Report

R06922074　吳柏威

Model description

我的model是使用了S2VT架構，利用tensorflow去實現，使用了兩個GRU unit。在encode階段，首先input先利用dense先將4096維降到了512，接著與padding concat，padding的部分是為了在decode階段輸入attention使用。接著輸出後在concat一次padding，是為了在decode階段輸入經過embedding lookup之後的label，最後在encode輸出的將會作為attention layer的輸入。

到了decoder階段，input將會是attention和padding的結合。接著第一層的輸出會再和embedding lookup label 做結合再傳入第二層，第二層輸出之後最後再做一個dense去預測最後的句子。

Attention mechanism

我的attention layer是參考該網頁實作而成
https://talbaumel.github.io/attention/
而選定的encode input是選擇了在encode階段最後的output，decode input則是選在decode階段第二層的state，首先將encode在每一次處理timestep的output給記錄下來，接著利用一個w1矩陣去和每一個output相乘，以及在decode階段我們會將每一次處理1個timestep時第二層的GPU state抓出來與w2矩陣相乘後分別加至剛剛的矩陣中，此時的shape為(batch, 80, 512)。接著經過tanh函式之後再乘上一個v矩陣，此時shape為(batch, 80, 1)，然後再進行softmax，稱之為attention weight，代表的意義是每一個timestep上的encode狀態對之後decode每一次處理時的重要程度，數字越接近1代表越重要。最後attention weight在與原本的encode input相乘並相加，最後在代入decode第一層的輸入當中。

加入attention之後，loss的下降曲線變得較穩定，而且epoch多次之後獲得的blue分數也較高，或是句子可以選到類似的字。

How do you improve your performance

首先把句子裡面每一個字出現的次數統計起來，接著將句子裡面所有只有出現一次的句子

使用schedule sampling，在decode第二層輸入label時，將會有一定的機率會輸入上一次輸出的字元

在predict出來後，將句子重複的字元縮短，例如：a a a a 將會壓縮成 a

Experiment settings and results

schedule sampling一開始機率設為1，在每一次epoch結束之後將會乘上0.999，如果decay速度太快的話，會發現epoch後來字數反而會過短。

unit使用512效果稍微好一點

使用pretrain好的word2vec model可以在一開始就獲得較小的loss，以及更快地收斂速度，但是效果沒有比較好。

Attention input使用encode的第二層output以及decode的第二層state效果最好。