```
loss = nn.BCEWithLogitsLoss()
In [35]:
         optimizer = optim.SGD(net.parameters(), 0.01)
         m = nn.Sigmoid()
In [*]:
         for epoch in range(5):
             for i, batch in enumerate(data iter):
                 center, context negative, label = batch
                 v = net[0](center.to(torch.int64))
                 u = net[1](context negative.to(torch.int64))
                 pred = torch.tensordot(v, torch.transpose(u, 1, 2))
                 1 = loss(m(pred), label.to(torch.float32))
                 optimizer.zero grad()
                 1.backward()
                 optimizer.step()
                 if (i + 1) % 50 == 0:
                      print(epoch, i, float(1))
         0 49 0.9402065277099609
         0 99 0.9058647155761719
         0 149 0.8934478163719177
         0 199 0.9536991119384766
         0 249 0.9259445667266846
         0 299 0.9144635796546936
         0 349 0.9567026495933533
         0 399 0.9884507656097412
         0 449 0.9946126341819763
         0 499 0.9190007448196411
         0 549 0.9105140566825867
         0 599 0.9254959225654602
         0 649 1.076527714729309
         0 699 0.9692239165306091
         0 749 1.020203948020935
         0 799 0.9534463882446289
         0 849 1.0168224573135376
         0 899 0.9782829880714417
         0 949 0.965518593788147
In [44]: | def get similar_tokens(query_token, k, embed):
                 W = embed.weight
                 x = W[token to idx[query token]]
                 \#cos = np.dot(W,x)/np.sqrt(np.sum(W*W, axis=1) * np.sum(x * x) +
                 cos = nn.CosineSimilarity(dim=0, eps=1e-9)
                 \cos f = \cos(W, x)
                 topk = torch.topk(cos f, k=k+1, dim=0)
                 for i in topk.indices:
                      print('cosine sim=%.3f: %s' % (cos_f[i],(idx_to_token[i])))
```