Kaggle Report

109062607 黃麟凱

Preprocessing

```
labels = pd.read_csv("data/emotion.csv")
all_data = pd.read_csv("data/data_identification.csv")
all_data = all_data.merge(labels,how='left', left_on='tweet_id', right_on='tweet_id')
```

I used “read_csv” to make the datas into a dataframe and used “merge” to merge datas from different files depending on their tweet_id.
Then, datas look like this:

	tweet_id	identification	emotion	text
0	0x28cc61	test	NaN	@Habbo I've seen two separate colours of the e...
1	0x29e452	train	joy	Huge Respect @JohnnyVegasReal talking about l...
2	0x2b3819	train	joy	Yoooo we hit all our monthly goals with the ne...
3	0x2db41f	test	NaN	@FoxNews @KellyannePolls No serious self respe...
4	0x2a2acc	train	trust	@KIDSNTS @PICU_BCH @uhbcomms @BWCHBoss Well do...
5	0x2a8830	train	joy	Come join @ambushman27 on #PUBG while he striv...

In the end, I separated datas into training and testing datas by the column of identification, and splited training datas into 80% training datas and 20% validation datas.

```
train_data = all_data.loc[all_data['identification'] == 'train']
test_data = all_data.loc[all_data['identification'] == 'test']
train_data, val_data =  np.split(train_data.sample(frac=1, random_state=42),
                       [int(.8*len(train_data))])
```

Models

In this competition, I totally used two models to solve the problem. Here’re they.

I. W2V+DNN

I used nltk to tokenize all texts. Next, I used word2vec to convert all tokens into vectors, and averaged all token vectors in a sentence to be a sentence's embedding.

```
all_data['text_tokenized'] = all_data['text'].apply(lambda x: nltk.word_tokenize(x))#[word fo
```

```
from gensim.models import KeyedVectors

## Note: this model is very huge, this will take some time ...
model_path = "../GoogleNews/GoogleNews-vectors-negative300.bin.gz"
w2v_google_model = KeyedVectors.load_word2vec_format(model_path, binary=True)
```

```
def sentence_embedding(text):
    w2v_list = [w2v_google_model.wv[word] for word in text if word in w2v_google_model.vocab]
    return np.mean(w2v_list, axis=0)
```

```
all_data['sent_embd'] = all_data['text'].apply(lambda x: sentence_embedding(x))
```

Finally, I took keras to produce a model with two fully connected layers to classify and I got only 0.35 on accuracy.

w2v_3dnn.csv 0.35109

5 days ago by KAI

w2v with 3dnn

II. Bert

I cloned the code implementing bert from a repository of google-research(https://github.com/google-research/bert) and modified some code to train and predict.

First of all, I need to convert the datas into the format bert can read:

	guid	label	alpha	text
1	0x29e452	joy	a	Huge Respect @JohnnyVegasReal talking about l...
2	0x2b3819	joy	a	Yoooo we hit all our monthly goals with the ne...
4	0x2a2acc	trust	a	@KIDSNTS @PICU_BCH @uhbcomms @BWCHBoss Well do...
5	0x2a8830	joy	a	Come join @ambushman27 on #PUBG while he striv...

P.S. The alpha is an extra field bert need, but I don’t use other data except for text, so I use “a” to represent alpha for each row. I thought that wouldn’t influence the result of the prediction.

the code is here:

```
df_bert_train = pd.DataFrame({'guid': train_data['tweet_id'],
    'label': train_data['emotion'],
    'alpha': ['a']*train_data.shape[0],
    'text': train_data['text']})

df_bert_dev = pd.DataFrame({'guid': val_data['tweet_id'],
    'label': val_data['emotion'],
    'alpha': ['a']*val_data.shape[0],
    'text': val_data['text']})

df_bert_test = pd.DataFrame({'guid': test_data['tweet_id'],
    'label': ['joy']*test_data.shape[0],
    'alpha': ['a']*test_data.shape[0],
    'text': test_data['text']})


df_bert_train.to_csv('dataset_nodev/train.tsv', sep='\t', index=False, header=False)
df_bert_dev.to_csv('dataset/dev.tsv', sep='\t', index=False, header=False)
df_bert_test.to_csv('dataset_nodev/test.tsv', sep='\t', index=False, header=True)
```

Next, I only change the labels of google's code, and start to train.

```
def get_labels(self):
  """See base class."""
  return ['joy', 'trust', 'anticipation', 'sadness', 'disgust', 'fear', 'surprise', 'anger']
  # return ["0", "1"]
```

In my best result, my parameters is set like this:

```
CUDA_VISIBLE_DEVICES=0 nohup python run_classifier.py --task_name=cola --do_train=true --data_dir=./dataset --vocab_file=./model/vocab.txt --bert_config_file=./model/bert_config.json --init_checkpoint=./model/bert_model.ckpt --max_seq_length=64 --train_batch_size=32 --learning_rate=2e-5 --num_train_epochs=3.0 --output_dir=./bert_output/ --do_lower_case=False --save_checkpoints_steps 5000 &
```

learning rate: 2e-5 epoch:3 batch size: 32 sequence_length: 64

In the end, the accuracy is 0.538 and made me on the twelfth place, which is top 18% in this competition.

conclusion

Obviously, bert had much better result than DNN with W2V. I think the reason is that bert is a robust model because of its structure(self-attention and deep) and its pre-trained model which was trained by many datas. In addition, bert can be regarded as a time series model, but DNN can't. In the NLP task, time series model is more proper, because the order of the word in the sentence is meaningful.
I am glad to attend this competition, which made me implement bert without others' help. Thank TA!