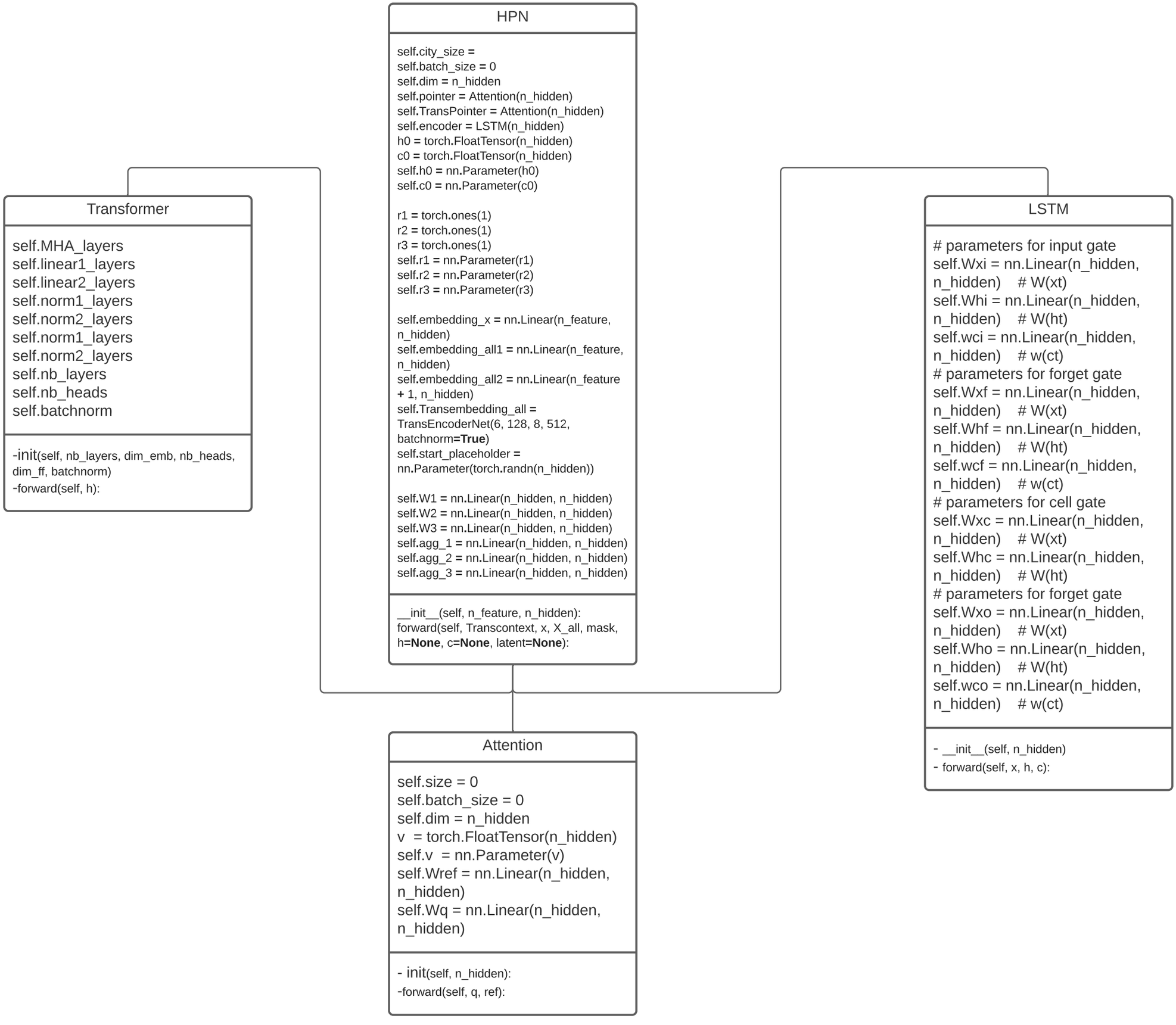
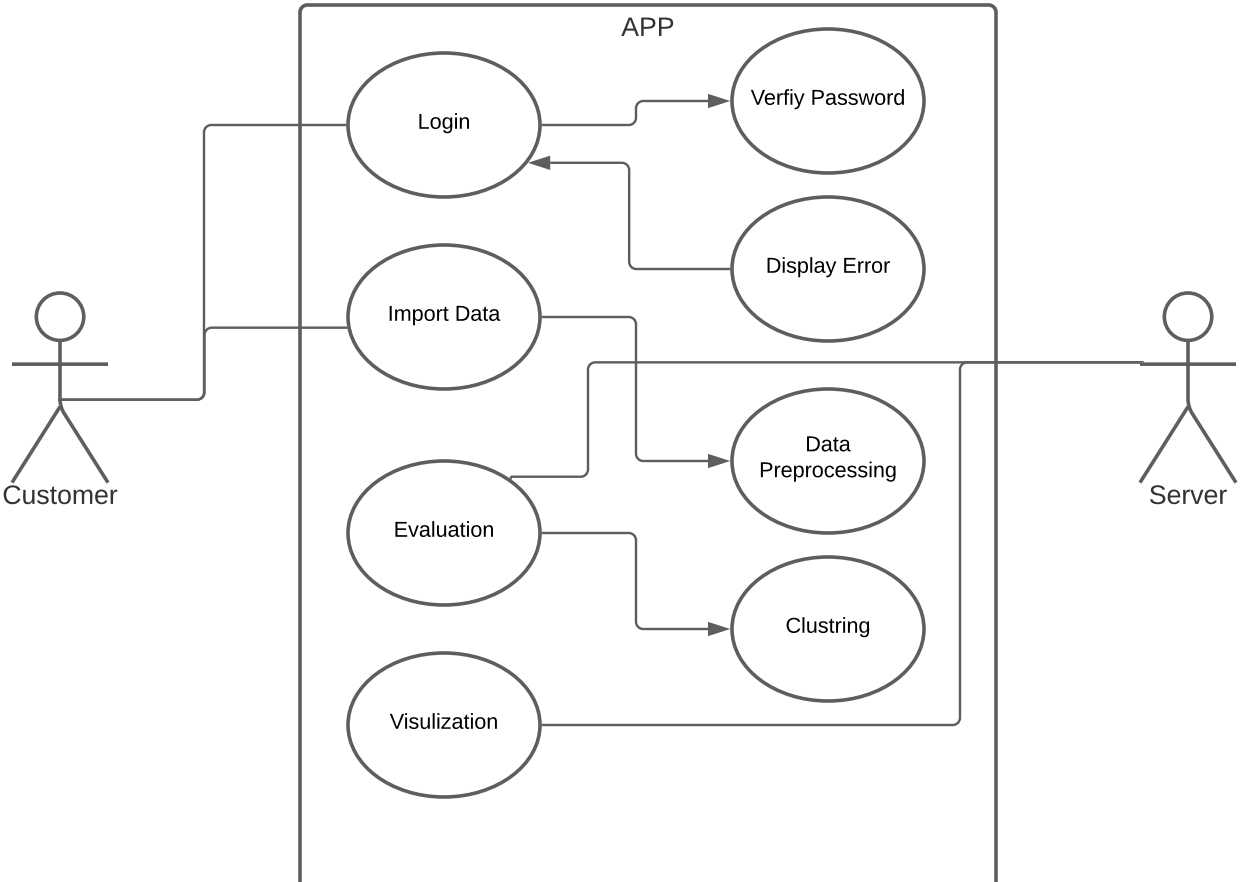

HPN
self.city_size =
self.batch_size = 0
self.dim = n_hidden
self.pointer = Attention(n_hidden)
self.TransPointer = Attention(n_hidden)
self.encoder = LSTM(n_hidden)
h0 = torch.FloatTensor(n_hidden)
c0 = torch.FloatTensor(n_hidden)
self.h0 = nn.Parameter(h0)
self.c0 = nn.Parameter(c0)
r1 = torch.ones(1)
r2 = torch.ones(1)
r3 = torch.ones(1)
self.r1 = nn.Parameter(r1)
self.r2 = nn.Parameter(r2)
self.r3 = nn.Parameter(r3)
self.embedding_x = nn.Linear(n_feature, n_hidden)
self.embedding_all1 = nn.Linear(n_feature, n_hidden)
self.embedding_all2 = nn.Linear(n_feature + 1, n_hidden)
self.Transembedding_all = TransEncoderNet(6, 128, 8, 512, batchnorm=True)
self.start_placeholder = nn.Parameter(torch.randn(n_hidden))
self.W1 = nn.Linear(n_hidden, n_hidden)
self.W2 = nn.Linear(n_hidden, n_hidden)
self.W3 = nn.Linear(n_hidden, n_hidden)
self.agg_1 = nn.Linear(n_hidden, n_hidden)
self.agg_2 = nn.Linear(n_hidden, n_hidden)
self.agg_3 = nn.Linear(n_hidden, n_hidden)
__init__(self, n_feature, n_hidden):
forward(self, Transcontext, x, X_all, mask, h=None, c=None, latent=None):
Transformer
self.MHA_layers
self.linear1_layers
self.linear2_layers
self.norm1_layers
self.norm2_layers
self.norm1_layers
self.norm2_layers
self.nb_layers
self.nb_heads
self.batchnorm
-init(self, nb_layers, dim_emb, nb_heads, dim_ff, batchnorm)
-forward(self, h):
LSTM
# parameters for input gate
self.Wxi = nn.Linear(n_hidden, n_hidden) # W(xt)
self.Whi = nn.Linear(n_hidden, n_hidden) # W(ht)
self.wci = nn.Linear(n_hidden, n_hidden) # w(ct)
# parameters for forget gate
self.Wxf = nn.Linear(n_hidden, n_hidden) # W(xt)
self.Whf = nn.Linear(n_hidden, n_hidden) # W(ht)
self.wcf = nn.Linear(n_hidden, n_hidden) # w(ct)
# parameters for cell gate
self.Wxc = nn.Linear(n_hidden, n_hidden) # W(xt)
self.Whc = nn.Linear(n_hidden, n_hidden) # W(ht)
# parameters for forget gate
self.Wxo = nn.Linear(n_hidden, n_hidden) # W(xt)
self.Who = nn.Linear(n_hidden, n_hidden) # W(ht)
self.wco = nn.Linear(n_hidden, n_hidden) # w(ct)
- __init__(self, n_hidden)
- forward(self, x, h, c):
Attention
self.size = 0
self.batch_size = 0
self.dim = n_hidden
v = torch.FloatTensor(n_hidden)
self.v = nn.Parameter(v)
self.Wref = nn.Linear(n_hidden, n_hidden)
self.Wq = nn.Linear(n_hidden, n_hidden)
- init(self, n_hidden):
-forward(self, q, ref):


APP
Login
Verfiy Password
Display Error
Import Data
Data
Preprocessing
Evaluation
Clustring
Visulization
Customer
Server