

Bridget Damweber
Muntaser Syed
Artificial Intelligence
Hurricane Data Project – Second Report
12/5/2017

The hurricane data for Atlantic Basin hurricanes can be found online at the National Hurricane Center website under the HURDAT2 database. Different approaches have been taken to analyze North Atlantic hurricane data, leveraging different clustering algorithms. One such study by Corporal-Lodangco, et al, approaches the problem with K-means clustering and examines the source location of the hurricane, i.e. where the hurricane originated. This study also examines the track data and finds a correlation between track patterns and the month of the year in which the hurricane track incidence is found.¹ One disadvantage of this approach is that, for our purposes, we are looking for an arbitrary number of clusters based on track, not genesis location. Using K-means on the hurricane data groups the hurricanes by location, but not path. For geographical location independent study, this approach will not lead to successful clustering of hurricanes in disparate geographical locations. The scope of the current paper is to analyze the hurricane data provided by the HURDAT2 data by applying different comparison algorithms to group hurricane tracks into meaningful clusters, then use hierarchical clustering to group the hurricanes together using a minimum comparison value, or “distance”, between the two hurricanes as criteria for building the cluster, then using hierarchical clustering to produce groups of hurricanes with similar track trajectories.

When approaching the clustering of the hurricane tracks, there are a few considerations to keep in mind. First, two hurricanes could have a similar path, but be geographically distinct. For this reason, we choose not to pin the hurricane path to a set of physical latitude and longitude points. Another consideration is the speed of the hurricane. This is a more challenging problem, because two similar hurricanes in terms of path could be moving through that path at different speeds, and the analyst must decide if two such hurricanes should be grouped or not. We chose two different algorithms to compute the similarity between two hurricane paths: Longest Common Subsequence, in strict and relaxed mode, and a variation on the Longest Common Subsequence that simply looks for a sequential pattern, but is not required to be a continuous subsequence. The first algorithm LCS will not cluster two similar path hurricanes moving at different speeds, but the variation on LCS, which looks for the sequential pattern, will pick up two similar hurricanes moving at different speeds. To simplify the calculation and reduce a hurricane path from a sequence of latitudes and longitudes to an integer sequence, we use star-calculus as developed by Mitra, et al to determine which radial sector a given bearing calculation between two geographical points belongs². We can test the need for the star calculus by dividing the 360-degree circle into 36 sectors, and then into 360 sectors to see if clustering improves. By modifying the LCS and LCS-variant algorithms to accept a range of valid sequential matches, we can accomplish a similar effect to the star calculus sectoring approach: {1,1,1,1,2,3,4,5} and {1,1,1,1,3,4,5,6} can then be clustered together, because their values are only off by the allowed factor of x between two integers.

¹ Corporal-Lodangco, et al. Science Direct. Procedia Computer Science 36 (2014) 293 – 300. Available online at <https://www.sciencedirect.com/science/article/pii/S1877050914013453>.

² Mitra, et al. Available online at <http://rutcor.rutgers.edu/~amai/aimath04/AcceptedPapers/Mitra-aimath04.pdf>

HURDAT2 data was parsed from CSV format the data and inserted into an indexing engine called Elastic Search, which is an open source free software download with limited initial features, more complex features available through license³. The ELK stack provides a visualization interface called Kibana which allows the data indexed into Elastic Search to be viewed. There is a limitation, however, since Kibana does not support drawing Polylines on a map, so we developed our own Model-View-Controller software to retrieve data from Elastic Search and draw the hurricane paths as polylines on a map. We installed the Elastic Search and Kibana on a AWS EC2 instance. The software uses the NEST and ElasticSearch.Net libraries to index data into the ELK (ElasticSearch-LogStash-Kibana) Stack. The code for indexing into ELK can be found in the ElasticDataReader C# project included with the submission, and the code for visualizing the data is available in the MVC project included with the submission.

The NEST and ElasticSearch.Net libraries require a data model with the correct annotations to properly index the data. The raw hurricane track data model was enriched with additional fields to include the bearing and distance calculation between each successive latitude-longitude pair for each hurricane. For this calculation, the following formula was used⁴:

```
public static double GetBearing(double lat1, double long1, double lat2, double long2)
{
    double angle = Math.Atan2(Math.Cos(lat1) * Math.Sin(lat2) - Math.Sin(lat1) * Math.Cos(lat2)
        * Math.Cos(long2 - long1), Math.Sin(long2 - long1) * Math.Cos(lat2));
    // = ATAN2(COS(lat1) * SIN(lat2) - SIN(lat1) * COS(lat2) * COS(lon2 - lon1), SIN(lon2 - lon1) * COS(lat2))
    // = ATAN2(COS(lat1)*SIN(lat2)-SIN(lat1)*COS(lat2)*COS(lon2-lon1), SIN(lon2-lon1)*COS(lat2))
    // convert to degrees
    angle = angle * (180 / Math.PI);
    if (angle < 0)
    {
        angle = 360 + angle;
    }
    return angle;
}
```

The next task is to leverage the Star Calculus method, mentioned earlier, of converting a sequence of Latitudes and Longitudes into a sequence of integers to represent the path. For this method, a granularity of 10 degrees was chosen to divide the 360 degrees into sectors. For each bearing, the floor value of the bearing divided by the granularity returned an integer:

```
public static int GetStarCalculusValue(int granularity, double bearing) {
    //0: 0-10
    //1: 11-20
    //2: 21-30
    int star_calculus_value = (int)Math.Floor(bearing/granularity);

    return star_calculus_value;
}
```

³ Online reference materials are available starting at <https://www.elastic.co/webinars/introduction-elk-stack>.

⁴ Bearing calculation assistance was derived from the online help at the following website: <https://www.movable-type.co.uk/scripts/latlong.html>.

Initially, it made sense to convert the values obtained through star calculus into a modular form which would enable the distance between sector 1 and 36 to be 1 instead of 35. However, we realized that this can only be done correctly at the time that the comparison between the two hurricane tracks is completed:

```
public static bool isMatchLCS(int a, int b)
{
    //return a == b;
    int c = a - b;
    while (c > +18) c -= 36;
    while (c < -18) c += 36;

    return Math.Abs(c) < 2;
}
```

This leverages modular arithmetic to view the sectors as distances along a circular path instead of linearly to facilitate the clustering step.

This data was indexed into Elastic Search index “hurdat5”. The data was manipulated to provide a dictionary of hurricane names as the key, and the list of integer sequences as the values. The result is the “sequences.txt” file included in the submission.

At this point, it became necessary to determine how the difference between two hurricane paths could be calculated. On advice from the instructor, the dynamic programming algorithm Longest Common Subsequence was used to compare each hurricane path with each other hurricane path⁵. As discussed above, the true LCS algorithm, a relaxed version of the algorithm, and a simpler algorithm which simply identifies sequences were used.

This alone was not sufficient to accomplish the clustering of the hurricanes. Further reading and research was done to understand different clustering approaches⁶. The clustering algorithm approach of hierarchical clustering was used to temporarily store the result of each LCS (Longest Common Subsequence) result in an initial matrix of values. The LCS algorithm returns an integer value corresponding to the number of matches between the two integer sequences.

The initial calculations of the LCS between each hurricane pair are cached on demand to save computational time. Each subsequent time the LCS is needed, the result is retrieved from the cache. The LCS algorithm was modified to input arrays of integers instead of character arrays.

In the initial pass, there is a single cluster id and a corresponding cluster created for each hurricane. At each iteration, the algorithm merges only one cluster with another cluster, and makes the determination of which two clusters to merge based on the largest LCS score of all LCS scores calculated

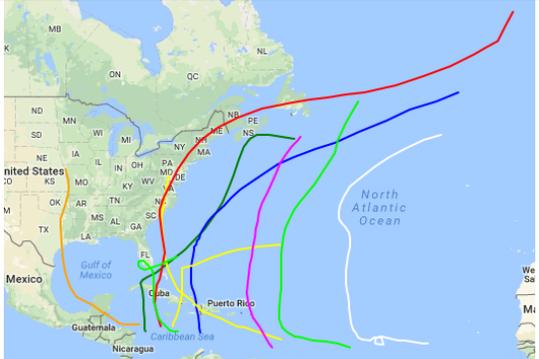
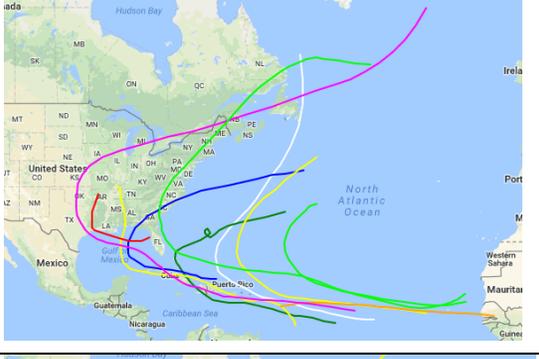
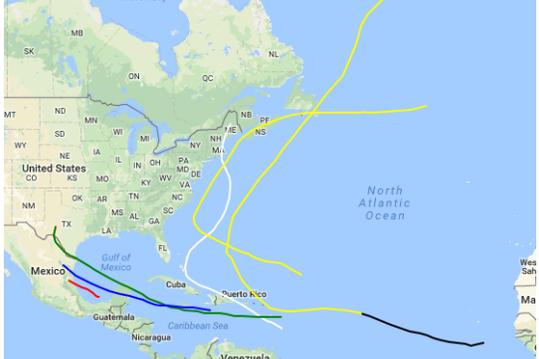
⁵ An example of the LCS algorithm was used as a basis for our algorithm. The example is available online at <http://seesharpconcepts.blogspot.com/2013/10/longest-common-substring-and-longest.html>.

⁶ Hierarchical clustering algorithm support was obtained from the online resource available at https://home.deib.polimi.it/matteucc/Clustering/tutorial_html/hierarchical.html.

between each cluster pair. When cluster pairs are scored, the lowest LCS score of all possible hurricane permutations becomes the LCS score for that cluster pair.

The function ConsolidateLCS searches for the best matching clustering to merge. If the minimum LCS score is not met, then no clusters are merged, and the original cluster list is returned. Each time the ConsolidateLCS function executes, two clusters, if they met the minimum LCS threshold, are merged, and all other clusters are copied each to a new cluster.

The algorithm stops when no qualifying cluster pairs' LCS return a value greater than threshold of 5 matching integers in the sequence. At this point, the cluster list is sorted by cluster count descending, and when equal, the clusters are sorted by the greatest average LCS score between all permutations of hurricanes in the two clusters with equal total hurricane counts.

True LCS minScore =10		
Cluster Name	Images plotted on map	Included Hurricanes
Cluster1		UNNAMED_03_1872, UNNAMED_09_1899, UNNAMED_07_1878, GUSTAV_08_1990, UNNAMED_03_1883, UNNAMED_04_1884, UNNAMED_04_1904, UNNAMED_01_1921, LISA_13_2004, UNNAMED_17_1933,
Cluster2		UNNAMED_05_1856, UNNAMED_08_1908, UNNAMED_06_1893, UNNAMED_02_1927, UNNAMED_02_1940, UNNAMED_15_1967, UNNAMED_05_1899, BAKER_02_1950, UNNAMED_01_1900, UNNAMED_02_1900,
Cluster 3		UNNAMED_02_1880, LUIS_13_1995, UNNAMED_09_1888, UNNAMED_03_1928, UNNAMED_02_1896, GEORGE_08_1951,

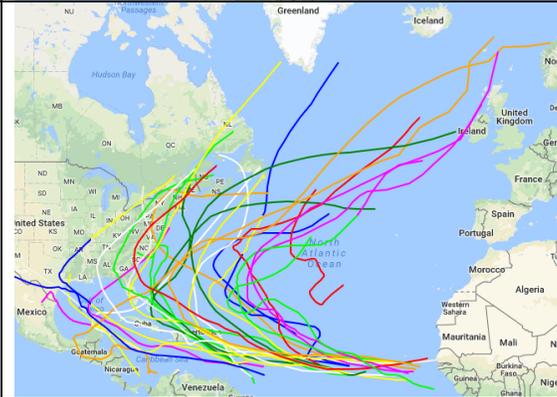
Cluster 4		<p>UNNAMED_07_1865, UNNAMED_06_1882, UNNAMED_02_1886, MICHELLE_15_2001, GINNY_08_1963, SANDY_18_2012,</p>
Cluster 5		<p>UNNAMED_04_1864, UNNAMED_05_1901, UNNAMED_09_1893, UNNAMED_05_1895, DOG_05_1951,</p>

Non-sequential match minScore =10		
Cluster Name	Images plotted on map	Included Hurricanes
Cluster 1		

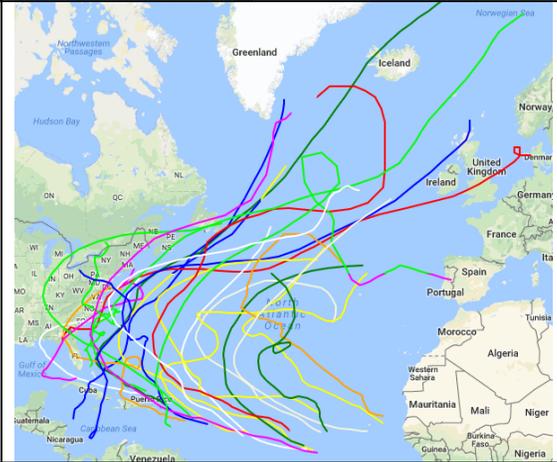
Cluster 2

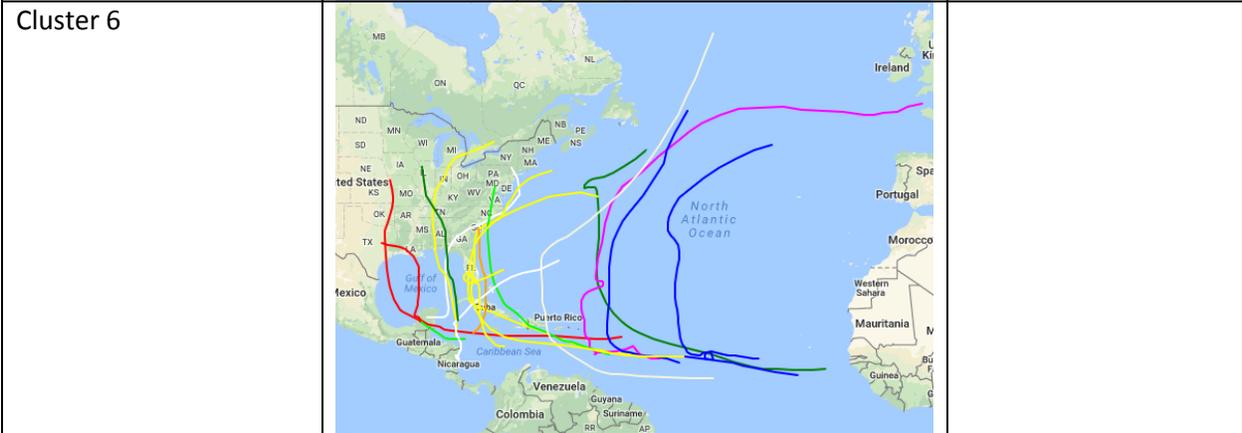
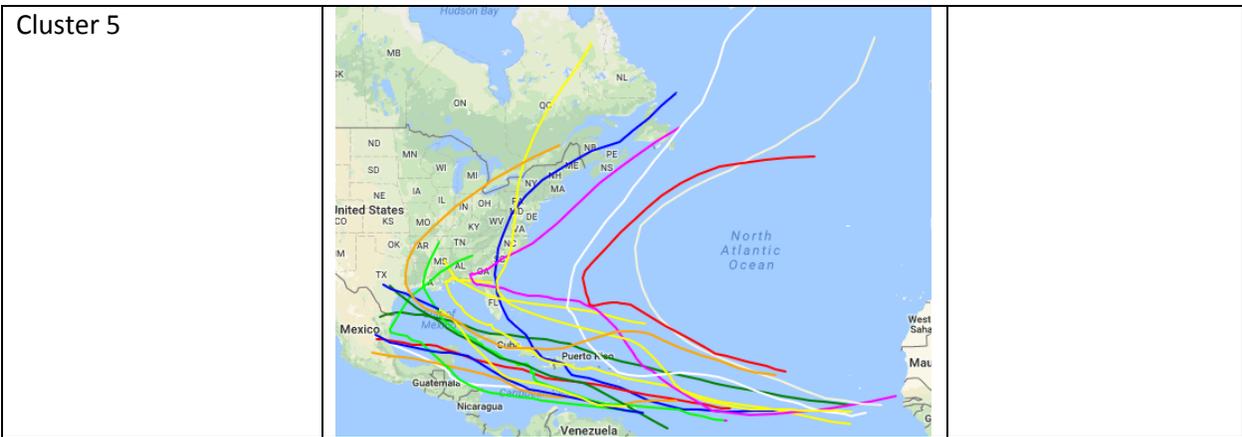


Cluster 3



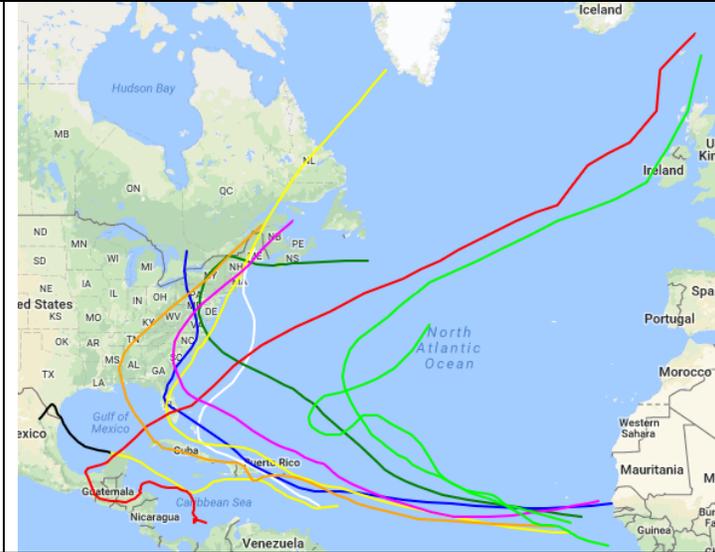
Cluster 4



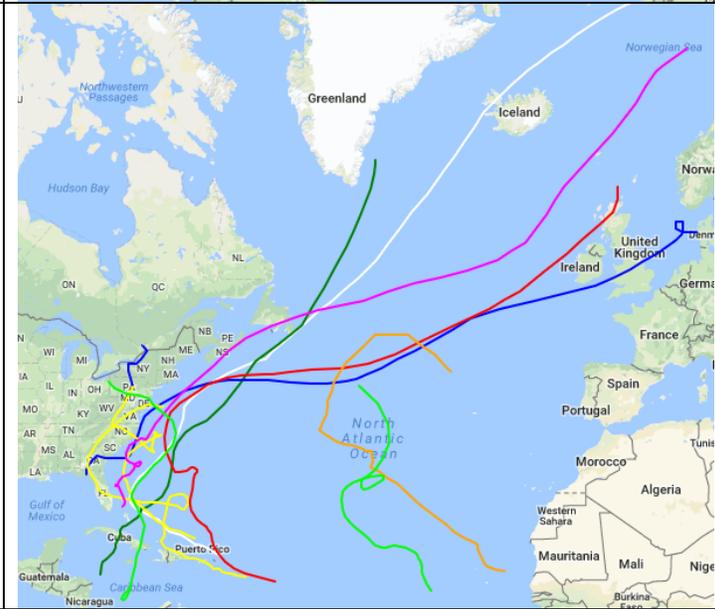


Non-sequential match minScore =15		
Cluster Name	Images plotted on map	Included Hurricanes
Cluster 1		

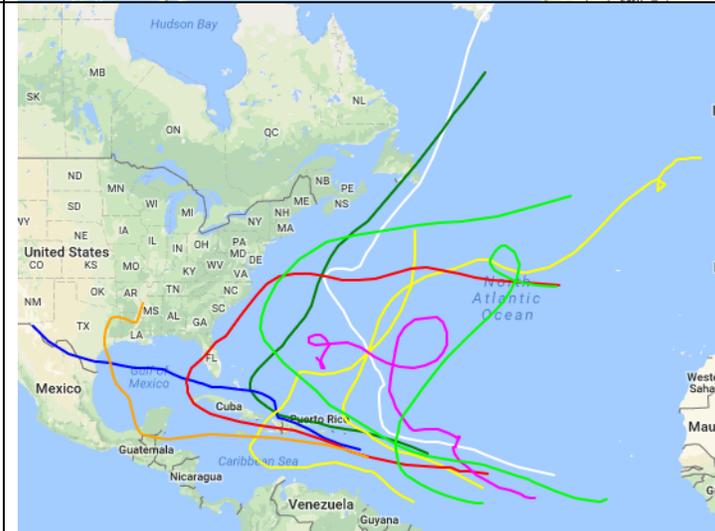
Cluster 2

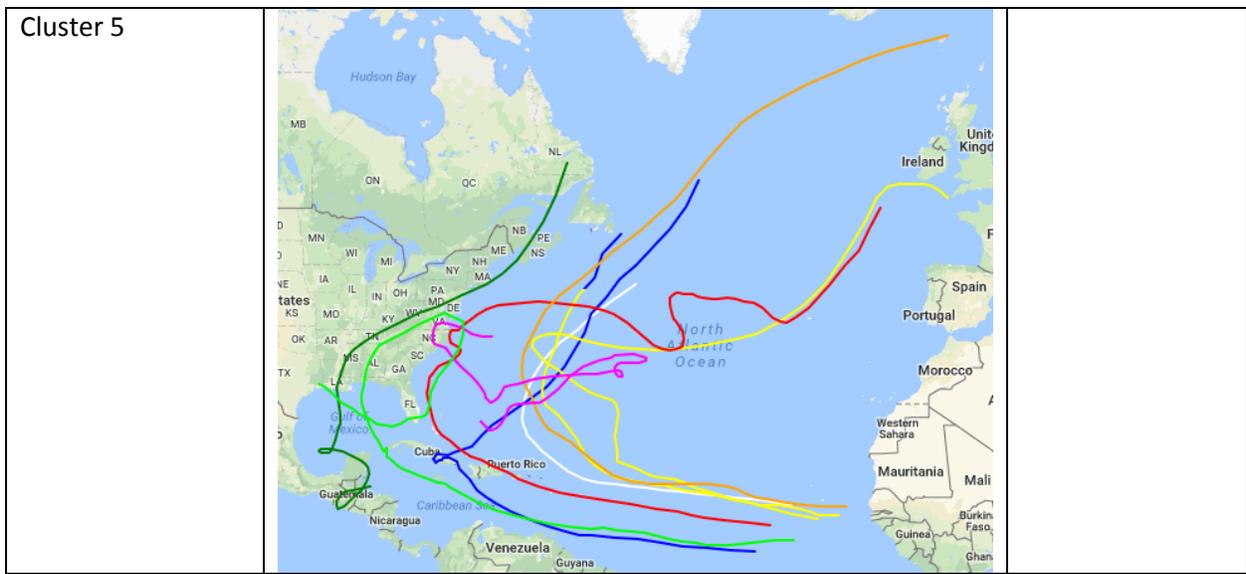


Cluster 3



Cluster 4





Non-sequential match minScore =20		
Cluster Name	Images plotted on map	Included Hurricanes
Cluster 1		

<p>Cluster 2</p>		
<p>Cluster 3</p>		
<p>Cluster 4</p>		<p>NOEL_16_1995, MATTHEW_14_2016</p>

Cluster 5		UNNAMED_04_1887, UNNAMED_03_1888
-----------	---	-------------------------------------

The clustering output is found in the ClusterGrps.txt and ClusterLog.txt files in the HurricaneProject folder.

We expanded our algorithm range to include other string comparison algorithms, leveraging the F23 library in .NET. The first algorithm tried is the **Jaro-Winkler algorithm**, which, when applied to two strings, calculates the minimum number of single-character transpositions required to change one word into the other⁷. Although this algorithm can match two similar paths, it fails to cluster two similar hurricanes, such as NOEL_16_1995 and MATTHEW_14_2016, two hurricanes which the modified LCS were able to identify as belonging to the same cluster. The Jaro-Winkler calculation results in few clusters being formed, and the most populous cluster contains less than 5 hurricanes. The second algorithm tried uses the **Levenstein distance**, which calculates the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other, in order to determine the similarity between two strings⁸. This algorithm produced inferior results when compared to the LCS and modified LCS implementations, most likely because deletions are penalized in these algorithms, but in the LCS algorithm, deletions are not penalized.

Returning to the LCS and modified LCS algorithms, we set out to determine the usefulness of the star calculus in clustering the hurricanes. Although we thought that the star calculus and forgiveness factor in the string matching would produce similar results, we find that the star calculus sectors of 36 sectors in a 360-degree representation greatly facilitates clustering. Reducing the sectors to 360, or the equivalent of converting the bearing itself to an integer, few clusters formed at all.

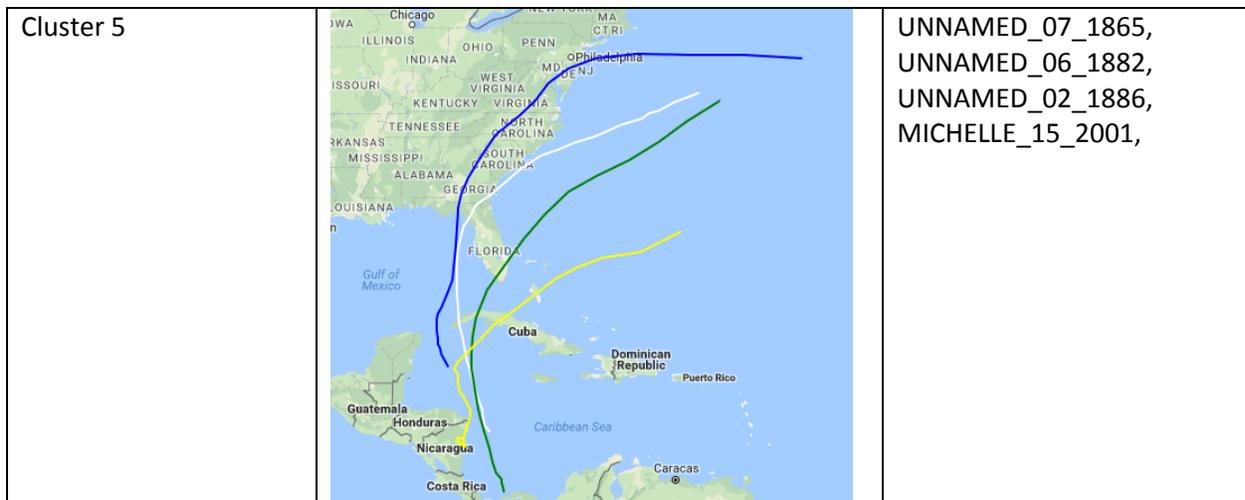
Using the pure LCS with a forgiveness factor of plus or minus 2 and a minimum LCS value of 12 produces good results:

True LCS (allowing +/-2 on match with 36 sector star calculus and		
---	--	--

⁷ For definition of Jaro-Winkler distance, see the Wikipedia article found here https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance.

⁸ For definition of the Levenshtein distance, see the Wikipedia article found here https://en.wikipedia.org/wiki/Levenshtein_distance

<p>minScore = 12 on LCS value</p>		
<p>Cluster 1</p>		<p>UNNAMED_03_1872, UNNAMED_09_1899, UNNAMED_07_1878, GUSTAV_08_1990, UNNAMED_03_1883, UNNAMED_04_1884, UNNAMED_04_1904, UNNAMED_01_1921, LISA_13_2004, UNNAMED_17_1933</p>
<p>Cluster 2</p>		<p>UNNAMED_06_1893, UNNAMED_02_1927, UNNAMED_02_1940, UNNAMED_15_1967, UNNAMED_05_1899, BAKER_02_1950, UNNAMED_01_1900, UNNAMED_02_1900,</p>
<p>Cluster 3</p>		<p>UNNAMED_02_1880, LUIS_13_1995, UNNAMED_09_1888, UNNAMED_03_1928, UNNAMED_02_1896,</p>
<p>Cluster 4</p>		<p>UNNAMED_04_1880, ISABEL_13_2003, UNNAMED_04_1888, UNNAMED_02_1890, INEZ_09_1966,</p>



Viewing the results, there is some support for the conclusions found in the Lodangco, et al, paper, which cluster the hurricanes based on genesis location. We can see from the above images that Cluster 5 hurricanes all originate near Central America. However, we see that Cluster 1 hurricanes can also originate near Central America. For this reason, we suggest that the hurricane path is not geographically dependent as the Lodangco paper suggests, but rather, dependent on other variables which can cause similar behavior in disparate geographical locations.

Further work can include exploring other clustering algorithms beyond the hierarchical clustering algorithm and using other dynamic programming approaches beyond the Longest Common Subsequence approach. Another opportunity for research is to take into account the presence of other weather systems in the vicinity of a hurricane to see how that affects the path it takes. In the direction of eye-wall replacement, it would be interesting to leverage the Google static API to determine when a hurricane is close to land and correlate the eye-wall replacement with proximity to land. It will be interesting to explore the geographical location of the hurricane in relation to the clustering. The star-calculus approach taken, and the sequences which were derived from the paths do not account for the global position of the hurricane. Finally, the time series nature of the hurricanes could be explored to see whether, over time, hurricanes trend toward a certain cluster behavior, such as following the Atlantic seaboard along the coast more so in recent years as a possible hypothesis. Much further work could be supported by this very interesting data set.