Technical tips: Step 6

1. How to compute and present quality metrics

The key element of Step 6 is to evaluate the *quality* of your end-to-end retrieval system. For this, we need to be able to say how similar the returned shapes, from several queries, are to the actual query shapes. Since we do not have a ground-truth value of similarity (distance, for example), we use the *class labels* as proxy for that. In other words:

- We pick a shape Q from the database. Let C(Q) be the class label of Q, as listed by the database
- We execute a query with Q as input. Let $R_1,\ldots,R_n$ be the results of the query (here *n* is the query length)
- Let $C(R_1)$ … $C(R_n)$ be the labels of the returned items, as listed by the database
- We compute the query *quality* by assessing any of the evaluation metrics presented in Module 8. These metrics essentially compare C(Q) with $C(R_1)$ … $C(R_n)$. If $C(Q)= C(R_i)$, then R_i is a true positive (TP) for the query, else it is a false positive (FP). The number of false negatives (FN) is given by the size of class C(Q), i.e. the number of shapes in the database having label C(Q), minus the number of true positives.

It is very important to compute the quality metric you select on *all* shapes in the database. Just running a few queries on a few hand-picked shapes Q is biased – you could select shapes for which your system works very well, or very poorly.

Separately, it is important to *present* the results aggregated at different levels. The different shape-classes in your database can yield very different quality metrics: For some class, retrieval can work very easily; for other classes, the converse is true. Hence, it is important to present quality metrics both *per class* and aggregated as a grand total.

Putting it all together, the pseudocode to perform the evaluation is as follows:

Notations

DB: database of shapes, has |DB| shapes in total
query(): your query function reading a shape Q and returning a list of n results $R_1,\ldots,R_n$
M: your desired quality metric
C(Q): class label of shape Q. Takes values in $\{C_1,\ldots,C_k\}$
K: sizes of each class. K(C) = number of shapes of class C
M: average quality values for all class labels. M(C) = average quality value for class C
M_{avg}: overall average quality for entire database.

Algorithm

```
for (Q in DB)
   {R1,…,Rn} = query(Q)
   compute desired quality metric(s) M based on labels C(R1),…,C(Rn) and label C(Q)
   M(C(Q)) += M
   Mavg += M

for (C in {C1,…,Ck})
   M(C) /= K(C)

Mavg /= |DB|
```

The results of the above computation, i.e. the average quality values for all classes M(C) and the grand average M_{avg}, are to be presented as results for the evaluation.

2. How to interpret the result of dimensionality reduction

In Step 5 (bonus), you use dimensionality reduction (DR) to create a visual map of the entire shape database. For this, you project the extracted feature vectors using the t-SNE DR algorithm and create a scatterplot. Next, you color each scatterplot point by the class label of its respective shape.

To make such a scatterplot interpretable, you need to

- Add a *legend*, explaining which color means which class
- Use *colors* that are as different from each other as possible. Do not use e.g. variations of the same hue. A good guideline to choose such colors for categorical data (class labels) is given here: https://colorbrewer2.org

How do you next *interpret* such a plot?

- Dense same-color clusters indicate shape classes with strong intra-class (within-class) similarity. These are desirable, since they indicate that querying shapes in that class will likely yield results from the same class
- Same-color clusters which are far from each other indicate shape classes which are very different. Hence, querying a shape in one cluster will likely not return shapes in the other cluster
- Clusters which overlap, or areas in the plot that show mixed colors, indicate classes (shapes) that are hard to separate from each other. Hence, querying a shape in one of these classes may return shapes in another class.

You can test the above insights: Pick a shape in a dense-packed, same-color, well-separated cluster and query it. You should likely obtain shapes in the same class. Conversely, pick a shape corresponding to a point in a mixed-color region in the plot and query it. You should likely obtain shapes from different classes in the query result.

Note: For the interested ones: There is a proven correlation between the degree of visual separation in a t-SNE projection and how easily the projected items are separable (by a classifier) from each other. See

https://webspace.science.uu.nl/~telea001/uploads/PAPERS/InfVis17/paper.pdf