DATA301 Final Report

Cameron Bodger
78993602

Abstract

This report proposes to develop an understanding of geopolitical similarities and alliances by analyzing emotional tone from the GDELT projects Global Knowledge Graph dataset. By clustering countries by their media's reaction to the Syrian government, the rebel forces and the Kurdish rebels it should be possible to group each country into their international alliances of the time.

How similar was each country's sentiment towards the various fighting factions of the Syrian Civil War? Specifically, can different country's similarity of tone towards the Syrian army, the Rebel forces and the Kurdish forces be clustered to determine geopolitical alliances?

To answer this, histogram vectors of the tone will be created, then clustered using the bisecting K means algorithm.

Introduction

The Syrian Civil War was, and remains, one of the most important geopolitical events of the 21st century so far. Beginning from the Arab Spring crises in 2011, it soon devolved into a proxy war with numerous countries overtly or covertly involved. The Global Database of Events Language and Tone (GDELT) hosts several massive datasets recording human events. The Global Knowledge Graph (GKG) dataset contains records of machine learned analysis of news articles, including who is involved, where, and the emotion portrayed. Unfortunately, the dataset begins in November 2013, with an improved version beginning in September 2014. Since data did not exist for the start of the war, 2015 was chosen to be analyzed as it represented new intensity in the conflict. GKG version 2 records country codes in the column V2Locations, which can be subset to containing 'SY' for Syria. The column 'V2Themes' contains codes for the type of event represented in the article, allowing for further subset based on themes such as 'ARMEDCONFLICT', 'TERROR', 'MILITARY'. The GKG column 'V2Tone' records emotional dimensions of the article such as 'Tone', 'Positive Score', 'Negative Score', and so on. 'Tone' is the average of the positive and negative score, and since articles about war will largely be negative, is the most appropriate measure.

Bisecting K means algorithm combines top down clustering and K means clustering. Top down hierarchical clustering involves splitting clusters from the top down based on the measure the dissimilarity between clusters. K means clustering begins with predetermined K number of randomized centroids, which are used to cluster the data based on distance. The centroids are continually rebased towards the mean distance of their cluster until the mean no longer changes.

Bisecting K means uses both, it begins like hierarchical with one large cluster, then uses K means with k=2 to divide it into two clusters. Which one of these to split is decided by which of these has the largest sum of the square distance, then this process is repeated until K clusters have been created.

Since K is predetermined, it is necessary to evaluate the optimal K. For this project an elbow plot was used, with sum of the squared distance between points and centre of the clusters plotted over k, for each k from 2 to 10. At the 'joint' of the elbow, the gained cost begins to increase slower for each added cluster, so the K at the joint will be optimal.

Experimental Design and Methods

Data is from Feb 18 2015 to April 23 2015. Data was taken in by the day, then processed to include only media url, organizations, locations, themes and tone, then resaved as csv. This allowed it to be stored in Google Drive, then put into a dataframe. The tone takes on values from -10 to 10, this was grouped into histogram buckets then each row was normalized. A bisecting k means model was fitted for k = [2, 6], then the cost of each model was plotted over k to determine the optimal k.

Libraries:

Most datawrangling was done using pyspark Dataframes and RDD's, also numpy and pandas arrays and matrices.

gdelt, for interfacing with the dataset.

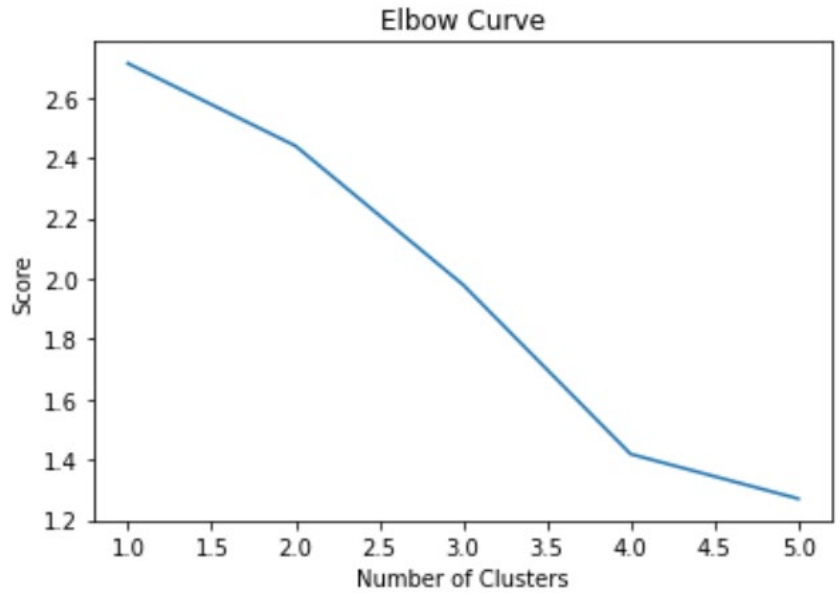
Normalizer from pysparks ml features was useful for easily normalizing rows.

BisectingKMeans performed the clustering and clustering analysis.

pylab for plots.

Results

From the elbow plot, it may be seen that 4 clusters are optimal.



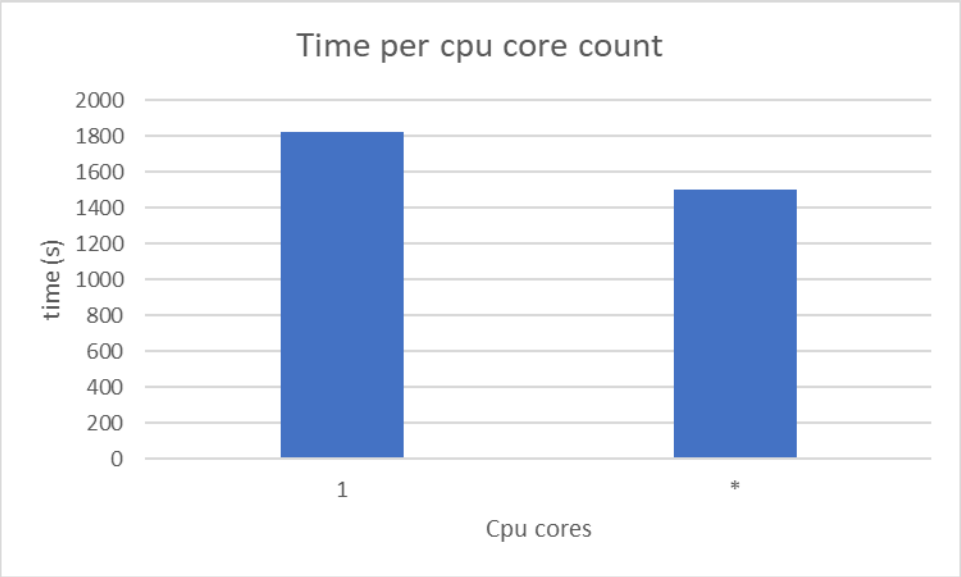
Bisecting K means with K=4 produced the following clusters:

0) Azerbaijan, Egypt

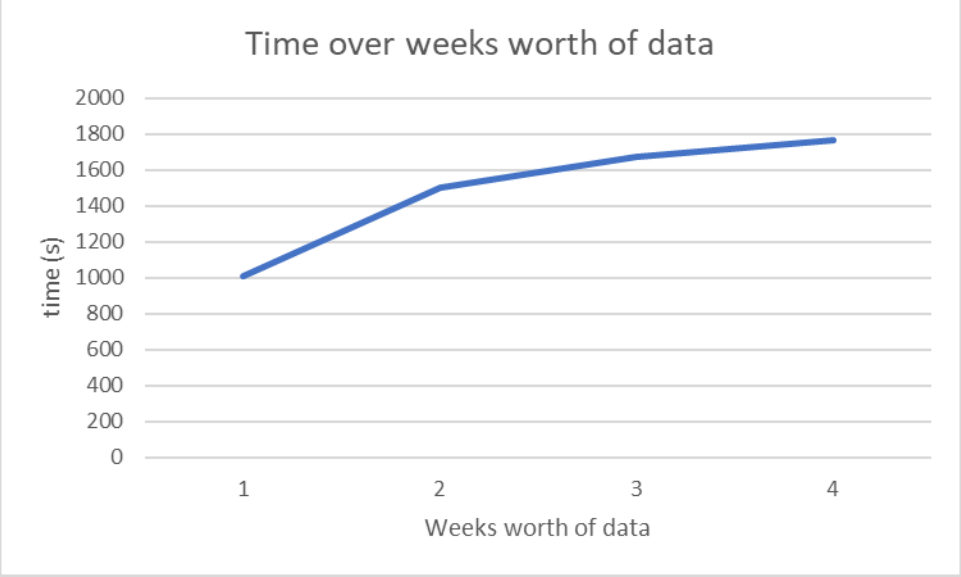
1) Canada, American Samoa, Malaysia, Pakistan, India
2) United Kingdom, United States, Turkey, United Arab Emirates
3) Serbia, Iceland, France, Iraq

As the cluster vectors are very large, they have not been plotted. The cluster centres have been included in the appendix.

It appears that yes, to an extent, geopolitical alliances can be seen by clustering them by the tone of the articles their country generates. Cluster 2 represents the international community that (at the time) strongly opposed to the Syrian government, whereas both Azerbaijan and Egypt strongly supported the Syrian government. However, the other two clusters are murkier. It appears that cluster 3 is related by not having much data, but the data they did have was similar, and I suspect that cluster 1 represents general negative tone towards the war in general.



Using only one cpu core significantly slowed down the experiment.



Conclusion

I believe it to be an interesting result, particularly the cluster with United Kingdom, United States, Turkey and United Arab Emirates, as they were known to be allied and pro-rebel forces at the time. Azerbaijan and Egypt is also an interesting cluster, as they were known to support the Syrian government. It is surprising to see India and Pakistan in the same cluster. It is disappointing that Russia and China were not included in the dataset.

The implications are that this technique could be used for other events of geopolitical importance to tease out alliances and similarities. As the GDELT project is practically created in real time, this could be used to create daily snapshots of the fast-changing world of geopolitics.

I am disappointed in the amount of data that I was able to be tease out of the dataset in relation to this topic. If I could master its secrets, I would like to continue this question, but based on the war in Yemen. There are far fewer media articles written about the Yemeni civil war, so I think it would add another layer of interest for me, as not just could the clusters represent alliances but also show which countries are actually reporting on it.

Critique of Design and Project

As predicted in the project outline, lack of knowledge in both the Syrian Civil War and the GKG dataset were extremely prohibitive. This is very apparent in the amount of related data that was acquired. The organizations column was difficult to separate the groups out of, seemingly rarely mentioned by name. Trying to separate them based on themes was no more useful and no less difficult, with so many themes and little documentation of what they meant. A lot of time was wasted on this, and resulted in frustratingly small amounts of useful data. When it became apparent that the option to use coverage=False in pythons gdelt library (15 minutes of articles per day) would not give enough points to be useful, coverage was set to True so entire days' worth were downloaded. This yielded better results, but with 2.6 gb files per day, was time and resource intensive. It also had to be babysat lest it use up Colab's ram and crash, needing to be rerun. This led to only a few months of data being able to be acquired, and a tricky design decision had to be made about whether to use temporally spaced, but smaller data, or more data for a shorter time period. Neither of these were ideal, however more data for a shorter time period was chosen.

Reflection

The Mining of Massive Datasets book was very useful, as was the Data301 lecture notes and lab examples. I lived inside of the pyspark documention for the duration of the project, particularly the BisectingKMeans page, and also read the GKG codebook a dozen times. The Wikipedia page for the Syrian Civil War was useful for finding names of the organizations. I learned a lot about both hierarchical and Kmeans clustering, as well as cosine similarity. Numpy and Pandas were interesting to use as well, I haven't spent much time with them but I liked how nicely they played with each other and the pyspark libraries.

References

http://data.gdeltproject.org/documentation/GDELT-Global_Knowledge_Graph_Codebook-V2.1.pdf

https://medium.com/@afrizalfir/bisecting-kmeans-clustering-5bc17603b8a2#:~:text=Instead%20of%20partitioning%20the%20data,until%20k%20clusters%20are%20obtained.

http://www.mmds.org/

https://en.wikipedia.org/wiki/Belligerents_in_the_Syrian_Civil_War#Opposing_forces

https://blog.gdeltproject.org/introducing-gkg-2-0-the-next-generation-of-the-gdelt-global-knowledge-graph/

https://spark.apache.org/docs/latest/api/python/pyspark.ml.html#pyspark.ml.clustering.BisectingKMeans

Grey Harris and Reed Earl provided advice for this project.

Appendix

Cluster centres

```
[0.         0.         0.         0.11568898 0.         0.
 0.06525739 0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.06525739 0.
 0.18094637 0.06525739 0.91955763 0.19577216 0.06525739 0.
 0.         0.06525739 0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.11568898 0.11568898 0.06525739
 0.         0.06525739 0.         0.11568898 0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.        ]
[0.         0.         0.01267056 0.0776195  0.06074894 0.01477056
 0.07708585 0.02534112 0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.04556508 0.12828721 0.19639143
 0.24958356 0.35605822 0.58814781 0.19596036 0.55023523 0.11241332
 0.08287306 0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.01267056 0.03318663 0.03569992 0.1141395
 0.10019607 0.0704529  0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.        ]
[4.40743974e-04 1.32223192e-03 1.49419585e-02 5.98880397e-02
 3.95369479e-02 4.35903168e-02 1.34766865e-01 4.95964417e-02
 3.05224841e-02 1.14160067e-02 8.81487948e-04 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 5.46767295e-02 1.11605258e-01 2.35854760e-01
 2.90058974e-01 5.79899737e-01 3.55152153e-01 5.10392202e-01
 2.21974722e-01 1.54076422e-01 1.14850209e-01 9.42092676e-03
 9.42092676e-03 1.14160067e-02 1.76297590e-03 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 3.74264046e-02 1.75864224e-02 2.67319085e-02 1.51505984e-02
 5.13143199e-02 1.05778554e-02 3.03563628e-02 7.21720689e-03
 0.00000000e+00 5.45423099e-03 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00]
[0.         0.02033401 0.13440814 0.07875326 0.12109528 0.25550343
 0.24958635 0.12535205 0.13865122 0.         0.02033401 0.
 0.02200801 0.         0.         0.         0.         0.
 0.         0.         0.         0.02625109 0.02033401 0.04658509
 0.10926112 0.27269842 0.52563278 0.4729216  0.21427916 0.32095752
 0.04401603 0.06602404 0.         0.02033401 0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.02200801 0.02200801 0.02200801 0.04401603
 0.17842417 0.11004007 0.04401603 0.02200801 0.0482591  0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.        ]
```