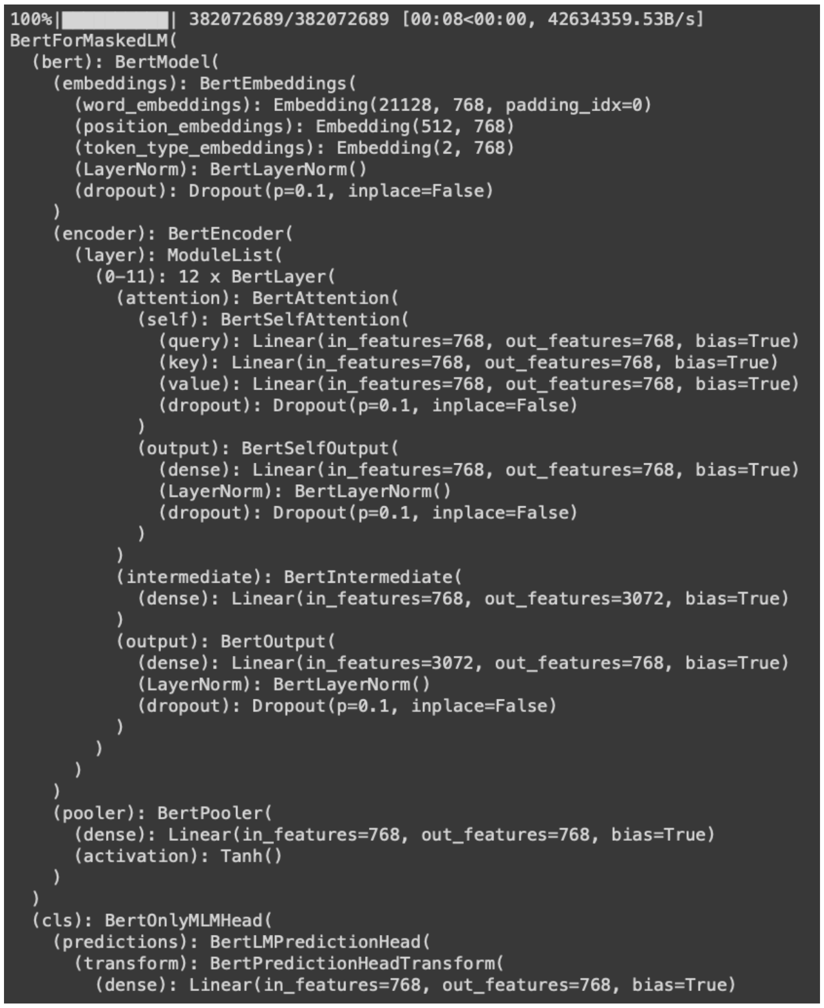
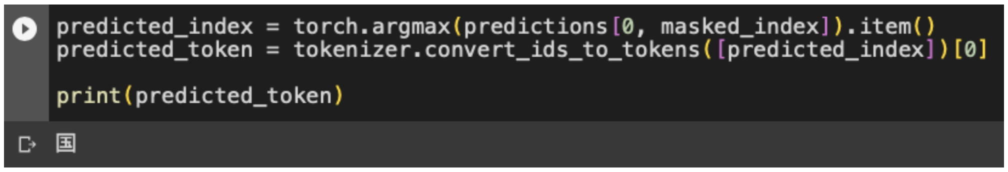
Kevin King
COSC 72: Accelerated Computational Linguistics
Exercise Week 6

Part 1: BERT masked words in Mandarin Chinese



```
100%|██████████| 382072689/382072689 [00:08<00:00, 42634359.53B/s]
BertForMaskedLM(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(21128, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): BertLayerNorm()
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): BertLayerNorm()
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): BertLayerNorm()
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (cls): BertOnlyMLMHead(
    (predictions): BertLMPredictionHead(
      (transform): BertPredictionHeadTransform(
        (dense): Linear(in_features=768, out_features=768, bias=True)
```



```
predicted_index = torch.argmax(predictions[0, masked_index]).item()
predicted_token = tokenizer.convert_ids_to_tokens([predicted_index])[0]

print(predicted_token)
```

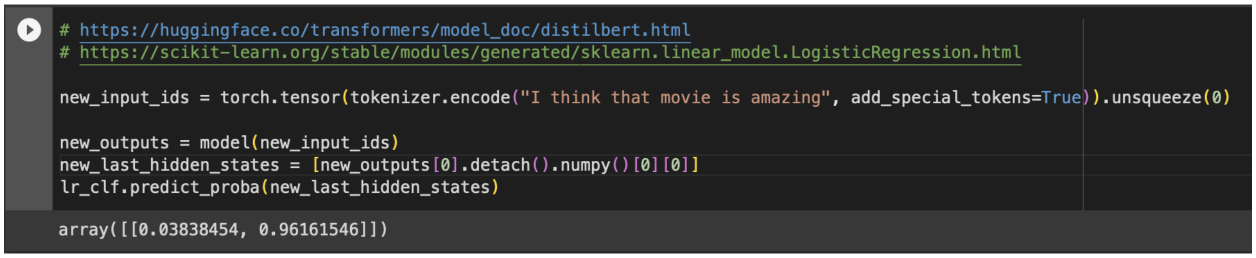
```
国
```

The lines I changed in the code included:

1) Changing the pre-trained model tokenizer to 'bert-base-chinese'
2) Changing the pre-trained model (weights) to 'bert-base-chinese'
3) The input text to '你是中 [MASK] 人吗'

Part 2: Using a BERT encoding to classify sentiment in English

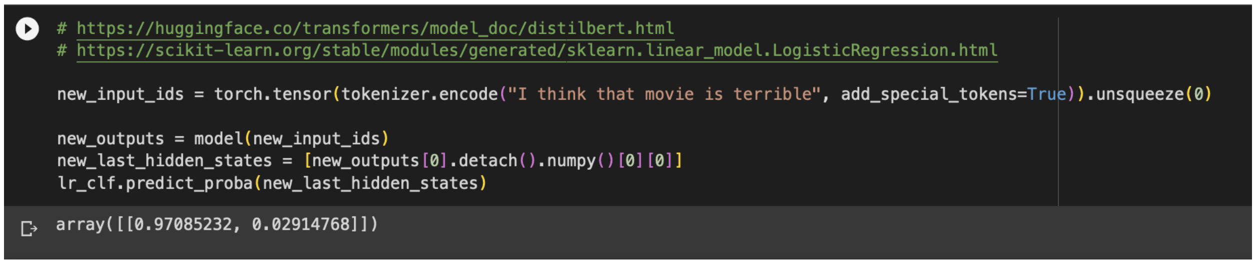
(1) *Please turn in a PDF/LibreOffice/Word document with an explanation of your results. What is the new code giving you as output? What is the meaning of the numbers in this array?*



```
# https://huggingface.co/transformers/model_doc/distilbert.html
# https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

new_input_ids = torch.tensor(tokenizer.encode("I think that movie is amazing", add_special_tokens=True)).unsqueeze(0)

new_outputs = model(new_input_ids)
new_last_hidden_states = [new_outputs[0].detach().numpy()[0][0]]
lr_clf.predict_proba(new_last_hidden_states)

array([[0.03838454, 0.96161546]])
```

The new code outputs a two-dimensional array of length 1 and an array of length 2 inside it with two floating point numbers. While the value of the first number is fairly small (0.0383), the second number is fairly large (0.962), which means that the first is the probability of the review having a negative connotation and the second is the probability of it having a positive connotation.

(2) *Change the string "I think that movie is amazing" for any other string in English. Make it 6~10 words long. Run the code again and explain the results you get. Please turn in a screenshot of your new code and the resulting array.*



```
# https://huggingface.co/transformers/model_doc/distilbert.html
# https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

new_input_ids = torch.tensor(tokenizer.encode("I think that movie is terrible", add_special_tokens=True)).unsqueeze(0)

new_outputs = model(new_input_ids)
new_last_hidden_states = [new_outputs[0].detach().numpy()[0][0]]
lr_clf.predict_proba(new_last_hidden_states)

array([[0.97085232, 0.02914768]])
```

I changed the string to "I think the movie is terrible," which outputted the above array as a result, in which the first floating point number was now substantially larger than the second. This further backs the reasoning for the first being the probability of a negative review and the second being the probability of a positive review.