Interactive-HSNE-plugin:

A small algorithm documentation

Alexander Vieth

June 7, 2022

1 Selection mapping

Adding selection maps between the data points (image viewer) and landmark in the embedding (embedding view).
Done in `HsneHierarchy::getSelectionMapsAtScale()`.

Algorithm 1: Selection mapping

Input : scaleLevel, idMap
Output: mappingLocalToBottom, mappingBottomToLocal

for *embID* ← 0 **to** *Number of points in embedding* **do**
    localIdOnScale ← idMap[embID]
    globalID ← embeddingHierarchy.MapToGlobal(scaleLevel, localIdOnScale)
    influencedIDs ← embeddingHierarchy.MapTopDown(scaleLevel, localIdOnScale)
    `// each embID maps to (potentially) several global IDs`
    mappingLocalToGlobal[embID] ← {globalID, influencedIDs}
    `// each global ID maps to all embIDs that influence it`
    for *globalID* **in** *mappingLocalToGlobal[embID]* **do**
        mappingBottomToLocal[globalID] ← embID
    end
end

`// data structure that hold hierarchy and scale information`
embeddingHierarchy

`// returns global data ID`
embeddingHierarchy.MapToGlobal(scale, localIdOnScale)

`// returns global IDs which are influenced the most by the local IDs`
embeddingHierarchy.MapTopDown(scale, localIdOnScale)

`idMap` holds pairs of (ID on scale, ID in embedding) for all points in the embedding.

Algorithm 2: Computing the idMap

Input : localIDsOnCoarserScale
Output: idMap

```
1 for embPos ← 0 in localIDsOnCoarserScale.size() do
2 |   idMap.insert( { localIDsOnCoarserScale[embPos], embPos } )
3 end
```

2 ID mapping

Adding id map between the data points (image viewer) and landmark in the embedding (embedding view).

2.1 Heuristic

Done in `computeLocalIDsOnCoarserScaleHeuristic()`.
`EmbeddingHierarchy.MapBottomUp()` is the reverse mapping of `EmbeddingHierarchy.MapTopDown()`:

1. MapTopDown(scale, localIDOnScale) returns a list of global IDs for which localIDOnScale has the highest influence, i.e. of all landmarks on the given scale localIDOnScale has the highest influence for each of the returned global IDs.

2. MapBottomUp(scale, globalID) returns the local ID on the given scale which has the highest influence globalID. It might not return any ID if the heuristic used to compute the influence hierarchy `InfluenceHierarchy::initialize()` did not find any landmark for globalID at the given scale.

Algorithm 3: ID mapping heuristic

Input : scaleLevel, imageSelectionIDs
Output: localIDsOnScale

```
1 for imageSelectionID in imageSelectionIDs do
2 |   localIDsOnScale ← embeddingHierarchy.MapBottomUp(scaleLevel, imageSelectionID)
3 end
4 localIDsOnScale.retainUnique()
  // data structure that hold hierarchy and scale information
5 embeddingHierarchy
  // returns local IDs which influence the global IDs the most
6 embeddingHierarchy.MapBottomUp(scaleLevel, imageSelectionID)
```

2.2 Precise

Done in `computeLocalIDsOnCoarserScale()`.

Instead of using the heuristically precomputed influence hierarchy as above, here we go from the bottom (data level/global) scale upwards.

Algorithm 4: ID mapping precise

Input : scaleLevel, imageSelectionIDs, threshold
Output: localIDsOnScale

```
1  localIDsOnScale ← imageSelectionIDs
2  for currentScaleLevel ← 0 to scaleLevel do
3  |  coarserScaleInfluence ← embeddingHierarchy.getInfluencingLandmarksInCoarserScale
   |   (currentScaleLevel, localIDsOnScale)
4  |  localIDsOnScale.clear()
   |  // threshold the influences
5  |  for (coarserID, influence) in coarserScaleInfluence do
6  |  |  if influence > threshold then
7  |  |  |  localIDsOnScale.append(coarserID)
8  |  |  end
9  |  end
10 end
   // data structure that hold hierarchy and scale information
11 embeddingHierarchy
   // Returns a map of local IDs on the coarser scale (currentScaleLevel + 1)
   and their influences on the localIDsOnScale
12 embeddingHierarchy.getInfluencingLandmarksInCoarserScale(currentScaleLevel,
   localIDsOnScale)
```