```
if __name__ == '__main__':

    data_input_file = 'demo/data/textTest/test_1_dropout_0_overtime.pkl'
    output_eq_file = 'demo/data/textTest/test_1_dropout_0.mat'

1.  epsPred = 9e-3 #Prediction error
    epsCorr = 1e-9 #Correction error
    maxSteps = 5e2 #Max number of steps

    thrPrune = .01

2.  lstm = LSTMNN(None)

3.  weights = readFromPickle(data_input_file)
4.  weights, pPrune = pruneW(weights,thrPrune)

5.  lstm.updateWeigths(weights)

    ndim = 2*lstm.dimOfOutput()
    x0 = np.zeros([ndim,1])
6.  lstm.setInitStat(x0)

    print("dim = ",str(ndim))
7.  (t,x,eCode) = lstm.findEquilibria(epsPred,epsCorr,maxSteps)

8.  xeq = lstm.refineTheEquilibria(x)
    xeq = np.array([xeq]).T

    lstm.errorInterpreter(eCode)

    print("||F(xeq)-xeq|| = ",linalg.norm(xeq-lstm.F(xeq),2))
```

Figure 1: Demo code

In 1., we define the parameters of the solver. Recall that the homotopy solver is mainly solving the ODE $\frac{dH(t,x)}{dt} = 0$ using the predictor-corrector scheme. Thus, epsPred defines the error of the Predictor and epsCorr defines the error of the Corrector. maxSteps defines the maximum number of integration steps. Then, in 2., we create an object that represent the LSTM map. In, this object, we define the lstm map and store the weights and biases.
In, 3, we read the weights and biases from the pickle file. readFromPickle() is defined by the user to read weights from an external file.
4., the pruning function is used.
5., we update the weights read from the external file to the lstm object.
6., we define the initial state of the homotopy solver.
7., the solver is called and return the equilibrium.

8., Once we have the equilibrium we can use newton method to refine the equilibirium and reduce the error.