學號：T05902136 系級： 資工一 姓名：Jenny Zhang(張臻凝)

1. (1%)請問softmax適不適合作為本次作業的output layer? 寫出你最後選擇的output layer並說明理由。

I don't think softmax is suitable for this assignment's output layer. For this assignment, prediction is multilabel, which means many binary labels instead of just picking one. In softmax, probabilities of all labels are added to 1(100%) and it usually picks the label with highest probability. For example, if label A's probability increases, label B's probability might decrease. It is usually good for multiclass prediction. Since we might pick more than one label, softmax is not the best final activation layer. Therefore, the output layer that I have chosen is sigmoid. In my kaggle result, my assumption is also proven to be true.

output layer	f1 score
sigmoid	0.34888
softmax	0.49778

2. (1%)請設計實驗驗證上述推論。

My code is based off the TA's sample with some changes. The sequential model layers consist of:

Embedding(embedding dimension = 100)

|

GRU(128, activation = tanh, dropout = 0.2)

|

Dense(256, activation = 'relu')

|

Dropout(0.15)

|

Dense(38, activation = 'sigmoid')

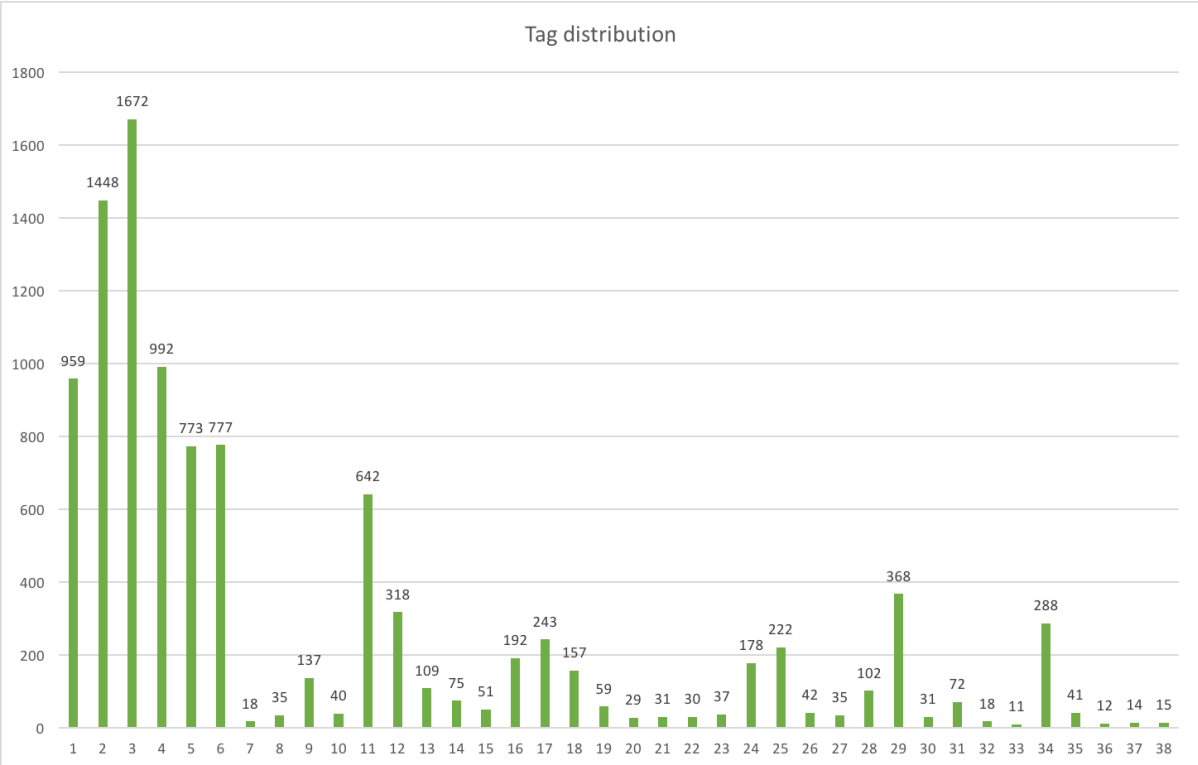
thresh = 0.4, batch_size = 128, epoch = 600, validation split ratio = 0.1

optimizer = Adam(learning rate = 1e-3, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-8)

Early stopping is added with a patience of 15 epochs and checkpoint is used to

save the best model. The reason that I have less layers than the sample code is that it gives a better f1 score. The f1 score improvement is around 0.05, which is quite significant. In addition to the above structure, I preprocess the data by removing stop words. Stop words are commonly used words that do not have a major impact on the text meaning and usually ignored. For example, “are”, “is” and “or” are stop words. In addition, when I save my results, I check to see if there are any labels above the threshold. If there isn’t any, I lower the thresh value for that data row.

3. (1%)請試著分析tags的分布情況(數量)。



1=SCIENCE-FICTION	2=SPECULATIVE-FICTION	3=FICTION	4=NOVEL	5=FANTASY	6=CHILDREN'S-LITERATURE	7=HUMOUR	8=SATIRE
9=HISTORICAL-FICTION	10=HISTORY	11=MYSTERY	12=SUSPENSE	13=ADVENTURE-NOVEL	14=SPY-FICTION	15=AUTOBIOGRAPHY	16=HORROR
17=THRILLER	18=ROMANCE-NOVEL	19=COMEDY	20=NOVELLA	21=WAR-NOVEL	22=DYSTOPIA	23=COMIC-NOVEL	24=DETECTIVE-FICTION
25=HISTORICAL-NOVEL	26=BIOGRAPHY	27=MEMOIR	28=NON-FICTION	29=CRIME-FICTION	30=AUTOBIOGRAPHICAL-NOVEL	31=ALTERNATE-HISTORY	32=TECHNO-THRILLER
33=UTOPIAN-AND-DYSTOPIAN-FICTION	34=YOUNG-ADULT-LITERATURE	35=SHORT-STORY	36=IC-FICTION	37=APOCALYPTIC-AND-POST-APOCALYPTIC-FICTION	38=HIGH-FANTASY		

From the above distribution, FICTION is the most common and UTOPIAN-AND-DYSTOPIAN-FICTION is the rarest. This means that if a testing data is FICTION, the model will probably predict it correctly. The labels with less occurrence will not be as accurate. This is reflected in my model's prediction, where FICTION label appears the most.

4. (1%)本次作業中使用何種方式得到word embedding?請簡單描述做法。

I chose Glove(The wikipedia 2014 + Gigaword 5) for my word embedding. I have tried three dimensions, 100, 200, and 300 and 100 seems to give the best result. To get an embedding dictionary, each word in the file is saved as a key in the dictionary and the value is its coefficient. The words in my data are then converted to those coefficients.

5. (1%)試比較bag of word和RNN何者在本次作業中效果較好。

classification	f1 score
bag of word	0.34781
RNN	0.50620

For my bag of word model, I first converted the data to bag of words by removing punctuations, stopwords and upper cases. The sentences are split into words and made into feature vectors. The features are then feed into a decision tree classifier. From the above results, we can see that RNN is better. I think it is because RNN enables discriminative training, which prevents over complexity. Furthermore, RNN is better at catching long term dependency between words.