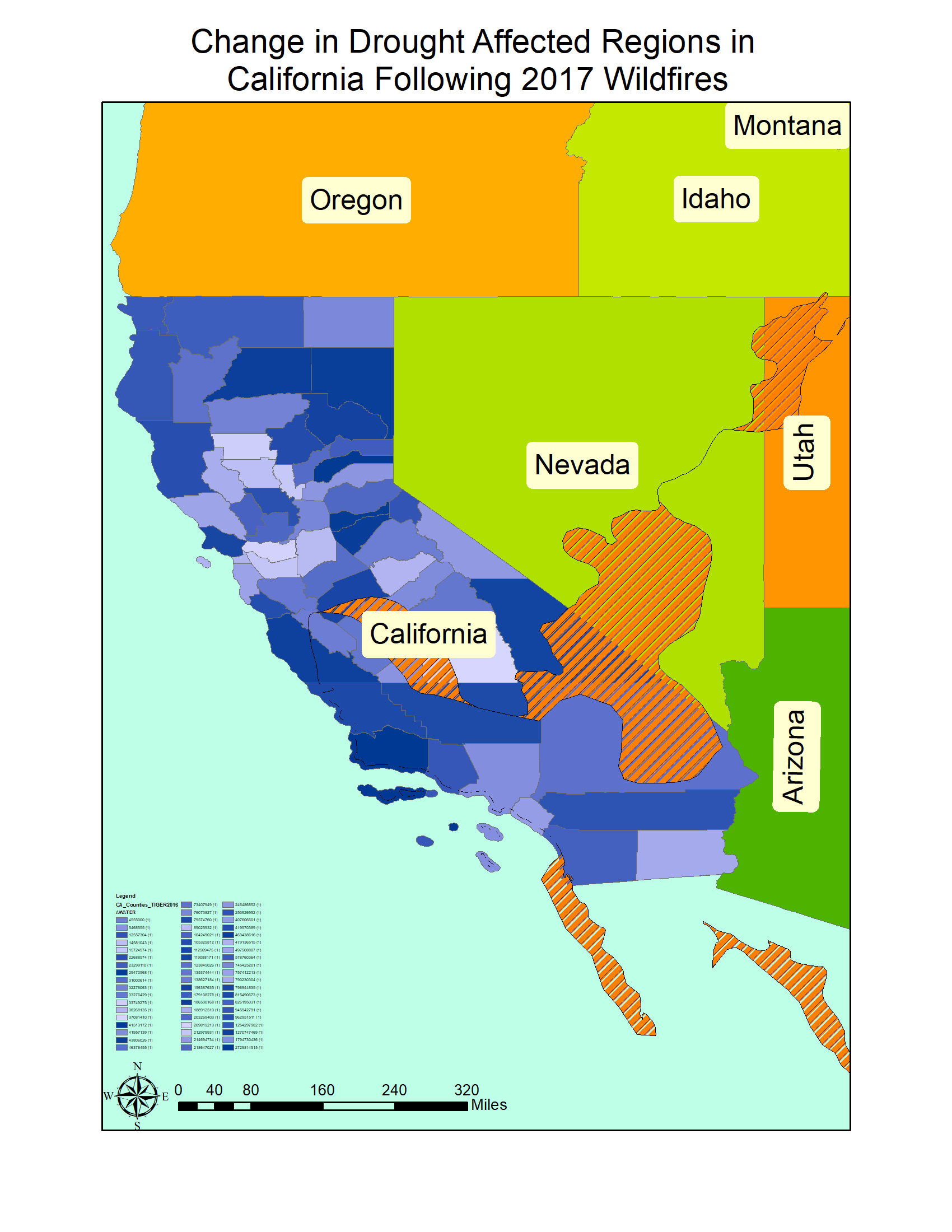


Figure 13: Shifting areas of drought affected regions in California following the 2017 wildfires

*Emotion Spatial Clusters from Twitter*

The tool used for creating the emotion clusters was the K2 framework. The spatial reference for the twitter data can be seen in figure 2.

According to K2, the best package for the emotion score was the Valence Aware Dictionary and Sentiment Reasoner (VADER) system, whose analyser parses the tokenized text and checks within a lexicon for words with strong sentiment. The final score is created from the weighted average of all sentimental words and lies within the range [-1, 1], as is required for our Aconcagua implementation.

The results of running our framework on June 5, 6 and 7 can be seen in Figure 15 and Figure 15.

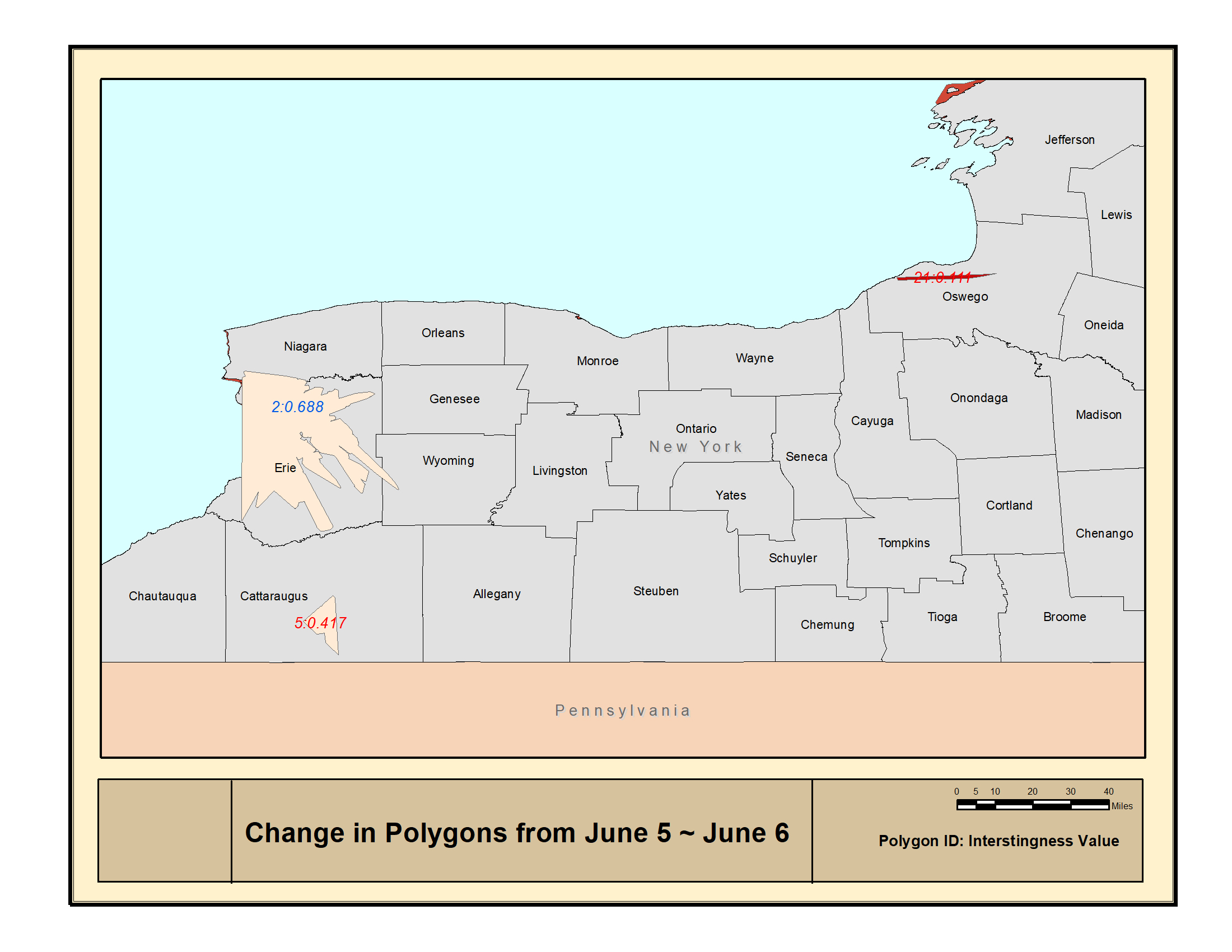


Figure 14: Change in Twitter emotion polygons from June 5 to June 6

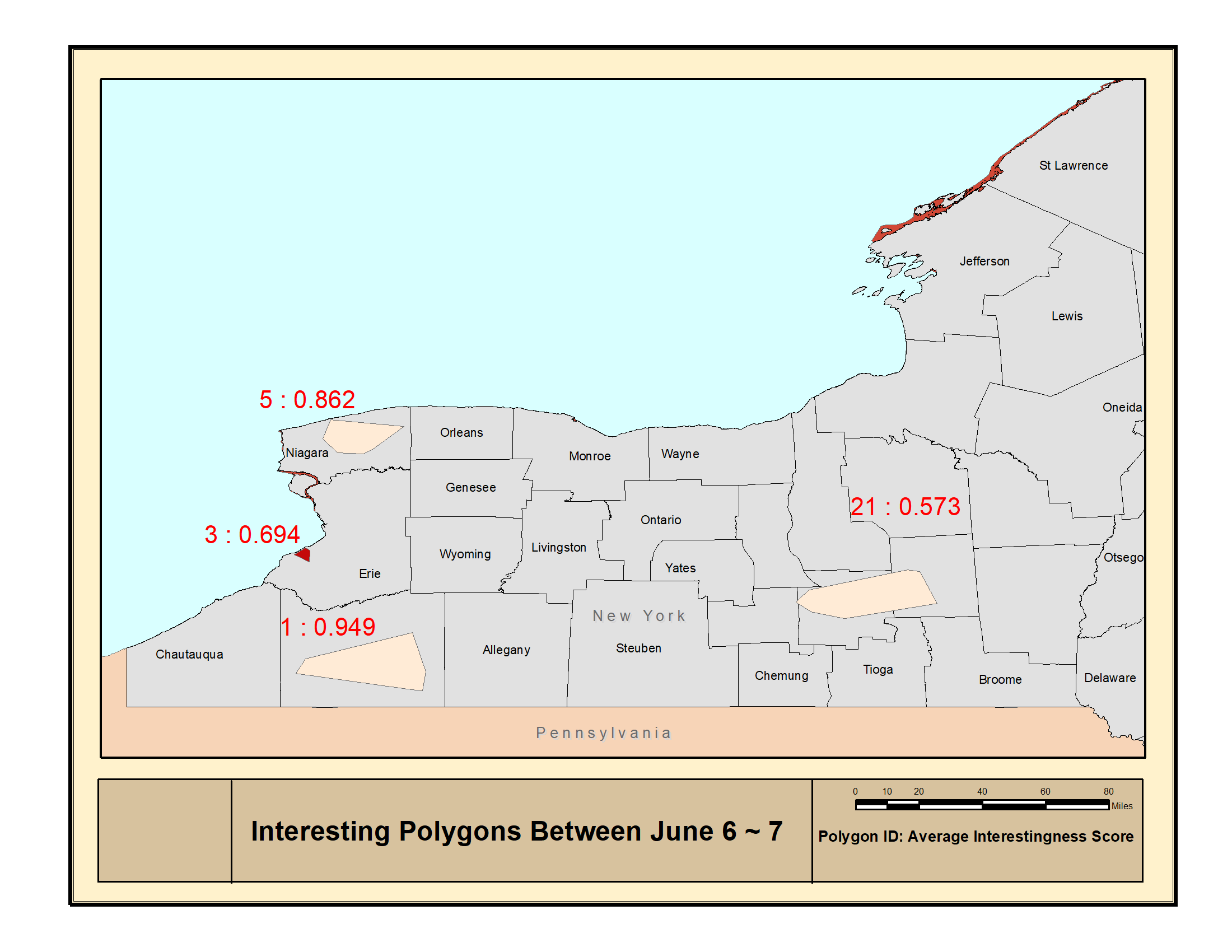


Figure 15: Change in Twitter emotion polygons from June 6 to June 7

*Analysis*

We focus on two specific regions for this case study. First, Figure 8 shows the regions where areas of drought grew in California after the wildfires in 2017. We noticed patterns of increasing drought surrounding the regions that were burnt down, with especially large period upstream from rivers. Our procedure works well dealing with the large number of small polygons in the region that occur due to isolated wildfires. However, we had difficult dealing with convex polygons that intersected with each other multiple times.

Next, we inspect the region in Texas after Hurricane Harvey. Figure 9 shows the regions where drought affected regions increased following the disaster and Figure 10} shows the regions decreased.

We observe that drought affected regions decrease at a high rate around the South Eastern Texas and Louisiana region. However, there is no clear relation between drought prone regions and river basins close to the coast. We believe that high amount industrial regions create a micro climate that affects the water content entering the soil.

According to our results, the drought prone regions increased substantially around the West Texas region both during and after Hurricane Harvey, with a larger increase as winter came around. However, this could be an artefact of the fact that the original dataset had very few polygons in the Texas area, which leads to broader conclusions. Our technique is still severely dependent on the resolution of the input imagery.

The results of our interestingness calculation, as seen in Figure 14 tell us that the smaller polygons with high emotion score and large amounts of variability are the most interesting. This tells us that regions with most outspoken or passionate tweets are the ones we should focus on when we are generating automated stories, which is obvious. Comparing Figure 12 with Figure 7 and Figure 8, we find that the large central polygon with very low emotion scores do not have much variance, telling us that that large group of people in a rural do not change their opinions dramatically day by day.

Chapter 5

Conclusion and Future Work

Our experimental studies show that our change detection and analysis framework can successfully detect changes in spatiotemporal datasets. Our change predicates can work on a variety of data sources, including both polygonal and point datasets. Our framework is able to handle both geographically large and small map sources without any significant georeferencing distortions being introduced.

There is a lot work left in dealing with irregularly shaped convex polygons that appear in poorly georeferenced real-world data. We believe this is because the sensitivity of the geographic operations in PostGIS or ArcGIS. We are currently working on extending our framework to deal with similar cases.

Bibliography

“Shapefiles of drought intensity and impact for the North American Drought Portal.” [Online]. Available: https://www1.ncdc.noaa.gov/pub/data/nidis/shapefiles/. [Accessed: 01-Aug-2019].

“Top 5 sources of big data | Artificial Intelligence | Data Science |,” 26-Nov-2017. [Online]. Available: *https://www.**allerin.com/blog/top-5-sources-of-big-data.* [Accessed: 26-Jul-2019].

Banerjee, Romita, K. Elgarroussi, S. Wang, A. Talari, Y. Zhang, and C. F. Eick, “K2: A Novel Data Analysis Framework to Understand US Emotions in Space and Time,” *Int. J. Semantic Computing*, vol. 13, no. 01, pp. 111–133, Mar. 2019.

Chen, Zhiqian, Xuchao Zhang, Arnold P. Boedihardjo, Jing Dai, and Chang-Tien Lu. 2017. “Multimodal Storytelling via Generative Adversarial Imitation Learning.” *ArXiv:1712.01455 [Cs]*, December. <http://arxiv.org/abs/1712.01455>.

Dos Santos, Raimundo F. Jr., Arnold Boedihardjo, Sumit Shah, Feng Chen, Chang-Tien Lu, Naren Ramakrishnan, “The big data of violent events: algorithms for association analysis using spatio-temporal storytelling,” *Geoinformatica*, v.20 n.4, p.879-921, October 2016

Dos Santos, Raimundo F. Jr., S. Shah, F. Chen, A. Boedihardjo, Chang-Tien Lu, N. Ramakrishnan, “Forecasting location-based events with spatio-temporal storytelling”, in *Proceedings of the 7th ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, pp. 13-22, November 04-04, 2014, Dallas/Fort Worth, Texas.

Dos Santos, Raimundo F. Jr., S. Shah, F. Chen, A. Boedi-hardjo, P. Butler, C.T. Lu and N. Ramakrishnan, "Spatiotemporal storytelling on twitter", *Virginia Tech Computer Science Technical Report*, 2015, [online] Available: http://vtechworks.lib.vt.edu/handle/10919/24701.

Egenhofer, Max J., and John Herring. "Categorizing binary topological relations between regions, lines, and points in geographic databases." *The* 9, no. 94-1 (1990): 76.

Elgarroussi K., S. Wang, R. Banerjee, and C. F. Eick, “Aconcagua: A Novel Spatiotemporal Emotion Change Analysis Framework,” in *Proceedings of the 2Nd ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery*, New York, NY, USA, 2018, pp. 54–61.

Geomenke, “How Accurate is the GPS on my Smart Phone? (Part 2),” *Community Health Maps*, 07-Jul-2014. .

Hossain, M. Shahriar, Patrick Butler, Arnold P. Boedihardjo, Naren Ramakrishnan, “Storytelling in Entity Networks to Support Intelligence Analysts,” *KDD '12 Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, Pages 1375-1383 ,Beijing, China , August 12 - 16, 2012

Huang, X., A. An, and N. Cercone, “Comparison of Interestingness Functions for Learning Web Usage Patterns,” in *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, New York, NY, USA, 2002, pp. 617–620.

Im, J. and J. Jensen. 2005. “A change detection model based on neighborhood correlation image analysis and decision tree classification.” *Remote Sensing of Environment* 99, 3 (2005), 326–340. doi:http://dx.doi.org/10.1016/j.rse.2005.09.008

Jin, Suming, Limin Yang, Patrick Danielson, Collin Homer, Joyce Fry, and George Xian. 2013. “A Comprehensive Change Detection Method for Updating the National Land Cover Database to circa 2011.” *Remote Sensing of Environment* 132 (May): 159–75. <https://doi.org/10.1016/j.rse.2013.01.012>.

Kumar, D., N. Ramakrishnan, R. F. Helm and M. Potts, “Algorithm for storytelling”, in *IEEE Transactions on Knowledge and Data Engineering*, 2008, Volume 20 Issue

Kumar, Deept, Naren Ramakrishnan, Richard F. Helm, and Malcolm Potts. 2008. *Algorithms for Storytelling. IEEE TKDE20*, 6 (2 2008), 736–751.doi:http://dx.doi.org/10.1145/1188913.1188915

Lu, D., P. Mausel, E. Brondízio, and E. Moran. 2004. Change detection techniques. *International Journal of Remote Sensing* 25, 12 (June 2004), 2365–2401. doi:http://dx.doi.org/10.1080/0143116031000139863

Madhavan, Jayant, Sreeram Balakrishnan, Kathryn Brisbin, Hector Gonzalez, Nitin Gupta, Alon Halevy, Karen Jacqmin-adams, et al. n.d. 2013 *Big Data Storytelling through Interactive Maps*. https://ai.google/research/pubs/pub39959/

Minu, S. and Amba Shetty. 2015. A Comparative Study of Image Change Detection Algorithms in MATLAB. *Aquatic Procedia* 4 (March 2015), 1366–1373. doi:http://dx.doi.org/10.1016/j.aqpro.2015.02.177

Ramakrishnan, N., D. Kumar, B. Mishra, M. Potts and R. R. Helm, Turning “CARTwheels: an alternating algorithm for mining redescriptions”, in *Knowledge Discovery and Data Mining*, 2004, pp. 266-275.

Ridd, Merrill K. and Jiajun Liu. 1998. A Comparison of Four Algorithms for Change Detection in an Urban Environment. *Remote Sensing of Environment* 63, 2 (1998), 95–100. doi:http://dx.doi.org/10.1016/s0034-4257(97)00112-0

Robertson, Colin, Trisalyn A. Nelson, Barry Boots, and Michael A. Wulder. 2007. “STAMP: Spatial–Temporal Analysis of Moving Polygons.” *Journal of Geographical Systems* 9 (3): 207–27. <https://doi.org/10.1007/s10109-007-0044-2>.

Thatcher, J., “Big Data, Big Questions| Living on Fumes: Digital Footprints, Data Fumes, and the Limitations of Spatial Big Data,” *International Journal of Communication*, vol. 8, no. 0, p. 19, Jun. 2014.

Vocht, L. D., C. Beecks, R. Verborgh, T. Seidl, E. Mannens and Rik Van De Walle, “Improving Semantic Relatedness in Paths for Storytelling with Linked Data on the Web”, in *Revised Selected Papers of the ESWC 2015 Satellite Events on The Semantic Web: ESWC 2015 Satellite Events*, May 31-June 04, 2015.

W. Z.-M.-L. I. for the S. S. Person, “Geotagged Twitter posts from the United States: A tweet collection to investigate representativeness,” Jan. 2016.

Wang, W, Lu Chen, K. Thirunarayan, and Amit P. Sheth, Harnessing Twitter "Big Data" for Automatic Emotion Identifcation, in *Proc. of the IEEE 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Conference on Social Computing*.

Wu, Qiang, Fatima Merchant, and Kenneth Castleman. 2010. *Microscope Image Processing*. 195 – 202. Elsevier.

Appendices

Appendix A Get Layer Specifications and Convert Raw Point Data to Polygons

Appendix B SQL Queries for Change Analysis

Appendix C Interestingness Function Definitions

Appendix A

SQL Queries for Change Analysis

*# Get Projection from layer*

layer = shp.GetLayer()

spatialRef = layer.GetSpatialRef()

print (spatialRef)

*# Get Shapefile Fields and Types*

layerDefinition = layer.GetLayerDefn()

print ("Name,Type,Width,Precision")

for i in range(layerDefinition.GetFieldCount()):

fieldName = layerDefinition.GetFieldDefn(i).GetName()

fieldTypeCode = layerDefinition.GetFieldDefn(i).GetType()

fieldType = layerDefinition.GetFieldDefn(i).GetFieldTypeName(fieldTypeCode)

fieldWidth = layerDefinition.GetFieldDefn(i).GetWidth()

GetPrecision = layerDefinition.GetFieldDefn(i).GetPrecision()

print (fieldName + "," + fieldType+ "," + str(fieldWidth) + "," + str(GetPrecision))

def convertPointToPoly(pointName, polyName,type='Concave',shrink=0.8):

connection = psycopg2.connect(database="twitter\_change",user="postgres", password="password")

cursor = connection.cursor()

cursor.execute('DROP TABLE IF EXISTS '+polyName)

cursor.execute('CREATE TABLE '+polyName+' (gid SERIAL PRIMARY KEY, avgscor numeric, numtwts integer, id integer, batchnm integer, geodata character varying(80) COLLATE pg\_catalog."default", geom geometry(Polygon), centroid geometry(Point))')

cursor.execute('INSERT INTO '+polyName+' (avgscor, numtwts, geodata, id, batchnm, geom)'+

'(SELECT d.avgscor, d.numtwts, d.geodata, d.id, d.batchnm, ST\_'+type+'HULL(ST\_Collect(d.geom), '+str(shrink)+')'+

' FROM '+pointName+' AS d'+

' GROUP BY (d.id, d.avgscor, d.numtwts, d.geodata, d.batchnm) '+

' HAVING ST\_GeometryType(ST\_'+type+'HULL(ST\_Collect(d.geom), '+str(shrink)+')) = \'ST\_Polygon\')')

cursor.execute('UPDATE '+polyName+' SET centroid=ST\_Centroid(geom)')

connection.commit()

Appendix B

SQL Queries for Change Analysis

We create two generic queries to test what kind of results we are getting from the grow and shrink predicates. Next we create the detector table, populate them and include the percentage change.

**def** **change\_shrink**(poly1, poly2, factor=**0.5**):

newpoly = 'public.'+poly1[:-**4**]+poly2[:-**4**]+'shrink'

connection = psycopg2.connect(database="twitter\_change",user="postgres", password="password")

cursor = connection.cursor()

cursor.execute('DROP TABLE IF EXISTS '+newpoly)

cursor.execute('CREATE TABLE '+newpoly+' (gid SERIAL PRIMARY KEY, avgscor numeric, numtwts integer, id integer, batchnm integer, geodata character varying(80) COLLATE pg\_catalog."default", geom geometry(Polygon), centroid geometry(Point), changval numeric)')

cursor.execute('INSERT INTO '+newpoly+' (avgscor, numtwts, geodata, id, batchnm, geom, centroid, changval)'+

' (SELECT DISTINCT j2.avgscor, j2.numtwts, j2.geodata, j2.id, j2.batchnm, j2.geom, j2.centroid,'+

' (ST\_AREA(ST\_INTERSECTION(j2.geom, j1.geom))/st\_area(j2.geom))'+

' FROM public.'+poly1+' j1, public.'+poly2+' j2'+

' WHERE j1.id = j2.id AND ST\_INTERSECTS(j1.geom, j2.geom) AND '+

'(ST\_AREA(ST\_INTERSECTION(j2.geom, j1.geom))/st\_area(j2.geom)) < '+str(factor)+') ')

connection.commit()

Now we do the same but for growing polygons. We insert those polygons with an overlap area of less than 25%. This can be modified based on need.

**def** **change\_growth**(poly1, poly2, factor=**0.5**):

newpoly = 'public.'+poly1[:-**4**]+poly2[:-**4**]+'growth'

connection = psycopg2.connect(database="twitter\_change",user="postgres", password="password")

cursor = connection.cursor()

cursor.execute('DROP TABLE IF EXISTS '+newpoly)

cursor.execute('CREATE TABLE '+newpoly+' (gid SERIAL PRIMARY KEY, avgscor numeric, numtwts integer, id integer, batchnm integer, geodata character varying(80) COLLATE pg\_catalog."default", geom geometry(Polygon), centroid geometry(Point), changval numeric)')

cursor.execute('INSERT INTO '+newpoly+' (avgscor, numtwts, geodata, id, batchnm, geom, centroid, changval)'+

' (SELECT DISTINCT j2.avgscor, j2.numtwts, j2.geodata, j2.id, j2.batchnm, j2.geom, j2.centroid,'+

' (ST\_AREA(ST\_INTERSECTION(j2.geom, j1.geom))/st\_area(j2.geom))'+

' FROM public.'+poly1+' j1, public.'+poly2+' j2'+

' WHERE j1.id = j2.id AND ST\_INTERSECTS(j1.geom, j2.geom) AND '+

'(ST\_AREA(ST\_INTERSECTION(j2.geom, j1.geom))/st\_area(j2.geom)) > '+str(factor)+') ')

connection.commit()

Now we investigate if polygons are moving around. This done by the comparing their centroids and checking if they have moved over 75km.

Remember these polygons are created through a convex hull of points, which cannot ensure the centroid will lie within the polygon itself. Which is why we are taking a sufficiently large bounding value for the polygon.

**def** **change\_shift**(poly1, poly2, distance=**75000**):

newpoly = 'public.'+poly1[:-**4**]+poly2[:-**4**]+'shift'

connection = psycopg2.connect(database="twitter\_change",user="postgres", password="password")

cursor = connection.cursor()

cursor.execute('DROP TABLE IF EXISTS '+newpoly)

cursor.execute('CREATE TABLE '+newpoly+' (gid SERIAL PRIMARY KEY, avgscor numeric, numtwts integer, id integer, batchnm integer, geodata character varying(80) COLLATE pg\_catalog."default", geom geometry(Polygon), centroid geometry(Point), shiftdist numeric)')

cursor.execute('INSERT INTO '+newpoly+' (avgscor, numtwts, geodata, id, batchnm, geom, centroid, shiftdist)'+

'(SELECT j1.avgscor, j1.numtwts, j1.geodata, j1.id, j1.batchnm, j1.geom, j1.centroid, d.distval'+

' FROM public.'+poly1+' j1,'+

' (SELECT ST\_Distance\_Spheroid(j1.centroid, j2.centroid,' +

' \'SPHEROID["WGS 84",6378137,298.257223563]\') as distval, j1.id AS jid'+

' FROM public.'+poly1+' j1, public.'+poly2+' j2 '+

' WHERE j1.id = j2.id AND '+

' ST\_Distance\_Spheroid(j1.centroid, j2.centroid, \'SPHEROID["WGS 84",6378137,298.257223563]\') > '+str(distance)+') as d'+

' WHERE j1.id = d.jid)')

connection.commit()

Now we carefully insert the polygons into the table, comparing all sequential days.

#Comparing all sequential days

polygons =[]

**for** val in range(**1**,**8**):

polygons.append('june'+str(val)+'poly')

**for** val in range(**6**):

**if** val < **5**:

change\_shrink(polygons[val],polygons[val+**1**])

change\_growth(polygons[val],polygons[val+**1**])

change\_shift(polygons[val],polygons[val+**1**])

Appendix C

Interestingness Function Definitions

Sample code to generate the parameters for the interestingness function we defined. First, some helper functions, such as inserting a column into our database with the area of every polygon contained within.

**def** **addAreaCol**(tablename):

connection = psycopg2.connect(database="twitter\_change",user="postgres", password="password")

cursor = connection.cursor()

cursor.execute('ALTER TABLE '+tablename+' ADD COLUMN IF NOT EXISTS area double precision')

cursor.execute('UPDATE '+tablename+' SET area=ROUND((ST\_Area(geom::geography))::numeric,2)')

connection.commit()

A function to get the geodata from the database and add it to a python list:

**def** **get\_data**(tableName):

connection = psycopg2.connect(database="twitter\_change",user="postgres", password="password")

cursor = connection.cursor()

cursor.execute('SELECT j2s.\* '+

'FROM '+tableName+' j2s ')

temp\_table = cursor.fetchall()

connection.commit()

**return** (temp\_table)

1. Alpha

**def** **interesting\_alpha**(tablename):

table = get\_data(tablename)

sum = **0**

alpha = {}

**for** val in table:

sum += val[**9**]

**for** val in table:

alpha.update({val[**0**]:(val[**9**]/sum)})

**return**(alpha)

1. Beta

**def** **interesting\_beta**(tablename):

#use for growth/shrinks

table = get\_data(tablename)

beta = {}

**for** val in table:

beta.update({val[**0**]:float(val[**8**])})

**return** (beta)

1. Gamma

**def** **interesting\_gamma**(tablename):

#use for shifts

table = get\_data(tablename)

connection = psycopg2.connect(database="twitter\_change",user="postgres", password="password")

cursor = connection.cursor()

cursor.execute('SELECT ST\_Distance\_Sphere(geometry(a.bot), geometry(b.top))'

'FROM '+

' (SELECT st\_astext(st\_makepoint(st\_xmin(st\_extent(j2s.geom)), st\_ymin(st\_extent(j2s.geom)))) AS bot FROM '+tablename+' j2s) a, '

' (SELECT st\_astext(st\_makepoint(st\_xmax(st\_extent(j2s.geom)), st\_ymax(st\_extent(j2s.geom)))) AS top FROM '+tablename+' j2s) b')

distance = cursor.fetchall()

connection.commit()

gamma ={}

**for** val in table:

gamma.update({val[**0**]:int(val[**8**] >= (distance[**0**][**0**]/**2**))})

**return** (gamma)

1. Delta

**def** **interesting\_delta**(tablename):

#Thinness ratio

table = get\_data(tablename)

connection = psycopg2.connect(database="twitter\_change",user="postgres", password="password")

cursor = connection.cursor()

cursor.execute('SELECT ST\_Perimeter(j2s.geom, true)'

'FROM '+tablename+' j2s ')

perimeter = cursor.fetchall()

connection.commit()

thin\_ratio = **0**

delta = {}

**for** p, a in zip(perimeter, table):

thin\_ratio = (**4** \* math.pi \* a[**9**]) / (p[**0**] \*\* **2**)

delta.update({a[**0**]:thin\_ratio})

**return** (delta)

1. Omega

**def** **interesting\_omega**(alpha, beta, delta):

omega = {}

**for** val in range(**1**, (len(alpha)+**1**)):

omega.update({val: (**0.3** \* alpha[val]) + (**0.4** \* beta[val]) + (**0.4** \* delta[val])})

**return** (omega)

A sample use case to calculate the parameters on several consecutive days:

#Now running this for all dates

#Comparing all sequential days

polygons =[]

**for** val in range(**1**,**7**):

temp = 'public.june'+str(val)+'june'+str(val+**1**)

polygons.append([temp+'shrink',temp+'growth',temp+'shift'])

**for** elem in polygons:

#Shrink

addAreaCol(elem[**0**])

a34 = interesting\_alpha(elem[**0**])

b34 = interesting\_beta(elem[**0**])

d34 = interesting\_delta(elem[**0**])

omegaToTable(interesting\_omega(a34, b34, d34), elem[**0**])

#Growth

addAreaCol(elem[**1**])

a34 = interesting\_alpha(elem[**1**])

b34 = interesting\_beta(elem[**1**])

d34 = interesting\_delta(elem[**1**])

omegaToTable(interesting\_omega(a34, b34, d34), elem[**1**])

#Shift

addAreaCol(elem[**2**])

a34 = interesting\_alpha(elem[**2**])

c34 = interesting\_gamma(elem[**2**])

d34 = interesting\_delta(elem[**2**])

omegaToTable(interesting\_omega(a34, c34, d34), elem[**2**])