Cipher Text Clustering

Preview

In this report I examined the results of K-means clustering on encrypted cipher text through various encryption techniques. The techniques used were imported from PyCipher external library, which includes: simple substitution, column transposition, and caesar. When encrypting, white space and punctuation are automatically removed and results in just the text cipher.
Simple substitution is an encryption method that consists of a key of alphabet letters that are jumbled, e.g ‘AJPCZWRLFBDKOTYUQGENHXMIVS’, which encrypts according to the mapping of a regular alphabet.
Column transposition is a fractionating cipher which utilizes a key word that constructs a table of word length. This table is filled from left to right and top down of the text to be encrypted with padding to complete the nxn table. The key word is then rearranged alphabetically to form the cipher.
Caesar encryption technique is a simple substitution technique where letters in the alphabet are shifted by a certain amount and mapped to the original placings.

Preprocessing

In order to perform clustering, data was generated through PyCipher which contains a library of encryption techniques. To accumulate text to encrypt, another library from ‘essential generators’ was used. This library randomly generates sentences to encrypt and words to use as keys. From there, we use a loop to generate ten thousand lines of ciphers for each encryption technique, resulting in a total of thirty thousand lines amongst three classes.
There is a consideration of variation when utilizing different plain text amongst different classes versus same text shared between classes. This is taken into consideration in our experiments

Tuning

The tuning parameters to take into consideration are the number of clusters, number of iterations, and initial clusters.

The numbers of clusters explored range from two to five. Although there are cases where the number of clusters and classes do not match, changing cluster amounts can help explore observations of the data.

Number of iterations and initial clusters are defaulted through SciKit's KMeans. The initial clusters are randomly chosen from the data and run through up to three hundred interactions until finalized. Then, new initial clusters are randomly chosen nine more times, totaling ten iterations, from which we choose the best iteration.

Evaluation of cluster quality is done through purity, an external evaluation criteria. Purity is the percent of total number of objects that are classified correctly. Since KMeans clustering is an unsupervised learning technique, when we say 'correct classification', we imply that each cluster has identified a group of objects belonging to the same class. This means we label our data accordingly, however when we train the algorithm the labels are excluded. The purity score is computed through SciKit's contingency matrix where we take the sum of maximum along each row divided by the total number of observations.

Experimentation

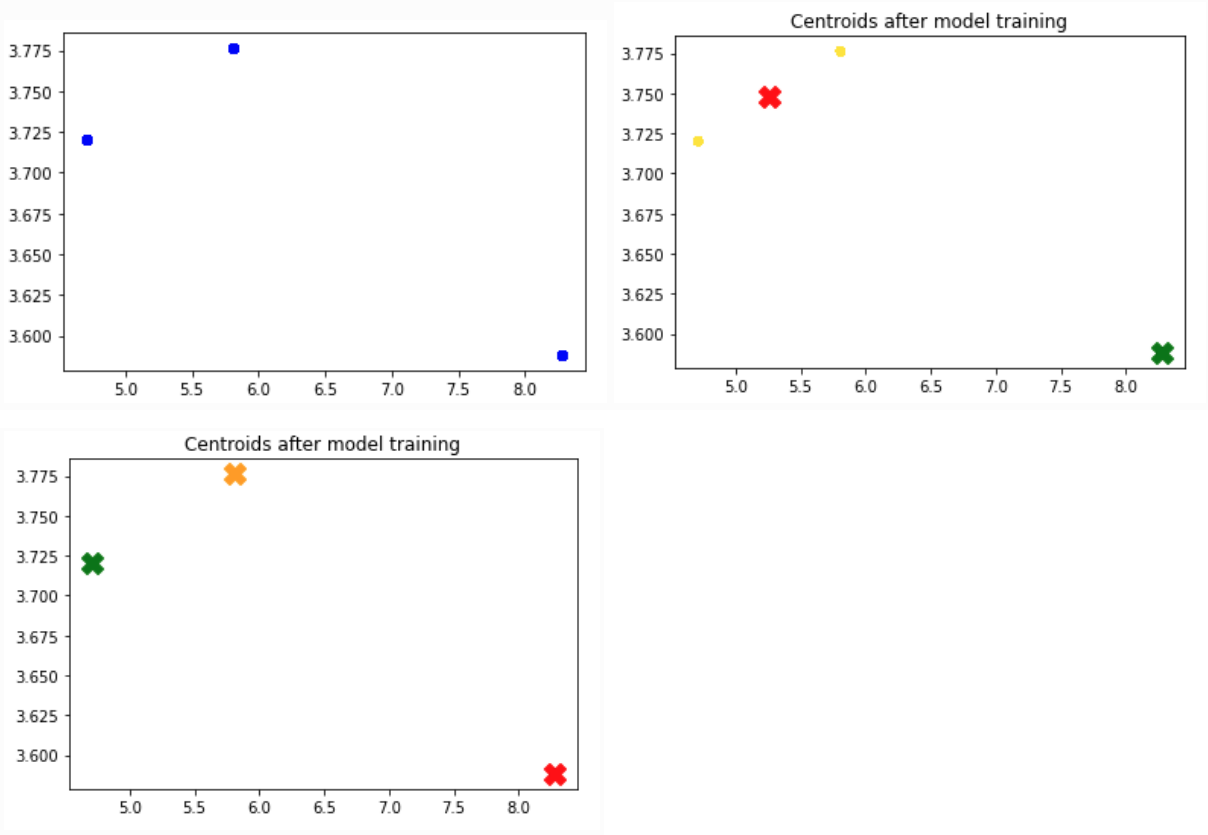
There were four experiments that were run. The first is when we initialize a single text that is used through all encryptions but with randomly generated keys. The second is the opposite, where a single key is generated for ten thousand iterations of random texts. The third is when we use a single key like in the second run, but we share the same text list amongst three cipher classes. The last is when we randomize keys like in experiment one, but share text, similar to experiment three.

Dimension 2: IOC and Entropy

Index of coincidence is the technique of comparing texts by counting the number of times that identical letters appear in the same position in both texts. This is useful for determining cipher techniques because its values do not change when scrambled by the same single-alphabet substitution cipher.

Entropy is used as a measure of unpredictability-the average level of information or uncertainty inherent is the variable's possible outcomes.

Experiment One: One Text (unique amongst classes), Randomized Keys

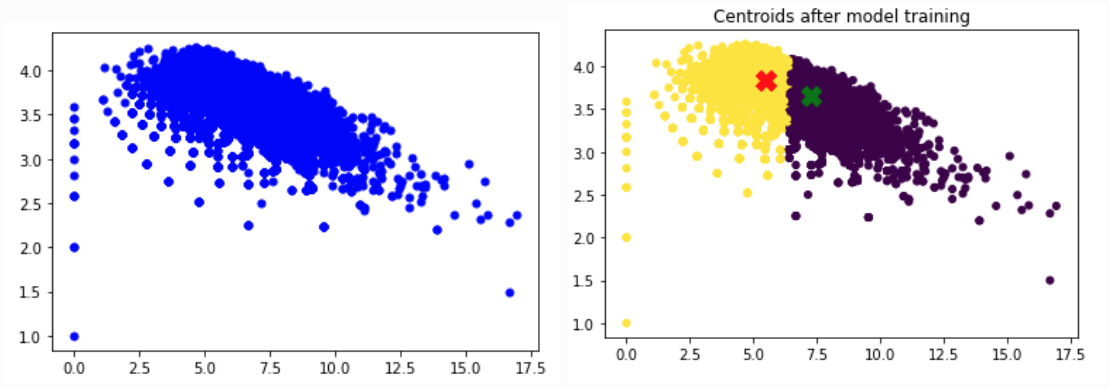


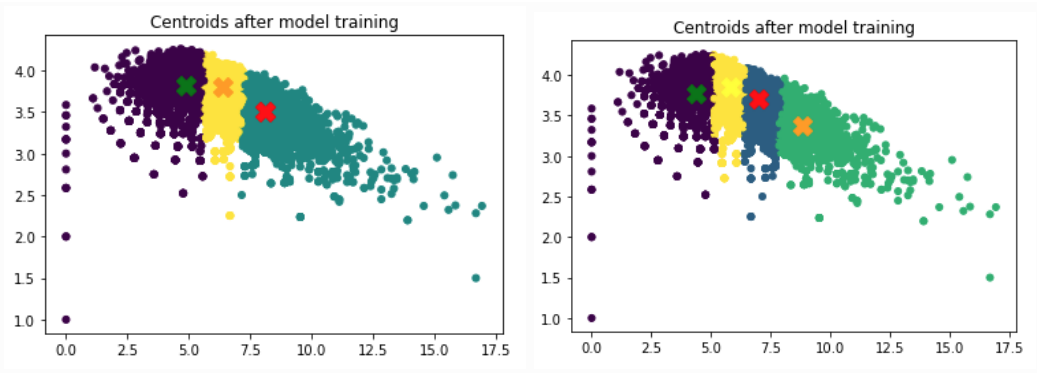
Graph 1 (Top left): This is the graph that displays points prior to clustering

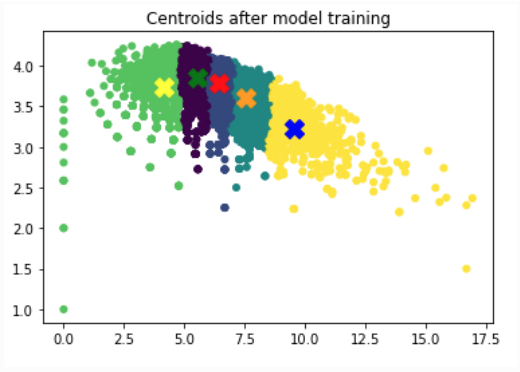
Graph 2 (Top right): This graph is KMeans with two clusters

Graph 3: (Bottom left): This graph is KMeans for three to five clusters (i.e same graph k=3-5)

What is immediately noticeable is that the points are centered around three distinct points. The purity score of the resulting graphs are all 1.0; proving the validity of the index of coincidence. Another noticeable fact is when K=2, one clusters' is centered in between two class points







The graphs generated were all very similar for experiments two, three, and four.

Graph 1 (Top left): This is the graph that displays points prior to clustering

Graph 2 (Top right): K = 2

Graph 3 (Middle left): K = 3

Graph 4 (Middle right): K = 4

Graph 5 (Bottom left): K = 5

Experiment Two: One Key, Randomized Text (unique amongst classes)

In this experiment, the points are distributed much more than compared to experiment one. This is because the text lengths are varied which results in different IOC and entropy. What remained consistent was the purity score, meaning the model was successful in clustering the same classes. As we increase clusters, the graph divides the center cluster into pieces that match the number of K accordingly.

Experiment Three: One Key, Shared Text

In experiment three, we observed the same distribution as in experiment two. The difference was in the purity score. For all K clusters, the purity score was 0.333. This is exactly one third, which can be expected since the points generated were all very similar in IOC and entropy amongst the three classes. When points are very similar, the probability of guessing the correct class is one third, which is as the model reflects.

Experiment Four: Shared Text, Randomized Key

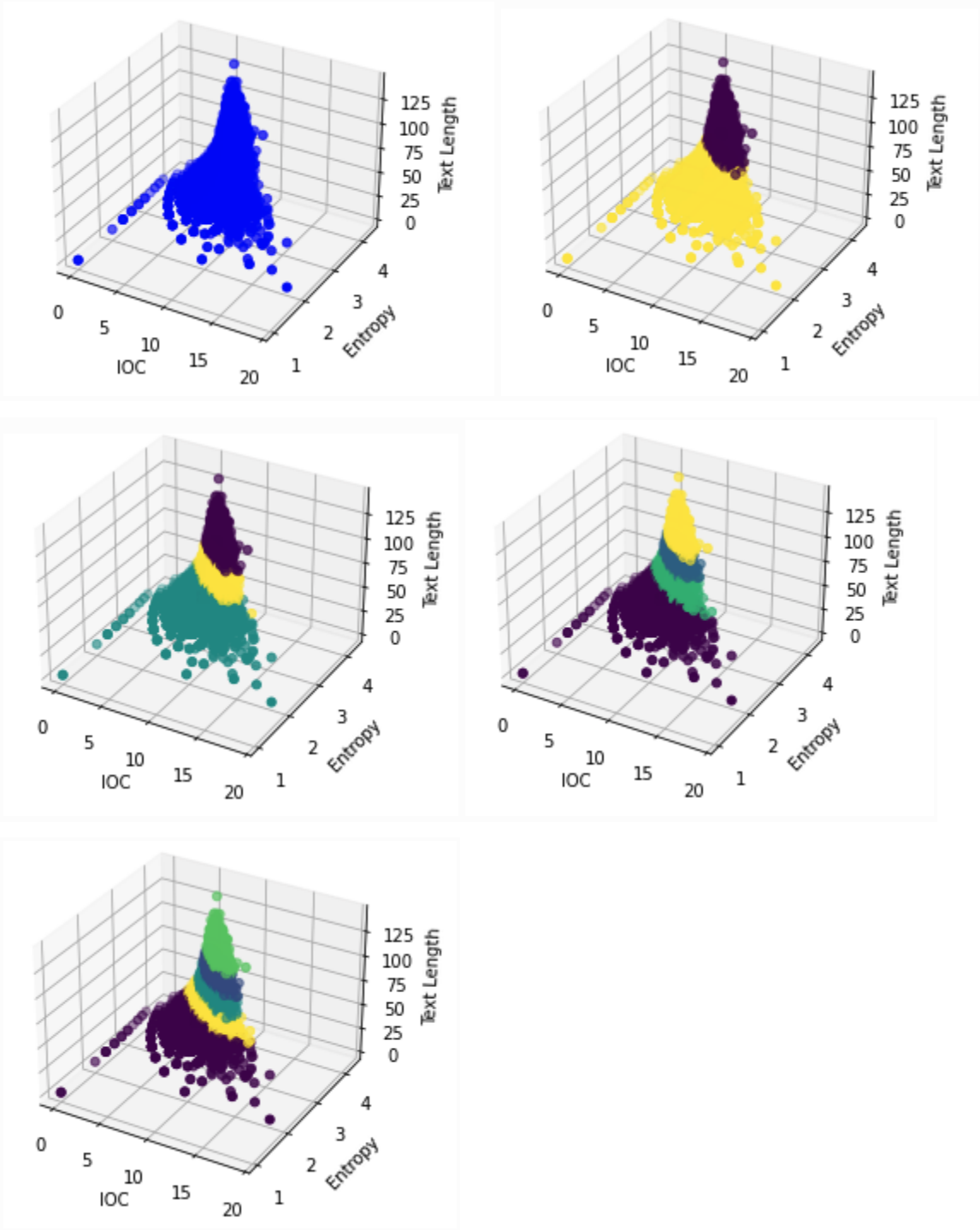
This experiment had basically the same results as in experiment three. The purity score was also one third and graphs/clusters formed were essentially identical.

The distinction in results between experiment two and experiment three/four would be the change in randomized texts. When using the same texts shared amongst all encryption techniques, it becomes significantly harder to distinguish through IOC and entropy. The points are actually so close that in some cases the difference could not be contained in python's built-in numeric type. In instances where difference was evidence, it would be in the decimal fourteen places after the dot. When we used randomized texts across all classes, the distinction is much more clear.

Increasing Dimensionality

Since length of cipher texts in shared lists have an impact on IOC and entropy, what happens when clustering takes length into consideration as well. This time, only two experiments will be run. This is because experiment one uses a single text so it is pointless to include, and

experiment three and four are almost the same besides the use of random keys. Therefore, we run only two scenarios, namely experiment two and four from the previous section.



The graphs reflect the same for both experiments five and six.

Graph 1 (Top left): This is the graph that displays points prior to clustering

Graph 2-5 (Left to right, top down): K= 2 to 5

Experiment Five: One Key, Randomized Text (unique amongst classes)

Contrary to experiment two, when adding length as a factor into KMeans, the purity score decreases to about an average of 0.338. This means that length is not a good factor to consider when clustering cipher text, despite the impact it has on IOC and entropy.

The purity is not quite one third, meaning the clustering is not exactly similar to guessing as seen in experiments three and four.

Experiment Six: Shared Text, Randomized Key

This experiment is the same as what was seen in experiment four, but on a 3D plane. The purity score is the same at 0.333 or one third, and the clusters are formed as a split horizontally across the points.

Conclusion

In our experiments we found clustered three distinct encryption techniques (simple sub, column transposition, and caesar). The generated data showed the ability of IOC and entropy when used for classifying ciphers to their corresponding encryption techniques. But this was only reflected when using distinct texts across each technique.

For future experiments, we can expand the number of encryption techniques to show the similarity of techniques that refer to each other. For example, the Vigenre cipher is a basis for other techniques like Autokey or Beaufort. Therefore we can test to see the similarity of clusterings between such techniques. We can also expand the dimensionality of data by implementing log probabilities from HMM training based on the english language, etc. This could result in better purity scores/clusterings as our model would have more information to base clusters off of.