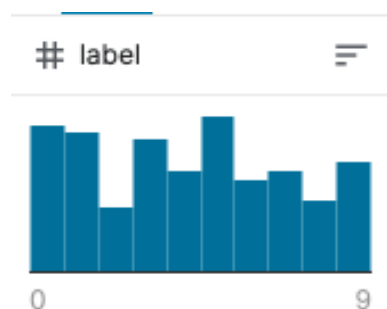


DL_Final 0513460 資財09胡明秀

Data Preprocessing

- We observe that there is an imbalance data problem for this dataset based on the following histogram of labels:



Thus, we simply do a data augmentation by adjusting all the training data to the same scale. We augmented the data to 40,000 per label.

- Training data is formed of "title" and "keyword". Firstly, drop the comma for the titles and the keywords. Secondly, we combine the title and the keyword for each ID. Notice that we also do the same processing for the testing data! This is because we plan to include the testing data's sentences and keywords features later in the embedding layer. Even though the testing data doesn't have labels, we can still leverage from them by including them into the embedding layer for the model.
- We use **jieba**, a popular library for processing Chinese, to prepare the training data we need. However, only splitting out the words doesn't provide enough information for our models. We only pick some certain POS(詞性) for our model to learn.

```
def get_sent_list(sentences):
    sent_list = []
    flag_list = ['n', 'ng', 'nr', 'nrfg', 'nrt', 'ns', 'nt', 'nz', 's', 'j', 'l', 'i', 'v', 'vn', 'eng']

    for num in range(len(sentences)):

        if num % 10000 == 0:
            print("No. %d" % (num))

        sent = sentences[num]
        sent = pseg.cut(sent) # get part of speech

        tmp_list = []
        for word, flag in sent:
            if flag in flag_list:
                tmp_list.append(word)

        sent_list.append(tmp_list)

    return sent_list
```

- Then we use Tokenizer from keras to split out the training inputs / labels we need for our models. We reserve 20% of training data for validation with random state = 42.

LSTM Model 1: Own trained Embedding Layer

- We experiment our first model with own trained embedding layer. There are lots of pre-trained word vectors, such as Bert and XLnet. However, we would like to experiment with own-trained embedding layer first to see the performance. Choosing Epoch = 10, we can get the acc = 0.93603. Notice that we also put back the 20% validation data back and trained the model again. The acc would rise slightly for learning more information from the data.

Model: "sequential_10"		
Layer (type)	Output Shape	Param #
=====		
embedding_10 (Embedding)	(None, 300, 232)	41188816
=====		
lstm_11 (LSTM)	(None, 512)	1525760
=====		
dropout_10 (Dropout)	(None, 512)	0
=====		
dense_15 (Dense)	(None, 10)	5130
=====		
Total params: 42,719,706		
Trainable params: 42,719,706		
Non-trainable params: 0		
=====		


LSTM Model 2: Pre-trained Embedding Layer


- Next, we tried using pre-trained word vector for the embedding layer. The pre-trained word-vectors are from this github repo (Mix-large綜合). However, the acc only goes to 0.93596 with 15 epochs of training. Maybe there are more powerful pre-trained word vectors.

Model: "sequential_2"		
Layer (type)	Output Shape	Param #
=====		
embedding_2 (Embedding)	(None, 300, 300)	53261400
=====		
lstm_2 (LSTM)	(None, 256)	570368
=====		
dropout_2 (Dropout)	(None, 256)	0
=====		
dense_3 (Dense)	(None, 10)	2570
=====		
Total params: 53,834,338		
Trainable params: 572,938		
Non-trainable params: 53,261,400		
=====		

Embedding/Chinese-Word-Vectors

This project provides 100+ Chinese Word Vectors (embeddings) trained with different representations (dense and sparse), context features (word, ngram, character, and more), and corpora. One can easily obtain pre-trained vectors with different properties and use them for downstream tasks.

 <https://github.com/Embedding/Chinese-Word-Vectors>




Bert Fine-tuning: Pre-trained Embedding Layer + Pre-trained Model

- Bert Fine-tuning model only gets acc 0.88158 with training epochs = 4. The paper of Bert notices that when using Bert for fine-tuning, the ideal training epoch = 2 - epoch = 4. However, even though the loss is really low for bert-model, the acc didn't reach 0.9 for testing. Maybe I didn't utilize the beauty of BERT.

Data for the code

- 以下為code裡面會需要的data以及pre-trained詞向量：

0513460_ECM_DL_Final_Competition_Data - Google Drive

 https://drive.google.com/drive/folders/14sBk9O_-8hum33IsDkMlfoJUmPEThMyg?usp=sharing

