Machine Learning Project 5

Multi-class & multi-label article classification

- b03902089 資工三 林良翰

Questions

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.

- Softmax不適合這次作業的輸出層(output layer), 因為輸出的結果能夠同時有多種不同的類別(class), 不會限定只有一種. Softmax出來的結果是找出所有類別當中機率最大的作為唯一的結果, 且所有的類別機率加起來的總和是100%. 因此我這次使用的是Sigmoid作為輸出層, 對所有的類別分別找出機率, 再用一個門檻(threshold)篩選出可能性比較高的類別作為結果.

2. 請設計實驗驗證上述推論.

- 使用同樣的訓練模型, 差別只在於最後一層的觸發方式(activation)

Layer	Neuron	activation
Embedding	-	-
GRU	128	tanh
Dropout 0.1	-	-
Dense	256	elu
Dropout 0.2	-	-
Dense	256	elu
Dropout 0.2	-	-
Dense	128	relu
Dropout 0.2	-	-
Dense	64	relu
Dropout 0.1	-	-
Dense	38	softmax / sigmoid

- 實驗結果

	Softmax	Sigmoid
Training F1 Score	0.1754	0.5297
Validation F1 Score	0.1460	0.4827
Kaggle Public F1 Score	0.3662	0.4902

3. 請試著分析tags的分布情況.

- 由小到大排序所有tag的數量

Tag Name	Count	Tag Name	Count
FICTION	1672	ALTERNATE-HISTORY	72
SPECULATIVE-FICTION	1448	COMEDY	59
NOVEL	992	AUTOBIOGRAPHY	51
SCIENCE-FICTION	959	BIOGRAPHY	42
CHILDREN'S-LITERATURE	777	SHORT-STORY	41
FANTASY	773	HISTORY	40
MYSTERY	642	COMIC-NOVEL	37
CRIME-FUCTION	368	MEMOIR	35
SUSPENSE	318	SATIRE	35
YOUNG-ADULT-LITERATURE	288	WAR-NOVEL	31
THRILLER	243	AUTOBOIGRAPHICAL-NOBEL	31
HISTORICAL-NOVEL	137	DYSTOPIA	30
HORROR	192	NOVELLA	29
DETECTIVE-FICTION	178	HUMOUR	18
ROMANCE-NOVEL	157	TECHNO-THRILLER	18
HISTORICAL-FICTION	178	HIGH-FANTASY	15
ADVENTURE-NOVEL	109	APOCALYPTIC-AND-POST-APOCALYPTIC-FICITON	14
NON-FICITON	102	GOTHIC-FICTION	12
SPY-FICTION	75	UTIPIAN-AND-DYSTOPIAN-FICTION	11

- 有上面的結果可以看到大部份的文章都會被歸類在FICTION, SPECULATIVE-FICTION, NOVEL, SCIENCE-FICTION.
- 有一半的tag出現的數量少於100

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

- 使用keras的標記生成器(tokenizer)把所有字都標上數字作為標記(token), 並把他對應的結果儲存起來. 然後在把每個文章(article)轉換成標記串, 並補上零使得每個文章長度一樣.

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(all_corpus)
word_index = tokenizer.word_index

train_sequences = tokenizer.texts_to_sequences(train_text)
test_sequences = tokenizer.texts_to_sequences(test_text)

train_data = pad_sequences(train_sequences)
train_label = np.array(train_label)
MAX_SEQUENCE_LEN = train_data.shape[1]
test_data = pad_sequences(test_sequences, maxlen=MAX_SEQUENCE_LEN)
```

- 讀入詞向量 `glove.6b.100d.txt`
- 建立詞矩陣, 每一列(row)是一個詞, 每一行(column)是每個詞的維度.
- 建立嵌入層(embedding layer)

```
from keras.layers import Embedding
embedding = Embedding(num_words, 100, weight=[embedding_matrix], \
                      input_length=MAX_SEQUENCE_LEN, trainable=False)
```

5. 試比較bag of word和RNN何者在本次作業中效果較好.

- 和第二題一樣用同樣的訓練模型, 最後一層的activation使用Sigmoid.

	Bag of Word	**Recurrent NN**
Training F1 Score	0.5403	0.5297
Validation F1 Score	0.5077	0.4827

- Bag of Word會比RNN好一些