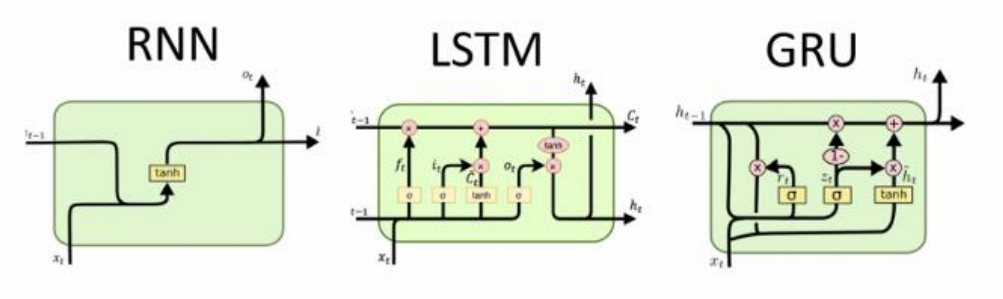
學號：R11921A16　系級：電機所碩一　姓名：何秉學

1. (1.5%) 請說明 RNN、GRU、LSTM 等模型之間的異同。另外，如果你 Kaggle 上最佳的預測結果並不是使用上述三種模型產生的話，請額外說明你使用的 model 為何，以及簡介其背後的原理/機制。

   Ans:

   - RNN Model VS GRU Model VS LSTM Model



     - The detailed formula of RNN is $h_t = \tanh(U \cdot X_t + W \cdot h_{t-1})$ and it usually has much more parameter to compute text data. Generally speaking, RNN has some problems about gradient vanish and gradient explode and it'll cause the long term memory lost (because of gradient vanish).
     - LSTM's model can handle these problems by two techniques – using gate to handle gradient vanish and control long term memory.
     - Meanwhile, there is a simplified version made from LSTM named GRU, which discard h_t. It thought that c_t contained all message from h_t, so in order to cut off the number of parameter, it can be discarded.

   - Reference
     RNN vs LSTM vs GRU -- 该选哪个？

2. (0.5%) 請解釋為何 RNN 模型會發生 gradient vanishing 以及 gradient exploding，以及這兩個現象對 training 可能會有什麼不良影響。

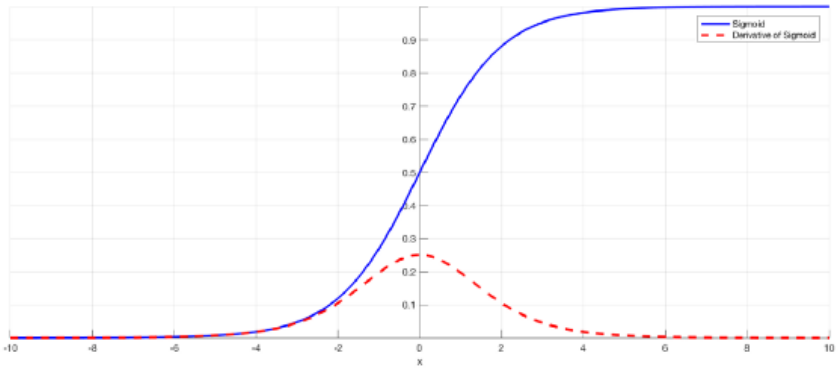
   Ans:

   - In RNN model, it's a simple activation function that given previous h_t and data x. So, for instance as below, we have 4 layer of neural network that each layer has one node and each layer activation function is sigmoid, that is $y_i = \sigma(x_i w_i + b_i)$ where $z_i = x_i w_i + b_i$. So, according to back propagation, we can compute the gradient like this: $\frac{\partial C}{\partial b_i} = \frac{\partial C}{\partial y_4}\frac{\partial y_4}{\partial z_4}\frac{\partial z_4}{\partial x_4}\frac{\partial x_4}{\partial z_3}\frac{\partial z_3}{\partial x_3}\frac{\partial x_3}{\partial z_2}\frac{\partial z_2}{\partial x_2}\frac{\partial x_2}{\partial z_1}\frac{\partial z_1}{\partial b_i}$.

   - And, derivative of sigmoid function is as below $S'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = S(x)(1 - S(x))$. The maximum of the derivative result is 0.25 and we'll set the initial value of our weight as a random value less than one, so $\frac{\partial y_i}{\partial z_i}\frac{\partial z_i}{\partial x_i} = \sigma'(z_i)w_i < 0.25$ and

it'll getting smaller as your layer getting deeper. And this is gradient vanish problem that we want to solve. For our model, it'll cause that our parameter can not be updated.

- Then, if we set the initial value bigger, we'll get $\frac{\partial y_i}{\partial z_i}\frac{\partial z_i}{\partial x_i} = \sigma'(z_i)w_i > 1$. That is, if our network get deeper, the back propagation result will times together, and all these value which are all bigger than one will obtain a huge value. This is gradient explode problem that we want to solve. For our model, it'll get NaN easily that means we can not converge our model.



- Reference
  RNN 的梯度消失问题

3. (1%) 相較於 Sample Code 來說，你做了哪些修改或嘗試(如模型架構、資料前處理、後處理等)？請描述你做的嘗試以及其理由，如果你認為你的做法帶來的進步與第一題的回答有關的話也請詳述之。(請注意若你的解釋太過不合理，則不論你在 leader board 上分數多高，這題都無法拿到滿分)
   - First, I change my model from simple RNN to LSTM and it'll get a higher accuracy and converge the model more easily.
   - Then, I implement the parsing text method for text data preprocessing by filter @ symbol, URL strings, hashtag strings, some HTML strings such as "<" or ">" and some sequence strings such as "…" or "!!!". But the effect is limited and this is as my expectation.
   - In word to vector embedding model, I set configuration shown as below. Iter is just like epochs and min_count is threshold value that we'll ignore all words with total frequency lower than this. I set it to filter some words that are not popular such as "pneumonoultramicroscopicsilicovolcanoconiosis" and it'll be getting better than I expected.
   - Then, I used these techniques above to train backbone and header model and based on the result to train again and again.

window	min_count	workers	iter	sg	w2v_dim
5	5	12	10	1(skip-gram)	250

4. (0.5%) 請簡述你 leader board 上表現最好的實驗結果中使用的 embedding 為何？如何產生？

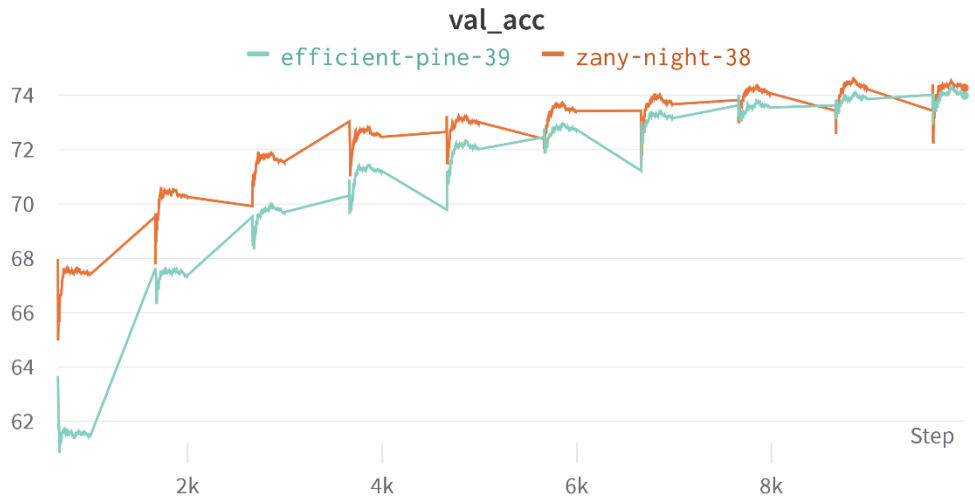
   Ans:

   The embedding I used in my experience is CBOW. The left side figure below is trained LSTM model with skip-gram and right side below is trained by CBOW and they are all set with the same configuration except the w2v method. Obviously, skip-gram method has a lower initial accuracy and can not train the model better than CBOW. So, that's why I choose CBOW as my w2v method.

```
[Training in epoch 1] loss:0.683 acc:56.472
[Validation in epoch 1] loss:0.664 acc:61.452
[Training in epoch 2] loss:0.652 acc:64.688
[Validation in epoch 2] loss:0.636 acc:67.372
[Training in epoch 3] loss:0.624 acc:68.698
[Validation in epoch 3] loss:0.609 acc:69.703
[Training in epoch 4] loss:0.601 acc:70.626
[Validation in epoch 4] loss:0.589 acc:71.218
[Training in epoch 5] loss:0.585 acc:71.778
[Validation in epoch 5] loss:0.576 acc:72.099
[Training in epoch 6] loss:0.574 acc:72.376
[Validation in epoch 6] loss:0.568 acc:72.754
[Training in epoch 7] loss:0.566 acc:73.061
[Validation in epoch 7] loss:0.562 acc:73.178
[Training in epoch 8] loss:0.560 acc:73.448
[Validation in epoch 8] loss:0.557 acc:73.559
[Training in epoch 9] loss:0.556 acc:73.873
[Validation in epoch 9] loss:0.552 acc:73.866
[Training in epoch 10] loss:0.552 acc:74.072
[Validation in epoch 10] loss:0.550 acc:73.995
```

```
[Training in epoch 1] loss:0.652 acc:62.140
[Validation in epoch 1] loss:0.617 acc:67.500
[Training in epoch 2] loss:0.603 acc:68.816
[Validation in epoch 2] loss:0.589 acc:70.262
[Training in epoch 3] loss:0.582 acc:70.905
[Validation in epoch 3] loss:0.573 acc:71.574
[Training in epoch 4] loss:0.569 acc:72.043
[Validation in epoch 4] loss:0.563 acc:72.471
[Training in epoch 5] loss:0.560 acc:72.870
[Validation in epoch 5] loss:0.556 acc:73.014
[Training in epoch 6] loss:0.553 acc:73.399
[Validation in epoch 6] loss:0.551 acc:73.428
[Training in epoch 7] loss:0.548 acc:73.927
[Validation in epoch 7] loss:0.547 acc:73.697
[Training in epoch 8] loss:0.544 acc:74.191
[Validation in epoch 8] loss:0.544 acc:74.077
[Training in epoch 9] loss:0.541 acc:74.443
[Validation in epoch 9] loss:0.541 acc:74.280
[Training in epoch 10] loss:0.539 acc:74.504
[Validation in epoch 10] loss:0.540 acc:74.280
```

   The orange line is CBOW and green one is skip-gram.



5. Play with your models!

   (1) (0.5%)在本題中，s1、s2 互為彼此的 valid permutation，若且唯若 s1、s2 兩句子的單字種類、數量相同、排列順序不同且各自皆為有意義並且合乎文法的句子。例如，A student is a banana 是 A banana is a student 的 valid permutation。請找出一組互為彼此的 valid permutation 且使你的 model 產生

相反的 prediction 的 s1、s2。(s1、s2 須具備合乎邏輯且有實際生活意義的語意)

Ans:

In my example, s1 is “Though the price is expensive, the food is excellent.” and s2 is “Though the food is excellent, the price is expensive.” Obviously, s1 is positive and s2 is negative about a restaurant.

The result is shown as below, that the model generate different prediction though s1 is s2’s valid permutation.

Notice that, the code is revised from the program I submitted, so there is no extra file for this problem.

id	label
1	1
2	0

(2) (1%，bonus) 請從網路上(如 FB、IG、Twitter) 找出一則能夠讓 model 預測錯誤的「反串」酸留言，並將截圖附於 report 上 (即找到一則真實世界存在的留言，使得人類知道這留言應是 negative，但 model outputs positive)

Ans:

In my example, the message from concern troll is shown as below and so as the prediction result. Model think that this message is positive but actually is negative from trolls or haters. So as the previous question that this problem has no extra code version but using the submission code instead.

Introduction to this troll message: In this time world football cup, the dominant hold country, Qatar, is not friendly to LGBTQ+ group and not allow the player to wear the armband. So, some people on web teasing Qatar that they don’t care about human rights however Qatar support that human rights are non-negotiable. You can find that these two things are contraction and ridiculous as well.

3	1

注：請將這兩小題的程式碼也附在繳交的 code 上，並截圖附上來

6. (4%) Math problem:
   https://hackmd.io/@lH2AB7kCSAS3NPw2FffsGg/H1ucYOpNo