Applied Deep Learning
Homework 1

系級：資工所 學號：r07922156 姓名：卓書宇

Q1 : Data processing

a. 三個皆使用 python 套件 nltk 去將資料分開
b. 前兩個皆使用 4個 negative samples 去訓練我的模型，best則用9個 negative samples
c. 三個皆保留 utterances 和 options 的原始長度
d. 與助教使用相同 fasttext -> crawl-300d-2M.vec

Q2 : Describe your RNN w/o attention model

a. 將 utterances 和 options 各自經過一層 LSTM 得到向量u, v，再利用 u 的轉置乘上 W 在乘上 v來計算 u 和 v 的距離，其中 W 用 PyTorch 中的 linear 來實作
b. 在public leaderboard 分數為 9.92 ，在 valid 的 Recall@10 為 0.45
c. BCEWithLogitsLoss in PyTorch 也就是先經過 sigmoid 的 Binary Cross Entropy
d. Optimization -> Adam , LearningRate -> 1e-5 , BatchSize -> 100

Q3 : Describe your RNN w/ attention model

a. step1 : 將 utterances 和 options 各自經過一層 LSTM 得到向量u, v
   step2 : 將每個 batch 的 u 和 v 做內積運算，得到之結果對 u 的維度做 softmax
   step3 : 拿 step2 得到之向量乘上 u，得到 a
   step4 : 將[u, a, u*a, u-a]串再一起，並且放入第二個 LSTM，得到結果再與 u 做內積
b. 在public leaderboard 分數為 9.42 ，在 valid 的 Recall@10 為 0.72
c. BCEWithLogitsLoss in PyTorch 也就是先經過 sigmoid 的 Binary Cross Entropy
d. Optimization -> Adam , LearningRate -> 1e-5 , BatchSize -> 100

Q4 : Describe your best mode

1. Describe
   a. step1 : 將 utterances 和 options 各自經過一層 LSTM 得到向量u, v
      step2 : 將每個 batch 的 u 和 v 做內積運算，得到之結果對 u 的維度做 softmax
      step3 : 拿 step2 得到之向量乘上 u，得到 a
      step4 : 將[u, a, u*a, u-a]串再一起，並且放入第二個 LSTM，得到結果再與 u 做內積
   b. 在public leaderboard 分數為 9.42 ，在 valid 的 Recall@10 為 0.72
   c. BCEWithLogitsLoss in PyTorch 也就是先經過 sigmoid 的 Binary Cross Entropy
   d. Optimization -> Adam , LearningRate -> 1e-5 , BatchSize -> 100
2. Describe the reason you think why your best model is better than your RNN w/ and w/o attention model.
   我的 best model 就是 attention model ， 我覺得他比沒有加上 attention model 好的原因就是因為有加上 attention，並且我將 context 串在了一起，所以總結來說比 Rnn w/o model 還好。

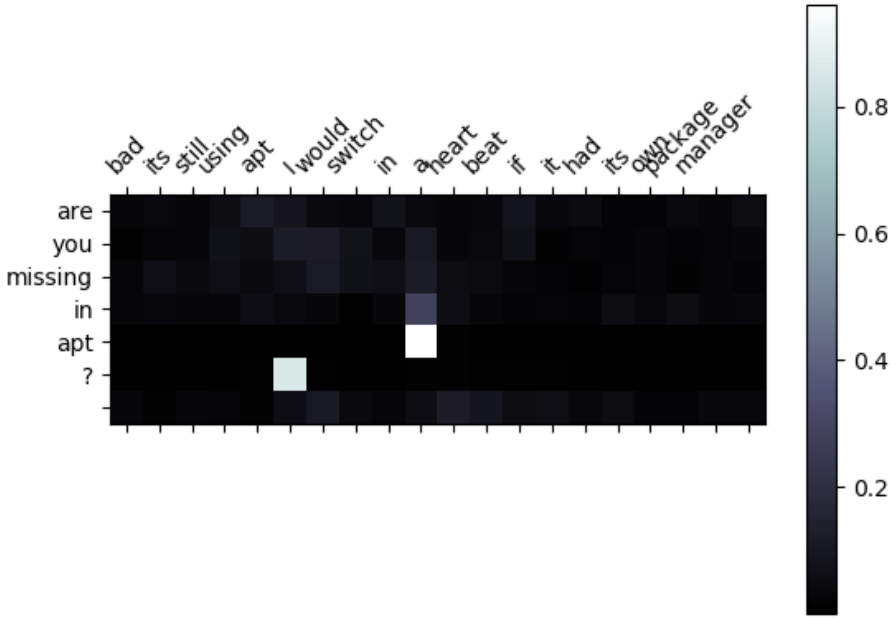
Q5 : Compare GRU and LSTM

以下情形皆是同一時間在同一張GPU上所跑的時間

a. 兩者做出來的 recall@10 極限都差不多在 0.72-0.73
b. 在 batch size = 25 且 negative samples 為 4的情況下，GPU memory使用了 4633MB 以及 5237MB ，分別為 GRU 以及 LSTM
c. GRU 之 training 以及 testing 的時間分別為 36~38分鐘、6~7分鐘，LSTM 之 training 以及 testing 時間分別為 40~42分鐘 以及 6~7分鐘

Q6 : Visualize the attention weights

1. Visualize the attention



2. 我發這個視覺化的圖，有點難理解機器到底學了什麼東西，單純就顏色來講，越白的就是 attention weight 越高的。

Q7 : Compare training with different settings:

- different reasonable loss functions
  使用過 MSELoss 和 BCELoss 使用成效為 BCELoss 較好
- different number of negative samples
  使用過4個和9個 negative samples，效果為9個較好，但速度較慢，資源耗費較多
- different number of utterances in a dialog
  使用過單純最後一句 utterance 和全部的 utterances ，全部的utterances效果較好，但速度較慢，資源耗費較多
- different pre-trained word embeddings
  使用過 fastest 的 crawl-300d-2M.vec 和 google 的 word2vec ，fasttext較為好用，因為他會自己處理 OOV 的字