NLP HW1 NER Report

姓名： 薛竣祐　　學號： 111526009

一. 資料前處理

在前處理部分，考慮到句子來源為Twitter，並在觀測資料後，發現有資料包含許多URL、Hashtag等特殊資料，因此在讀取資料時將此類資料轉換為個別Token，另外也移除掉字詞中的符號。使用到的Token有：

- <at>："@"符號開頭的字詞
- <hashtag>："#"符號開頭的字詞
- <url>："http"或"www."開頭的字詞
- <naw>：只包含符號的字詞

```python
def get_data(path="./datas/twitter/train.txt"):
    with open(path, "r") as f:
        lines = f.readlines()
    datas = [[[]], [[]]] # [ sequences, tags(or origin sequence) ]
    submit = "submit" in path # if submitting, change return tag to return origin word
    for line in lines:
        if line == "\n": # new sequence
            datas[0].append([])
            datas[1].append([])
            continue
        sp = line.split() # word, (tag)
        datas[0][-1].append(encode_word(sp[0]))
        datas[1][-1].append(sp[0] if submit else sp[1])
    return datas[0][:-1], datas[1][:-1]  # last one is empty
    # train, val: [["i", "am", "alvin"]...]    , [["o", "o", "b-person"]...]
    # submit:     [["i", "am", "<hashtag>"]...], [["i", "am", "#handsome"]...]


def encode_word(word):
    if word.startswith("@"): return "<at>" # start with @
    if word.startswith("#"): return "<hashtag>" #start with #
    if word.startswith("http") or word.startswith("www."): return "<url>" # start with http or www.
    return re.sub(r"\W", "", word) or "<naw>"  # remove non-word, if empty then <naw>
```

將資料使用DataLoader來存取，在經過collate_fn後會回傳兩項資料。
第一項：回傳embedding過的sequences (batch_size, seq_len, emb_size)
第二項：若為Training或Validating回傳label的index (batch_size, seq_len)
　　　　若為Testing則回傳未經前處理的原始資料 (batch_size, seq_len)

另外在Training及Validating時，會將兩項製作成PackedSequence物件，而在Testing時則只有第一項。此物件可直接通過GRU Layer，不需要考慮資料長度對齊的問題。

```python
        for seqs, tags_or_words in batched_data:
            seqs_len.append(len(seqs))
            batch_seq_indexs.append(torch.tensor([word_to_i.get(word, unk) for word in seqs]))
            if submit:
                batch_real_words.append(tags_or_words)
            else:
                batch_tag_indexs.append(torch.tensor([tag_to_i[tag] for tag in tags_or_words]))

        padded_seqs = pad_sequence(batch_seq_indexs, batch_first=True, padding_value=word_to_i["<pad>"])
        packed_seqs = pack_padded_sequence(embedding(padded_seqs), seqs_len, batch_first=True, enforce_sorted=False)
        if submit: return packed_seqs, batch_real_words

        padded_tags = pad_sequence(batch_tag_indexs, batch_first=True, padding_value=-1)
        packed_tags = pack_padded_sequence(padded_tags, seqs_len, batch_first=True, enforce_sorted=False)
```

二. 模型架構

1. Pre-train Embedding (freeze)

   使用GloVe作為Pre-train Embedding，其中經過測試發現其效果使用Common Crawl 840B 300d，會比使用Twitter 27B 200d好。並在vocab中加入在前處理時新增的特殊Token，其向量設定為其他向量的平均值。

```python
def create_pretrained_embedding(name, embedding_size):
    glove = GloVe(name=name, dim=embedding_size)
    mean_vector = glove.vectors.mean(dim=0).unsqueeze(dim=0) # mean vector as default vector for new token
    def add_token(token):
        if glove.stoi.get(token, None) is not None:
            print(f"\"{token}\" is already in Glove")
            return
        glove.itos.append(token)
        glove.stoi[token] = glove.itos.index(token)
        glove.vectors = torch.cat((glove.vectors,mean_vector),dim=0)
    add_token("<unk>")
    add_token("<pad>")
    add_token("<naw>")
    add_token("<hashtag>")
    add_token("<at>")
    add_token("<url>")
    return torch.nn.Embedding.from_pretrained(glove.vectors, freeze=True), glove.stoi
```

2. GRU

   在經過各種超參數設定的測試下，評估效率與效果後決定使用以下設定：
   - embedding_size=300
   - hidden_size=256
   - num_layers=2
   - batch_first=True
   - bidirectional=True

   另發現hidden_size與num_layer會嚴重影響執行時間，num_layer提升對模型可能造成反效果，而hidden_size提升對效果無明顯的影響。embedding_layer使用twitter 50d~200d測試，結果顯示embedding_layer越高效果越好。

3. Dropout

   Dropout Layer可以有效的避免模型過度依賴某些特徵，造成Overfitting，因此在最後的Output Layer前加入dropout。而不在GRU使用dropout，是因為GRU不會在最後一層套用dropout。

4. Linear

   Linear最後會輸出21維的結果，而21維對應的是21種的Label，並依照Label在Training Data出現機率由大至小排序後的index。

三. 模型訓練

相關超參數設定如下：

- Learning Rate：0.001
- Epoch：100
- Batch Size：32
- Clip Grad：0.5
- Loss Function：CrossEntropy
- Optimizer：AdamW

使用助教提供的Github程式計算F1成績，最佳模型成績約在36~43之間，訓練時間約為30分鐘。且hidden_size及num_layer與訓練時間呈正相關，例如如果為hidden_size=128且num_layer=1時，訓練時間約為7分鐘。

```python
for i in range(epoch):
    train_loss = []
    dev_loss = []
    # train
    net.train()
    for padded_seqs, tags_target in dataloader:
        tags_predict = net(padded_seqs)
        loss = loss_fn(tags_predict, tags_target)  # auto index class cal
        train_loss.append(loss.item())
        optimizer.zero_grad()
        loss.backward()
        torch.nn.utils.clip_grad_value_(net.parameters(), 0.5)
        optimizer.step()
    # dev
    net.eval()
    with torch.no_grad():
        padded_seqs, tags_target = next(iter(dev_dataloader)) # full batch
        tags_predict = net(padded_seqs)
        loss = loss_fn(tags_predict, tags_target)
        dev_loss.append(loss.item())
        _, _, f1, non_o_accuracy, accuracy = evaluate(
            [i_to_tag[idx] for idx in tags_target.tolist()],
            [i_to_tag[idx] for idx in tags_predict.argmax(dim=1).tolist()],
            True
        )
```

四. 實驗成果

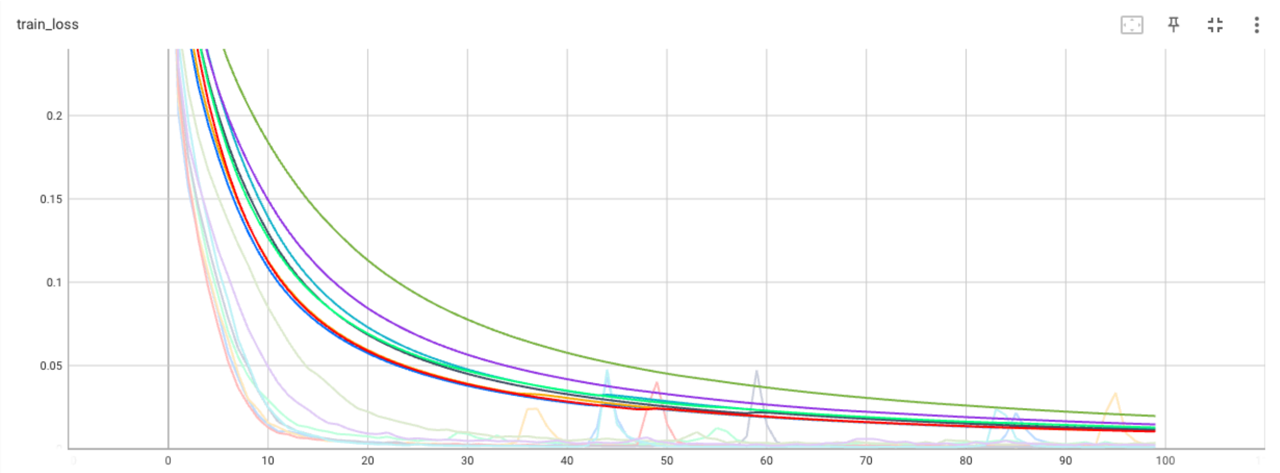
在此節將以圖示比較相同模型於不同超參數的部分執行結果，使用Tensorboard做圖，淺色線為真實資料，而深色線為平滑過後的資料，縱軸皆為Epoch(0~99)。

Model	num layers	dropout (inGRU, afterGRU)	hidden size	grad clip	token vector
GRU_layer2	2	X	128	X	random
GRU_layer3	3	X	128	X	random
GRU_layer2_dp0.5_h256	2	after 0.5	256	X	random
GRU_layer2_dp0.5_h256_clip0.5	2	after 0.5	256	0.5	random
GRU_layer1_dp0.5_h256_clip0.5	1	after 0.5	256	0.5	random
GRU_layer1_dp0.5_h128_clip0.5	1	after 0.5	128	0.5	random
GRU_layer2_dp0.5inGRU_h256_clip0.5_meanV	2	in 0.5	256	0.5	mean
submit	**2**	**after 0.5**	**256**	**0.5**	**mean**

此表顯示各model之間使用的超參數。num layers表示GRU layer使用的層數；dropout表示於GRU中或後使用的dropout比例；hidden size表示GRU中隱藏層大小；grad clip表示在gradient時clipping的正負範圍；token vector表示特殊vocab token使用的vector計算方式。若為X則為無使用。

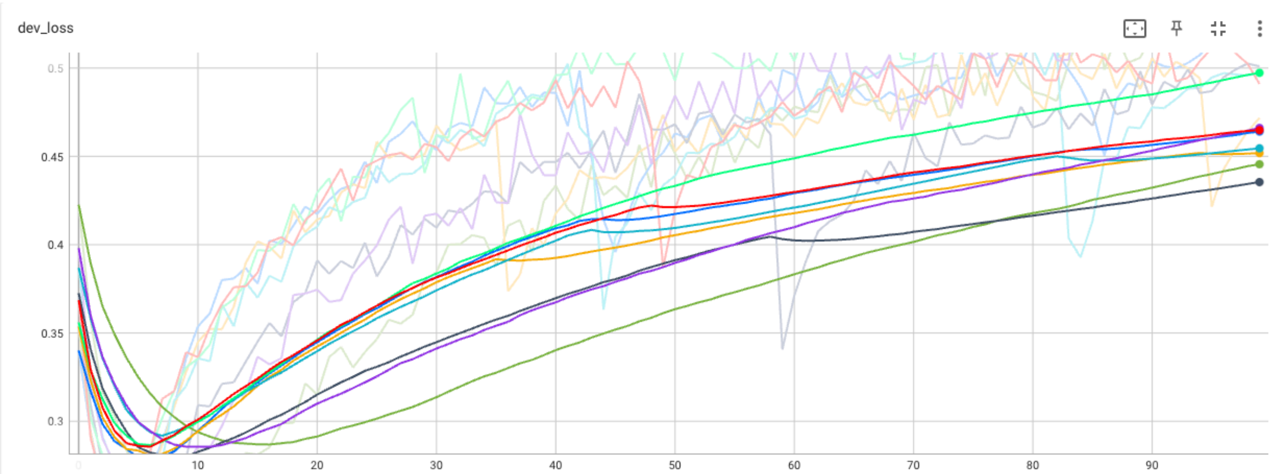
---

1. Training Loss



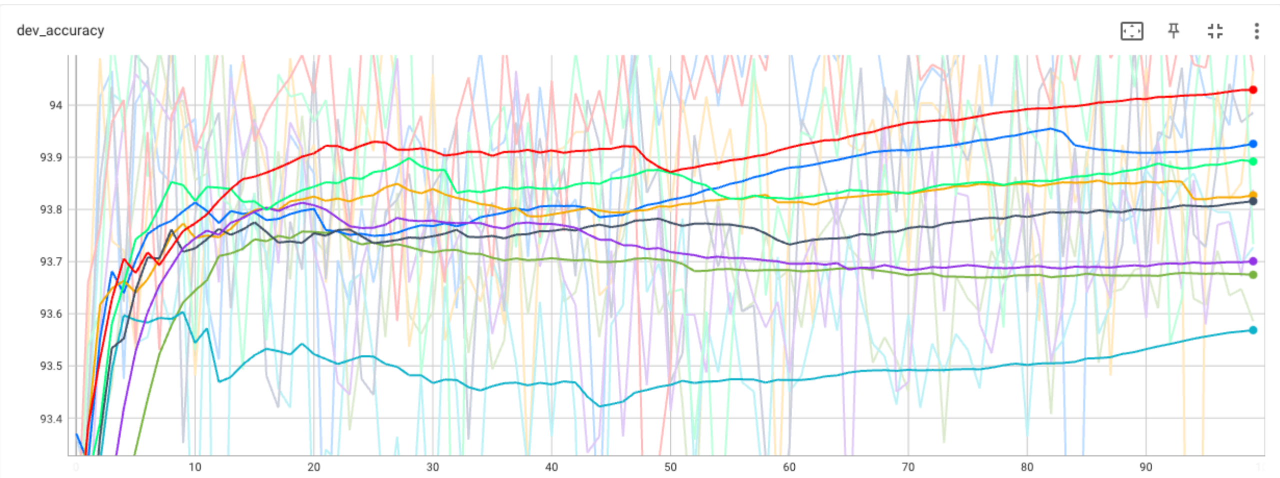
Run	Smoothed	Value	Step	Time	Relative
runs/GRU_layer1_dp0.5_h128_clip0.5.local	0.01971	0.003602	99	11/3/22, 2:13 PM	6.587 min
runs/GRU_layer1_dp0.5_h256_clip0.5.local	0.01464	0.002179	99	11/1/22, 11:33 PM	14.66 min
runs/GRU_layer2.local	0.01134	0.001125	99	11/1/22, 4:44 PM	12.79 min
runs/GRU_layer2_dp0.5_h256.local	0.01081	0.001656	99	11/1/22, 10:00 PM	34.79 min
runs/GRU_layer2_dp0.5_h256_clip0.5.local	0.0113	0.002253	99	11/1/22, 11:08 PM	34.66 min
runs/GRU_layer2_dp0.5inGRU_h256_clip0.5_meanV.local	0.0128	0.002293	99	11/4/22, 4:20 PM	34.53 min
runs/GRU_layer3.local	0.0125	0.001198	99	11/1/22, 5:07 PM	19.45 min
runs/submit.local	0.0104	0.002077	99	11/4/22, 6:23 PM	34.87 min

2. Validating Loss



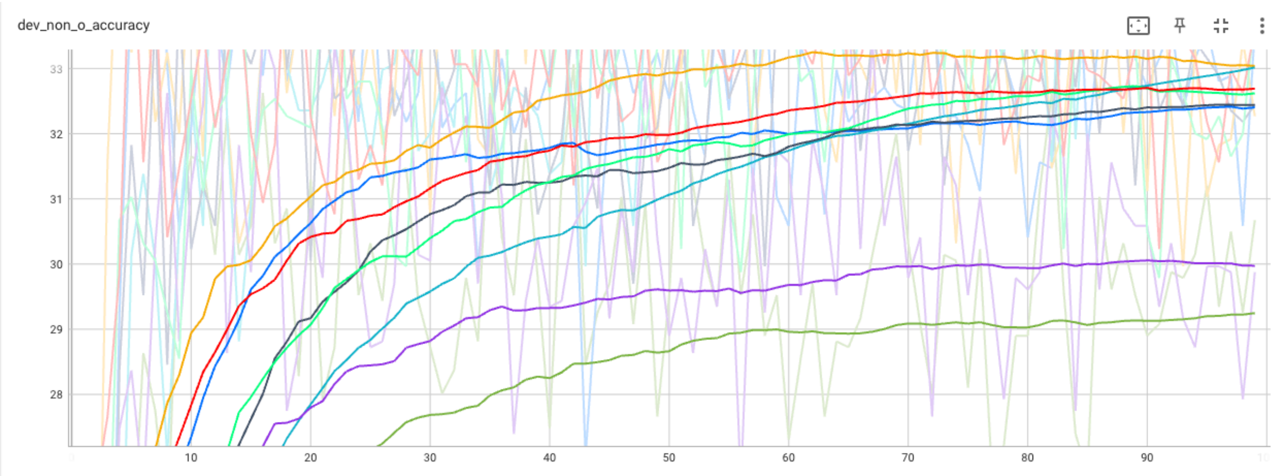
Run	Smoothed	Value	Step	Time	Relative
runs/GRU_layer1_dp0.5_h128_clip0.5.local	0.4456	0.5131	99	11/3/22, 2:13 PM	6.587 min
runs/GRU_layer1_dp0.5_h256_clip0.5.local	0.4661	0.5576	99	11/1/22, 11:33 PM	14.66 min
runs/GRU_layer2.local	0.4354	0.501	99	11/1/22, 4:44 PM	12.79 min
runs/GRU_layer2_dp0.5_h256.local	0.4641	0.5149	99	11/1/22, 10:00 PM	34.79 min
runs/GRU_layer2_dp0.5_h256_clip0.5.local	0.4519	0.4718	99	11/1/22, 11:08 PM	34.66 min
runs/GRU_layer2_dp0.5inGRU_h256_clip0.5_meanV.local	0.4973	0.569	99	11/4/22, 4:20 PM	34.53 min
runs/GRU_layer3.local	0.4546	0.4975	99	11/1/22, 5:07 PM	19.45 min
runs/submit.local	0.4648	0.491	99	11/4/22, 6:23 PM	34.87 min

3. Validating Total Accuracy



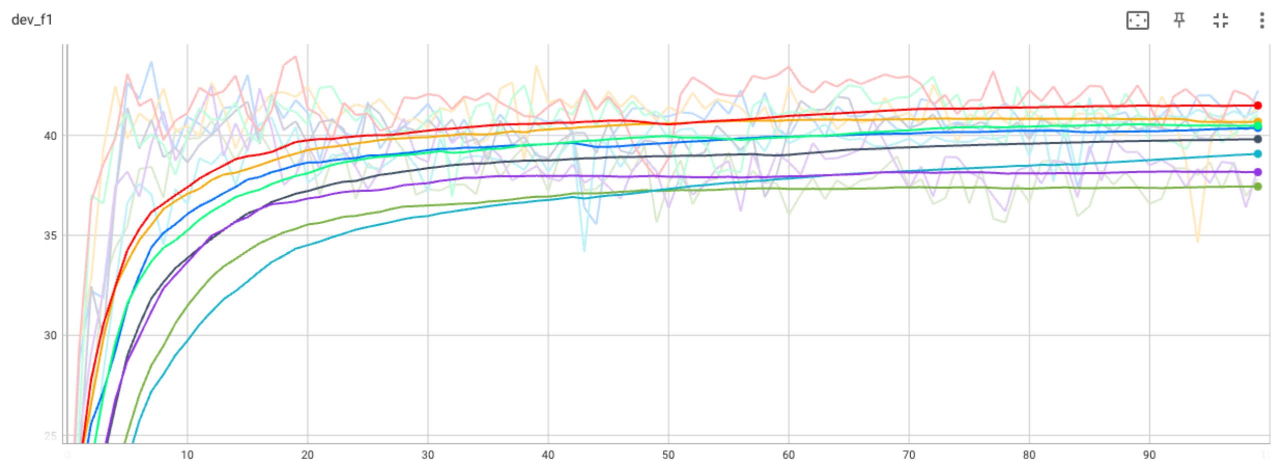
Run	Smoothed	Value	Step	Time	Relative
runs/GRU_layer1_dp0.5_h128_clip0.5.local	93.67	93.59	99	11/3/22, 2:13 PM	6.587 min
runs/GRU_layer1_dp0.5_h256_clip0.5.local	93.7	93.84	99	11/1/22, 11:33 PM	14.66 min
runs/GRU_layer2.local	93.82	93.99	99	11/1/22, 4:44 PM	12.79 min
runs/GRU_layer2_dp0.5_h256.local	93.93	94.17	99	11/1/22, 10:00 PM	34.79 min
runs/GRU_layer2_dp0.5_h256_clip0.5.local	93.83	94.07	99	11/1/22, 11:08 PM	34.66 min
runs/GRU_layer2_dp0.5inGRU_h256_clip0.5_meanV.local	93.89	93.73	99	11/4/22, 4:20 PM	34.53 min
runs/GRU_layer3.local	93.57	93.73	99	11/1/22, 5:07 PM	19.45 min
runs/submit.local	94.03	94.07	99	11/4/22, 6:23 PM	34.87 min

4. Validating non-o Accuracy



Run	Smoothed	Value	Step	Time	Relative
runs/GRU_layer1_dp0.5_h128_clip0.5.local	29.25	30.67	99	11/3/22, 2:13 PM	6.587 min
runs/GRU_layer1_dp0.5_h256_clip0.5.local	29.97	29.88	99	11/1/22, 11:33 PM	14.66 min
runs/GRU_layer2.local	32.44	32.45	99	11/1/22, 4:44 PM	12.79 min
runs/GRU_layer2_dp0.5_h256.local	32.4	33.6	99	11/1/22, 10:00 PM	34.79 min
runs/GRU_layer2_dp0.5_h256_clip0.5.local	33.04	32.27	99	11/1/22, 11:08 PM	34.66 min
runs/GRU_layer2_dp0.5inGRU_h256_clip0.5_meanV.local	32.62	33.87	99	11/4/22, 4:20 PM	34.53 min
runs/GRU_layer3.local	33.02	35.46	99	11/1/22, 5:07 PM	19.45 min
runs/submit.local	32.69	33.6	99	11/4/22, 6:23 PM	34.87 min

5. Validating F1 Score



Run	Smoothed	Value	Step	Time	Relative
runs/GRU_layer1_dp0.5_h128_clip0.5.local	37.44	37.82	99	11/3/22, 2:13 PM	6.587 min
runs/GRU_layer1_dp0.5_h256_clip0.5.local	38.16	38.35	99	11/1/22, 11:33 PM	14.66 min
runs/GRU_layer2.local	39.81	40.54	99	11/1/22, 4:44 PM	12.79 min
runs/GRU_layer2_dp0.5_h256.local	40.38	42.24	99	11/1/22, 10:00 PM	34.79 min
runs/GRU_layer2_dp0.5_h256_clip0.5.local	40.69	41.4	99	11/1/22, 11:08 PM	34.66 min
runs/GRU_layer2_dp0.5inGRU_h256_clip0.5_meanV.local	40.49	41.04	99	11/4/22, 4:20 PM	34.53 min
runs/GRU_layer3.local	39.08	41.21	99	11/1/22, 5:07 PM	19.45 min
runs/submit.local	41.5	41.54	99	11/4/22, 6:23 PM	34.87 min