Homework_1 Report

- **Model description (2%)**
  - *RNN (1%)*

```
model.add(Masking(
            mask_value=0.,
            input_shape=(time_steps, feature_size)))
model.add(BatchNormalization())
model.add(Bidirectional(LSTM(
            units = 256,
            return_sequences = True,
            activation='relu')))
model.add(Bidirectional(LSTM(
            units = 256,
            return_sequences = True,
            activation='relu')))
model.add(Bidirectional(LSTM(
            units = 256,
            return_sequences = True,
            activation='relu')))
model.add(Dropout(0.2))
model.add(BatchNormalization())

model.add(TimeDistributed(Dense(
            units = 48,
            activation = 'softmax')))
```

因為前面 data 有做 padding 所以後來要 mask
第一層就先加了 masking
之後接了三層雙向 RNN
接著有一層 dropout 是為了減低 overfitting 的機率
最後再一層 fully connected
但這層因為市街在 RNN 後面所以有加上 Timedistribute
然後直接 output

  - *RNN+CNN (1%)*

```
model.add(Conv1D(
            filters = filter_num,
            kernel_size = 100,
            input_shape=(time_steps, feature_size),
            padding = "same"))
model.add(Masking(mask_value=0.))
model.add(BatchNormalization())
model.add(TimeDistributed(Dense(units = 512)))
model.add(Bidirectional(LSTM(
            units = 512,
            return_sequences = True,
            activation='relu')))
model.add(Bidirectional(LSTM(
            units = 512,
            return_sequences = True,
            activation='relu')))
model.add(Dropout(0.1))
model.add(BatchNormalization())
model.add(TimeDistributed(Dense(
            units = 48,
            activation = 'softmax')))
```

跟 RNN 其實差不多
但前面多加了一層 convolution 層

- **How to improve your performance (1%)**
  - *Write down the method that makes you outstanding*
    我最後來不及在 Kaggle 的 Dead line 前 train 出結果
    在 kaggle 上跟 baseline 只差一點點 (分數大概 17.0 左右)
    Kaggle 上 17.0 的那次 keras 給出的 accuracy 大概在 0.85 左右(只 train 5 epoch)
    然後後來一模一樣的方法 train 到 20 個 epoch 有到 0.95 左右的 accuracy
    但因為來不及在 daedline 前上傳所以沒辦法拿來過 baseline….QQ
  - *Describe the model or technique (0.5%)*
    剛開始讀進 data 之後先把他轉成三維的
    因為要符合 RNN 吃進去的 shape
    所以還需要對每個 sentence 做 padding
    把每句的 time frame 數量都補到一樣長
    但這樣後面額外補的 time frame 會造成誤差
    所以之後要再使用 masking 去把他擋掉
    之後就是建構完 NN 之後丟進去 train
    再建構 NN 這邊有試過很多種方法
    從 simpleRNN , LSTM , Bidirectional 一直到 convolution
    最後我是選擇用 Bidirectional LSTM
  - *Why do you use it (0.5%)*
    因為經過測試觀察 Bidirectional LSTM 比較準確
    理論上因為一個句子不只是有從前往後的傳遞關西
    也是有從後往前的影響
    所以用 Bidirectional 感覺也還蠻合理的
- **Experimental results and settings (1%)**
  - *Compare and analyze the results between RNN and CNN (0.5%)*
    我的 best 是用 RNN 做出來的
    感覺這個 RNN 如果再 train 久一點應該可以更好
    但時間來不及所以沒 train 完
    再做 CNN+RNN 的時候
    不知道為甚麼 loss 很快就收斂到很小了
    但是精準度卻不太高
    我有試著調整 CNN 跟 RNN 的層數
    但好像效果都不怎麼好

- *Compare and analyze the results with other models (0.5%)*
  一開始有試著用 simpleRNN 跟 LSTM 去試看看
  結果 LSTM 有比較好一點
  最後發現 LSTM 加上 Bidirectional 會更好
  所以就決定用雙向 RNN
  但其實單只用這樣結果還是有點差
  後來有詢問助教
  發現原來是接再 RNN 後面的 Dense 層還要再加上 Timedistribute
  才可以保留 RNN 層的效果
  所以最後採用了這個 model
  (other models can be variant of basic RNN, like LSTM, or some novel ideas you use)