Okay, let's break down the cluster comparison calculations in both scripts, focusing on correspondence, metrics, and Precision/Recall/F1. This should give you a clear explanation for your professor.

\*\*Core Concept: Comparing Two Clusterings\*\*

Both scripts aim to compare two different ways of grouping the \*same set of items\* (foods).

\* `compare\_clusters\_embedding\_presence.py`: Compares clustering derived from \*\*food embeddings\*\* (using various algorithms like K-means, Hierarchical, DBSCAN) against a \*\*ground truth clustering\*\* (loaded from `filtered\_flavordb\_clusters\_683.txt`).

\* `compare\_cluster\_files\_flavordb\_clusterspresence.py`: Compares \*\*two ground truth-style cluster files\*\* directly (e.g., `filtered\_flavordb\_clusters\_683.txt` vs. `average\_linkage\_clusters\_reclustered\_gemini\_683.txt`).

The fundamental first step in both is to identify the \*\*common food items\*\* present in both clusterings being compared. The analysis only considers these common items.

\*\*1. Cluster Correspondence Calculation\*\*

\* \*\*What it is:\*\* It measures how the clusters in one set (e.g., predicted embedding clusters) overlap with the clusters in the other set (e.g., ground truth presence clusters).

\* \*\*How it's calculated (in both scripts):\*\*

1. \*\*Confusion Matrix (Raw Counts):\*\* A table is created where rows represent the clusters from the first file (let's call it "True/Ground Truth") and columns represent clusters from the second file ("Predicted/Comparison"). Each cell `(i, j)` contains the \*number\* of food items that are in True Cluster `i` AND Predicted Cluster `j`.

\* Implementation: `pd.crosstab(df['true'], df['pred'])` or `pd.crosstab(df['cluster1'], df['cluster2'])`. This raw matrix is printed by the `clustering\_precision\_recall\_hungarian` function in both scripts.

2. \*\*Percentage Matrix (Row Normalized):\*\* The raw counts matrix is often normalized to show percentages. The scripts primarily calculate row-normalized percentages: "For items in True Cluster `i`, what percentage fall into each Predicted Cluster `j`?".

\* Implementation: `correspondence.div(correspondence.sum(axis=1), axis=0) \* 100`.

3. \*\*Analysis (`analyze\_cluster\_correspondence` function in both):\*\*

\* This function takes the true labels, predicted labels, and food names.

\* It calculates and prints the \*\*percentage matrix\*\* described above.

\* It then finds the "best matching" predicted cluster for each true cluster by finding the column (predicted cluster) with the maximum percentage overlap for that row (true cluster). It prints these best matches.

4. \*\*Saving:\*\*

\* `compare\_cluster\_files...`: Saves the \*\*overall percentage correspondence matrix\*\* (returned by `analyze\_cluster\_correspondence`) to a CSV file (e.g., `cluster\_correspondence\_...\_gemini.csv`).

\* `compare\_clusters\_embedding\_presence...`: \*Doesn't\* save the overall correspondence matrix directly. Instead, within `analyze\_top\_models`, it calls `analyze\_cluster\_correspondence` for the top-performing models (based on Precision in the latest version) and saves \*those specific percentage correspondence tables\* to separate files in the `correspondence\_tables` directory.

\*\*2. Clustering Evaluation Metrics\*\*

Both scripts calculate standard metrics to quantify the similarity between the two clusterings, considering only the common food items.

\* \*\*Metrics Used (calculated in both scripts using `sklearn.metrics`):\*\*

\* \*\*Adjusted Rand Index (ARI):\*\* Measures the similarity between two clusterings, correcting for chance. A score of 1 means perfect agreement, while a score near 0 means the agreement is what you'd expect by random chance. It considers all pairs of samples and counts pairs that are assigned to the same or different clusters in both clusterings.

\* \*\*Normalized Mutual Information (NMI):\*\* Measures the mutual information shared between the two clusterings, normalized to be between 0 (no mutual information) and 1 (perfect correlation). It's based on information theory concepts.

\* \*\*Homogeneity:\*\* Measures whether each predicted cluster contains only members of a single ground truth class. Score of 1 means perfect homogeneity.

\* \*\*Completeness:\*\* Measures whether all members of a given ground truth class are assigned to the same predicted cluster. Score of 1 means perfect completeness.

\* \*\*V-Measure:\*\* The harmonic mean of Homogeneity and Completeness. A score of 1 indicates perfect homogeneity and completeness.

\* \*\*Internal Metrics (calculated \*only\* in `compare\_clusters\_embedding\_presence.py`):\*\*

\* \*\*Silhouette Score:\*\* Measures how similar an object is to its own cluster (cohesion) compared to other clusters (separation). Ranges from -1 to 1, where higher values indicate better-defined clusters. Calculated \*only\* on the embedding data (`X\_flat` or `X\_scaled`) and the predicted labels, \*without\* using the ground truth clusters.

\* \*\*Calinski-Harabasz Score:\*\* Ratio of between-cluster dispersion to within-cluster dispersion. Higher scores generally indicate better-defined clusters. Calculated \*only\* on embedding data and predicted labels.

\* \*\*Davies-Bouldin Score:\*\* Measures the average similarity ratio of each cluster with its most similar cluster. Lower scores indicate better separation between clusters. Calculated \*only\* on embedding data and predicted labels.

\*\*3. Precision, Recall, and F1-Score Measurement\*\*

These metrics are common in classification but need adaptation for clustering because cluster labels are arbitrary (Cluster 'A' in one set doesn't inherently match Cluster '1' in another). The goal is to see how well the \*structure\* aligns after finding the best possible matching between clusters.

\* \*\*Conceptual Definitions (after matching clusters):\*\*

\* \*\*Precision (for a matched pair True Cluster `i` <-> Predicted Cluster `j`):\*\* Of all items assigned to Predicted Cluster `j`, what fraction actually belong to True Cluster `i`? (Purity of the predicted cluster with respect to its matched true cluster).

\* \*\*Recall (for a matched pair True Cluster `i` <-> Predicted Cluster `j`):\*\* Of all items belonging to True Cluster `i`, what fraction were correctly assigned to Predicted Cluster `j`? (Completeness of the true cluster's recovery within its matched predicted cluster).

\* \*\*F1-Score:\*\* The harmonic mean of Precision and Recall for the matched pair \( (2 \* P \* R) / (P + R) \).

\* \*\*Macro-Averaging:\*\* The metrics are typically calculated for each optimally matched pair of clusters, and then averaged across all pairs to get a single "Macro Precision", "Macro Recall", and "Macro F1-Score".

\* \*\*Calculation Methods Used:\*\*

1. \*\*Simple Best Match (`clustering\_precision\_recall` in `compare\_cluster\_files...`)\*\*:

\* \*How:\* For each "true" cluster, it finds the single "predicted" cluster with the highest number of overlapping items (`idxmax` on the correspondence row). Precision and Recall are calculated based \*only\* on these simple best matches.

\* \*Limitation:\* This is a greedy approach. A predicted cluster might be the best match for multiple true clusters, or some predicted clusters might never be chosen as the best match. It doesn't guarantee an optimal \*overall\* pairing.

2. \*\*Hungarian Algorithm (`clustering\_precision\_recall\_hungarian` in \*both\* scripts):\*\*

\* \*How:\* This method finds the \*optimal one-to-one assignment\* between the true clusters and the predicted clusters that \*maximizes the total number of correctly assigned items\* (maximize overlap). It uses the raw count confusion matrix and the `scipy.optimize.linear\_sum\_assignment` function.

\* \*Calculation:\* Precision and Recall are calculated \*only\* for the pairs identified by this optimal assignment. The macro scores are then averaged over these optimal pairs.

\* \*Advantage:\* This is generally considered the more robust and standard way to calculate Precision/Recall/F1 for clustering, as it finds the best global matching.

\*\*In Summary:\*\*

\* Both scripts use confusion matrices (correspondence tables) to visualize overlap.

\* Both calculate standard external metrics like ARI and NMI to measure overall similarity.

\* Both calculate Precision, Recall, and F1 using the preferred \*\*Hungarian Algorithm\*\* method.

\* `compare\_cluster\_files...` \*also\* calculates a simpler (less robust) Precision/Recall/F1 based on greedy best matching.

\* `compare\_clusters\_embedding\_presence...` \*also\* calculates internal metrics (Silhouette, etc.) based on the embedding space itself, independent of the ground truth comparison.

\* The correspondence tables \*saved\* differ: `compare\_cluster\_files...` saves the overall percentage matrix, while `compare\_clusters\_embedding\_presence...` saves specific percentage tables for the top models identified by Precision score.