Lauretta Take home ML Engineer test

Test taker - Leela Srija Alla

Question 2 - Person Re-Identification

Code Implementation

The following steps were followed in the code implementation:

Data Loading and Preprocessing:

- Loaded the final_detections.json file.
- Extracted feature vectors and detection IDs.
- Applied L2 normalization to the feature vectors to prepare them for cosine similarity.

Spectral Clustering:

- Performed Spectral Clustering using cosine similarity as the affinity metric.
- Ran the clustering for 10 iterations, each with a different random seed (random_state).
- For each iteration, the Silhouette Score to evaluate clustering quality.

Best Result Selection:

- Tracked the clustering result with the highest Silhouette Score and selected it as the best clustering solution.

Output Generation:

- Grouped detection IDs based on the best clustering result and saved the output in prediction.json.

Real-Time vs. Batch Processing

The current implementation assumes that all detection data is available at once, which is appropriate for batch processing. However, if the task required real-time detection processing (i.e., as new detection data is streamed in), the approach would need to be adjusted:

Real-Time Adjustments

Incremental Clustering: Instead of clustering the entire dataset at once, I would implement incremental clustering where each new detection is compared against the existing clusters based on cosine similarity of the feature vectors. The new detection would either be added to an existing cluster or form a new cluster if no match is found.

Efficient Data Structures: For real-time processing, I would use efficient data structures such as KD-trees or Locality Sensitive Hashing (LSH) to quickly find the nearest neighbors in the feature space, which would speed up similarity comparisons.

Batch Processing Adjustments

If all detection data were available from the beginning (as in the current setup), I would optimize the clustering algorithm by using methods that take global information into account. For example:

Hierarchical Clustering: This algorithm could be used to merge detections into clusters based on pairwise similarity.
Spectral Clustering: This method could help find non-linearly separable clusters, leveraging the similarity matrix of all detections.

Notes and Considerations

Imperfect Tracking IDs

The tracking IDs provided in the dataset are generated by an algorithm and are not perfectly reliable. This means that two detections with different tracking IDs from the same camera could still be the same person. I accounted for this by focusing primarily on the similarity of feature vectors rather than relying on tracking IDs for clustering.

Overlapping Camera Views

The dataset includes cameras with overlapping fields of view. This means that two detections with the same frame ID but from different cameras could be the same person. Conversely, two detections with the same frame ID and camera ID cannot be the same person. I accounted for this by ensuring that detections with the same camera and frame ID were treated as distinct entities, while still allowing for cross-camera comparisons.

there are internal evaluation metrics that assess the quality of clusters based on the data itself, without needing external ground truth labels. These metrics rely on the inherent properties of the clusters, such as cohesion (how tight the clusters are) and separation (how well-separated different clusters are). Here are some commonly used evaluation metrics that can be applied without using ground truth labels:

Evaluation Metric

Silhouette Score

Description: Silhouette score measures how similar a point is to its own cluster (cohesion) compared to other clusters (separation). It ranges from -1 to 1, where:
A value close to 1 indicates that points are well-matched to their own cluster and poorly matched to other clusters.

A value close to 0 indicates that points are on or near the decision boundary between two clusters.
A value of -1 indicates that points are potentially in the wrong cluster.

Formula - b(i) - a(i) / max(a(i), b(i))
Where a(i) is the average distance between the point and other points in the same cluster.
b(i) is the average distance between the point and points in the nearest neighboring cluster.