```
function d=hypergraphDissimilarity(mat1,mat2,type,NameValueArgs)
%computes tensor based distance based on g1, g2 datastructure generated by
%createTensorRepresentation for two given hypergraphs
%type is what distance measure to use
%params are parameters related to specific distance
arguments
    mat1
    mat2
    type = 'Hamming'
    NameValueArgs.Tolerance = 1e-4
    NameValueArgs.MaxIter = 3000
    NameValueArgs.Model = 'LogExp'
    NameValueArgs.Alpha = 10
end

import Computations.hypergraphCentrality;
import DissimilarityMeasures.tensor.*;

d=NaN;
switch type

    case 'Hamming'

        if ~isempty(mat1) && ~isempty(mat2)
            d = DissimilarityMeasures.TensorDis.Hamming(mat1, mat2);
        end

    case 'Spectral-S'

        if ~isempty(mat1) && ~isempty(mat2)
            d = DissimilarityMeasures.TensorDis.SpectralS(mat1, mat2);
        end

    case 'Spectral-H'

        if ~isempty(mat1) && ~isempty(mat2)
            d = DissimilarityMeasures.TensorDis.SpectralH(mat1, mat2);
        end

    case 'Centrality'
        % mat1 and mat2 are incidence matrices
        W1=ones(size(mat1,2),1);
        N1=ones(size(mat1,1),1);
        W2=ones(size(mat2,2),1);
        N2=ones(size(mat2,1),1);

        HG1.IM = mat1;
        HG1.edgeWeights = W1;
        HG1.nodeWeights = N1;

        HG2.IM = mat2;
        HG2.edgeWeights = W2;
```

```
        HG2.nodeWeights = N2;

        tol = NameValueArgs.Tolerance;
        maxIter = NameValueArgs.MaxIter;
        model = NameValueArgs.Model;
        alpha = NameValueArgs.Alpha;

        cenArgs = {'Tolerance', tol, 'MaxIter', maxIter, 'Model',
 model, 'Alpha', alpha};
        [nodeCentrality1, ~] = hypergraphCentrality(HG1, cenArgs{:});
        [nodeCentrality2, ~] = hypergraphCentrality(HG2, cenArgs{:});
        nodeCentrality1=nodeCentrality1/norm(nodeCentrality1);
        nodeCentrality2=nodeCentrality2/norm(nodeCentrality2);
        d=norm(nodeCentrality1-nodeCentrality2)/length(nodeCentrality1);
end
```

Published with MATLAB® R2021b