CS156 Final Project: Mandarin Chinese Text Classification

1. Problem Definition

In this final project, I aim to perform text classification on Mandarin Chinese news articles. Specifically, I will input raw news articles into the model, preprocess the model and then tag a label to the news. This is particularly hard for the Chinese text due to the difficulty in word separation (WS) in the tokenization phase. This is would also be a problem for further analysis, such as in sentiment analysis, but that is beyond the scope of the project. Simply, I will train and compare different models for classifying Mandarin Chinese text and then test the model by using it on unseen, unlabeled data to see the actual labeling in real life.

In the beginning, I will introduce the preprocessing necessary for normal Chinese text to be used as input our model and relevant datasets and packages. The following section will focus on the classification models. I will use three different models to compare and contrast their efficacy, namely, Convolutional Neural Networks (CNN), Long Short-Term Memory Networks (LSTM), and Bidirectional Encoder Representations from Transformers (BERT). I then focus on discussing the metrics of the models and using the three models on classifying unseen data, which can only be visually inspected due to the lack of labeling of this data. Lastly, I will evaluate the advantages and disadvantages of each model in Mandarin Chinese classification in the test and analysis section. I will also touch on the challenges from the preprocessing in Chinese text, overfitting, and future directions.

2. Solution Specification

Preprocessing

This section will introduce the tokenization in Mandarin Chinese and datasets/packages used in the Final Project. For Mandarin, lemmatization is not applicable due to the usage of Chinese characters that do not have any inflections according to grammatical changes. However, this also presents a challenge semantically as the exact same characters will have different functions in the sentence based on the exact position and context.

Datasets and Packages

1. `Kashgari` package is used to help me easily manipulate different aspects of the natural language processing (NLP) pipeline, including tokenization, model building, and visualizations. This package helps us build different models without actually building them from scratch, expediting the process of text classification to only import data and fine-tuning the models (Kashgari, 2021).

2. `SMP2018ECDT` is the dataset used for training the models. Released by Evaluation of Chinese Human-Computer Dialogue Technology, SMP2018ECDT contains 31 labels and

numerous text titles. This data is used to train the model due to the simplicity of the data.

3. `THUCNews` (THU Chinese News) is a database compiled by Tsinghua University in China. In this final project, I imported unseen text from this dataset into the three models in order to see how they will classify the text. Unfortunately, the labels do not match, so a quantitative analysis cannot be made.

Tokenization

Due to the difference between alphabets and Chinese characters, tokenization in the two languages is very different. Without spaces for splitting words, tokenization in Mandarin Chinese resorts to dictionaries and pre-trained models that store the tokenization rules in Chinese. The following is some popular models for this.

1. **jieba**:

   Most widely used tokenizer in the Chinese language due to its simplicity and history. However, its performance is not the state-of-art anymore. `jieba` uses a dictionary and statistics to predict the separation of words. It has different modes for exact word separations or search engine word separations catering to different needs.

2. **CKIPtagger**:

   CKIPtagger is NLP package for Traditional Mandarin Chinese for the purpose of Word Separation (WS), Named Entity Recognition (NER), and Part of Speech (POS) tagging. According to the documentation of the package, it is said that the accuracy is higher than `jieba` . However, it is not widely used and tested.

In `Kashgari` , there exists a tokenizer for the BERT model but not for other models. Therefore we would need to tokenize the text using the aforementioned models. Fortunately, the `SMP2018ECDT` dataset is already tokenized in the `kashgari` module; hence I did not use these models for the training data. However, for the `THUCNews` dataset, I used `jieba` to tokenize the data in the classification section.

In [1]:

```python
# Import pacakges
import kashgari
from kashgari.corpus import SMP2018ECDTCorpus
from kashgari.tasks.classification import BiGRU_Model
from kashgari.tasks.classification import BiLSTM_Model
from kashgari.tasks.classification import CNN_Model
from kashgari.embeddings import BertEmbedding
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

```
/Users/enjuichang/opt/anaconda3/lib/python3.7/site-packages/gensim/similarities/
__init__.py:15: UserWarning: The gensim.similarities.levenshtein submodule is di
sabled, because the optional Levenshtein package <https://pypi.org/project/pytho
n-Levenshtein/> is unavailable. Install Levenhstein (e.g. `pip install python-Le
venshtein`) to suppress this warning.
  warnings.warn(msg)
```

In [2]:
```
# Import the SMP2018ECDT dataset in Kashgari
train_x, train_y = SMP2018ECDTCorpus.load_data('train')
valid_x, valid_y = SMP2018ECDTCorpus.load_data('valid')
test_x, test_y = SMP2018ECDTCorpus.load_data('test')
print("Example data: ", train_x[2])
print("Example label: ", train_y[2])
```

```
2021-04-22 00:44:28,927 [DEBUG] kashgari - loaded 1881 samples from /Users/enjui
chang/.kashgari/datasets/SMP2018ECDTCorpus/train.csv. Sample:
x[0]: ['唠', '什', '么']
y[0]: chat
2021-04-22 00:44:28,933 [DEBUG] kashgari - loaded 418 samples from /Users/enjuic
hang/.kashgari/datasets/SMP2018ECDTCorpus/valid.csv. Sample:
x[0]: ['电', '视', '台', '放', '什', '么']
y[0]: epg
2021-04-22 00:44:28,943 [DEBUG] kashgari - loaded 770 samples from /Users/enjuic
hang/.kashgari/datasets/SMP2018ECDTCorpus/test.csv. Sample:
x[0]: ['这', '有', '啥', '意', '思']
y[0]: chat
Example data:  ['出', '门', '去', '年', '是', '个', '这', '么', '快', '呀']
Example label:  chat
```

Model Training

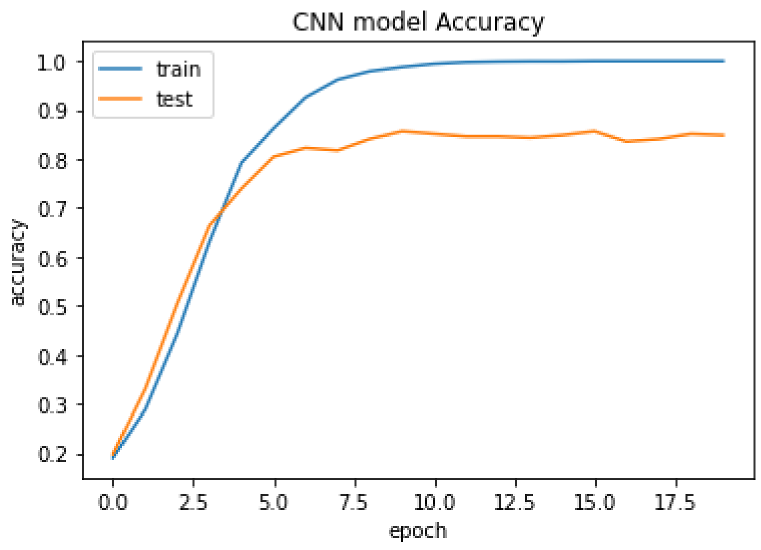
In this section, I will train three different models in order to perform text classification and compare the results. `Kashgari` package will be used for simplicity of the models, while I created a function called `plot_train_hist` to visualize the model's training process.

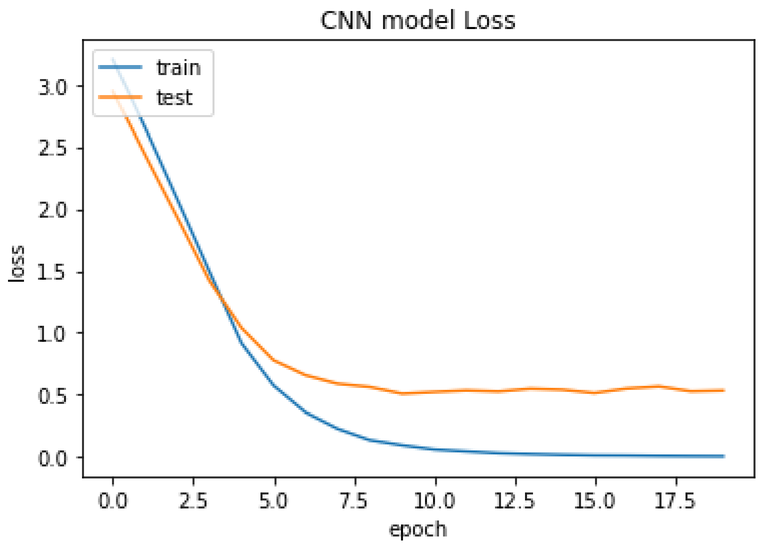
Convolutional Neural Network (CNN)

Here I constructed a CNN model with `adam` optimizer and the `accuracy` as the metrics. The model consists of one embedding layer, one 1D convolutional layer, one global 1D max pooling, and two dense layers. The output has an activation function of 31 neurons for each label in the dataset.

CNNs can be used in text classification by introducing an embedding layer in the model. This layer converts the text into vectors so that the convolution layer can identify different patterns in the text vectors suitable for classification. To reduce overfitting, max pooling is used to make the patterns "hazy" by generalizing the patterns of a certain area, reducing the resolution of the convolution layer. In the end, we use the patterns found my the convolution layer and input that to a series of dense layers and activate the output based on the number of labels in the model (Choubey, 2021).

In [5]:
```
# Plot training history
plot_train_hist(CNN_history, "CNN model")
```





From the training history, we see that the model tends to overfit the training data, while the test accuracy hovers at 0.8.

Bidirectional Long Short-Term Memory Network (BiLSTM)

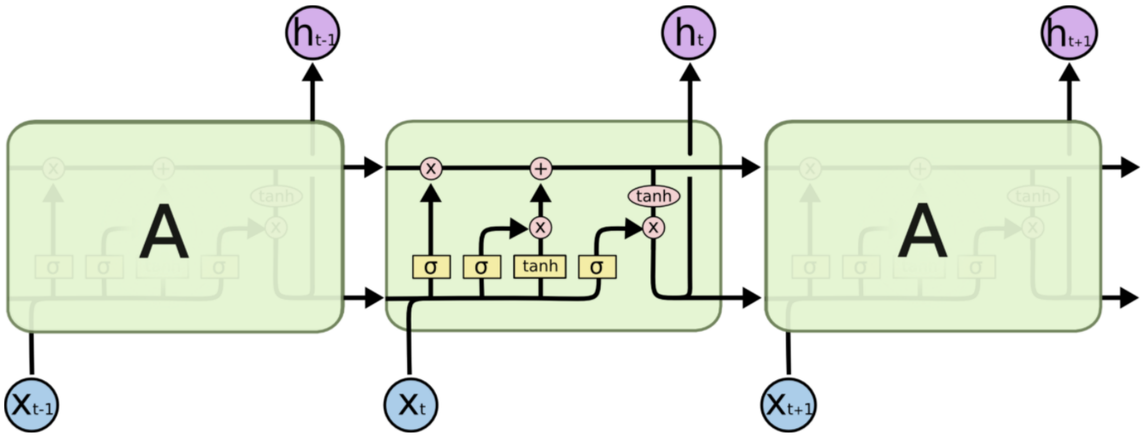
Here I constructed a BiLSTM model with `adam` optimizer and the `accuracy` as the metrics. The model consists of one embedding layer, two bidirectional LSTM layers, three dense layers, and two dropout layers. The output has an activation function of 31 neurons for each label in the dataset.

BiLSTMs can be used in text classification by introducing an embedding layer in the model. This layer converts the text into vectors so that the following layers (bidirectional LSTM layers) can identify store memories in the text vectors suitable for classification. To reduce overfitting, two dropout layers are used to avoid dependencies on layers that overfit the training set. In the end, we use the memories/sequences obtained from the BiLSTMs and input that to a series of dense layers and activate the output based on the number of labels in the model.

BiLSTMs are a variant of LSTM, which is a Recurrent Neural Network (RNN). LSTM is capable of storing long-term memories when solving problems down the pipeline. Simply put, the LSTM layer consists of three parts at each cell (Olah, 2015):

1. **Decide what information we're going to throw away from the cell state**: Based on the previous memories, decide what to remember for the previous cells.
2. **Decide what new information we're going to store in the cell state**: Decide what to remember and forget in this cell and update the old cell accordingly.
3. **Decide what we're going to output**: Based on the cell states, output a filtered version of the cell states using `tanh` function in the cell.

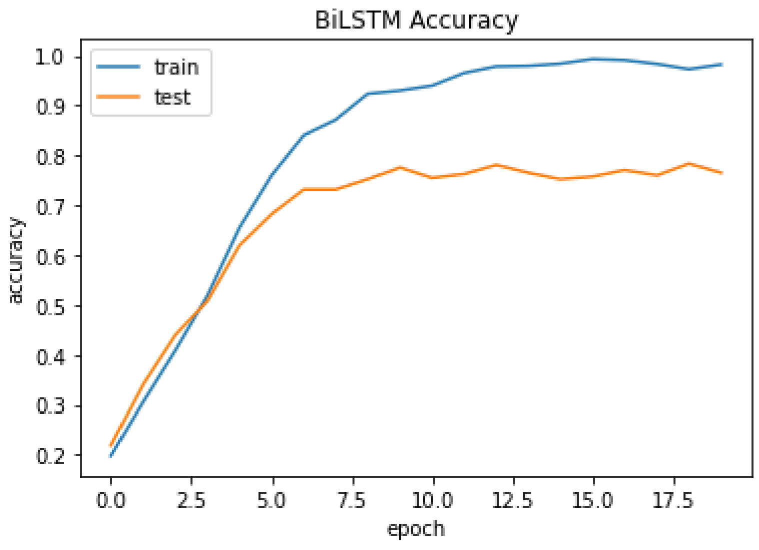
A bidirectional LSTM means that the LSTM is used forward and backward to fine-tune the actual output of the neural network. The major advantage of this model is that it can store past memories, which is especially useful for language models as past information is relevant to future words or classifications.

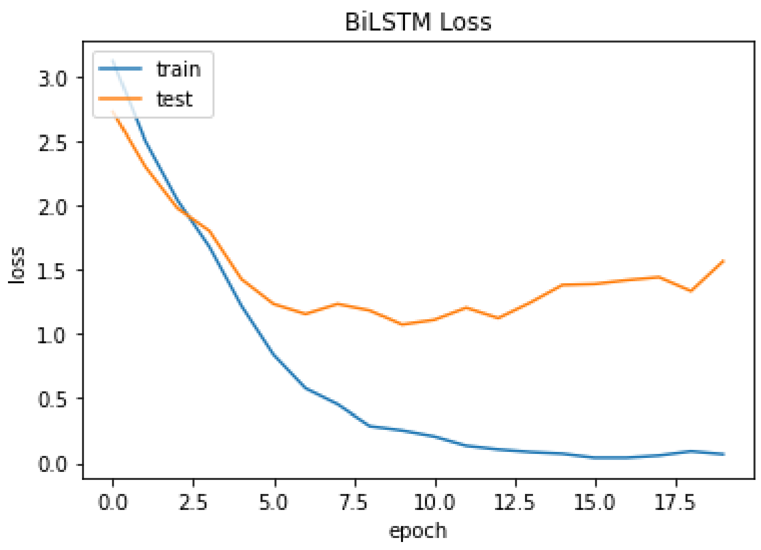


The repeating module in an LSTM contains four interacting layers.

Basic layout of a LSTM model. Retrived from Olah (2015).

In [8]:
```
# Plot the training history
plot_train_hist(BiLSTM_history, "BiLSTM")
```





From the training history, we see that the model tends to overfit the training data, while the test accuracy hovers at 0.7.

Bidirectional Encoder Representations from Transformers model (BERT)

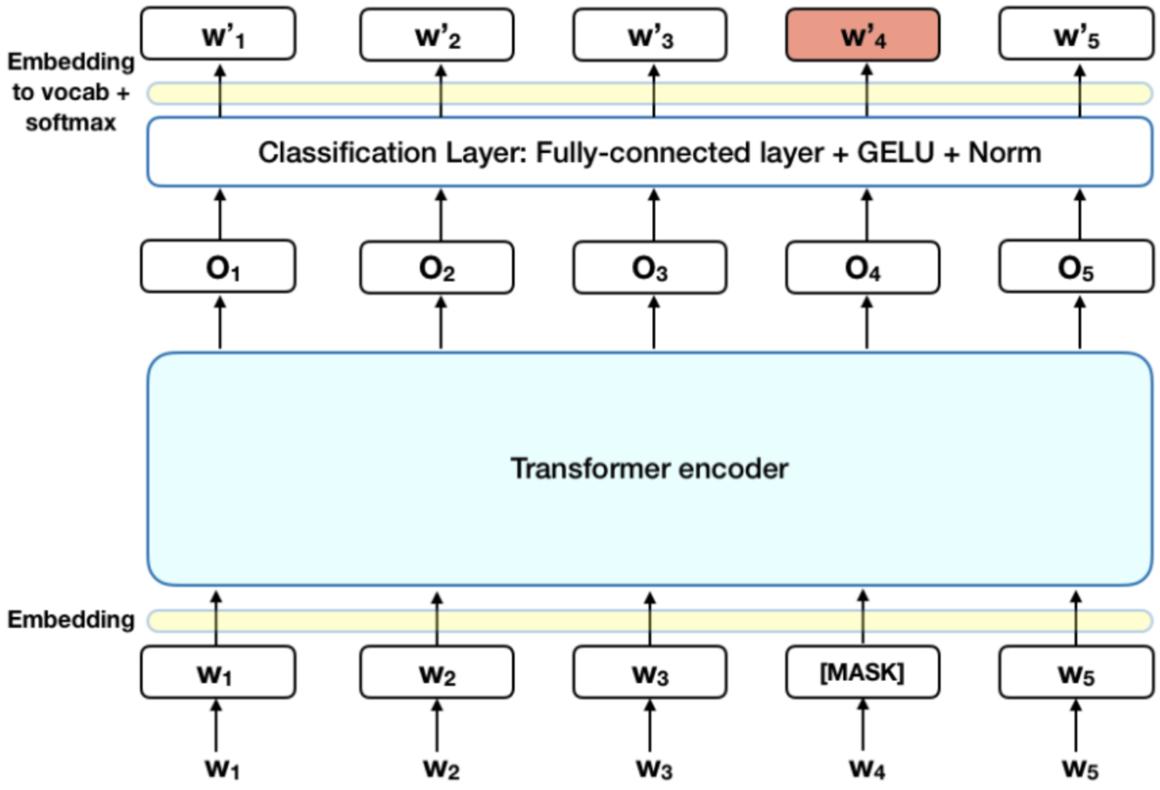
Here I constructed a BERT model with `adam` optimizer and `accuracy` as the metrics. The model consists of the BERT pre-trained model downloaded online, one bidirectional LSTM layer, and one dense layer. The output has an activation function of 31 neurons for each label in the dataset.

BERT is constructed to perform tasks in NLP. Since the pre-trained model should be pretty general, I did not include dropout layers to reduce overfitting.

The BERT model uses Transfomers, which uses attention mechanism to identify contextual information in the text. Transformers have an encoder and decoder that reads the text and produce a prediction for the task, respectively. In the BERT model, some of the word sequences are replaced by a mask token; the model then tries to predict the original value based on the surrounding non-masked words. After that, the BERT model input the word sequence encoder and classify through classification layer and a softmax activation layer (Horev, 2018).

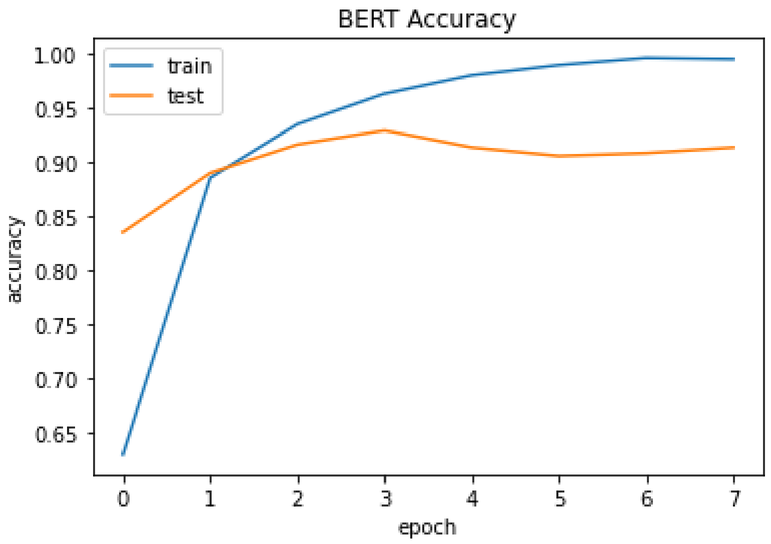
The attention mechanism is the core process for the transformer encoder to work. The idea behind this is to compute a weight distribution on the input sequence and assign higher values to more relevant elements in order to dynamically highlight the relevant features in a word sequence. By doing so, the encoder is able to perform directional-less memory and extract the most relevant feature at each word (Galassi et al., 2020).

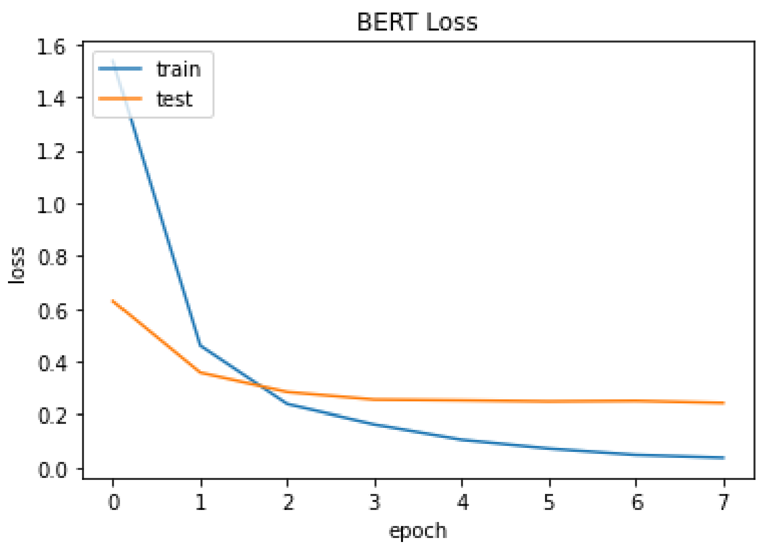
The BERT model used in this project is pre-trained specifically for Mandarin Chinese and was loaded before the training. The training itself was only conducted on the bidirectional LSTM layer and the dense layer for classification -- the pre-trained BERT model was not fine-tuned and was frozen.


Embedding to vocab + softmax
w'1
w'2
w'3
w'4
w'5
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

Schematic example of the BERT model. Retrieved from Horev (2018).

In [10]:
```
# Plot the training history
plot_train_hist(bert_model_history, "BERT")
```





3. Testing and Analysis

After training each model, I saved the trained models and loaded them here to analyze the results of the models when inputted with test data.

In [49]:
```
# Load saved model
CNN_loaded = CNN_Model.load_model('./CNN')
CNN_loaded.evaluate(test_x,test_y)
print(" ")
```

```
2021-04-22 02:33:28,708 [WARNING] kashgari - Sequence length is None, will use t
he max length of the samples, which is 67
2021-04-22 02:33:28,755 [DEBUG] kashgari - predict input shape (770, 67) x:
[[  2  66  20 ...   0   0   0]
 [  2 297   1 ...   0   0   0]
 [  2 374 764 ...   0   0   0]
 ...
 [  2  73 116 ...   0   0   0]
 [  2  66   5 ...   0   0   0]
 [  2 375 169 ...   0   0   0]]
2021-04-22 02:33:29,241 [DEBUG] kashgari - predict output shape (770, 31)
2021-04-22 02:33:29,248 [DEBUG] kashgari - predict output argmax: [ 0  0  1  1
0 13  7 13 14  0  2 10 13  0 21  4  2 10 24  0  2 11 26 14
  0  0  1 18 11  0  0 13  0  1  0  2  2  8  0 20  9 13  2  0  0  2  2  2
  7  6 19  0  0  0  1  6 16 12 23 27 22  1  0  0  1 13  0  3  0  1  0  8
  1 16 11 18 17  0 12  0  1 15  6 13 14  0  0  2 28  0  1  2 12  7  0  2
  2 15  8 26  9  9  3  0  1 17  2  2 19  4  4 12  9 14  8  0 20 17  1  8
  0  2  1  0 27  0 13  9 11 12  5  3  1 25  1  1 11 12  1  5 18  0 20  0
  0  2 26 14  9 19  5  3  1  2 28  0  0  2 30 14  4  0 27  3  8  7 20 16
 17 11 26 10 24  0 15  7 15  2 17  2  1 15  8  1  3  9  3  1 21 17  6 14
  0 20  5 19  1  0  0  3 13 16 19  0  0  3 21  9  0 15 14  0  9  6 16  3
  0  1  1  0  8  4  0  0 14  5  0  1  1  2 26 11 14 12  5  5 10 16  0 14
 16  6  4  6 10  9 11  0  9 15  0  0  4  1  4  1  1  8  7  1 11  1 25 16
 12  0 29 27  0  3 17 19 12  3  9 17  1  4 13 15  1 20  0  1  1 17 13  0
 11 15 22  3  4  0  6  1 23  0  0  2  2  1 21  0  0 11 12  2  3  0  9 16
  1 24  6 19 18 17  0 17  9  9  1 10  2  0  0  2  7  4 12  0 15  0  1 18
 25  1  3  7 28  9  0  1  0  4 27 17  2 17  1  2 18  7  4  2  5  1 16 28
  0 18  0 18  1  6 10 23 17 20  0 29  0 25  0  0  0  4 22  4  1 15 16  0
 12  0 20  1 18  3  0 29 30  1  8 27 30  6 14  3 11  6 24 20  1  7 23  0
  2  2 10  7  7  3 21 25  1  2  5  0 10 21  8  3  0 13  6  0  2  0  0  2
 19  0 15  3  0 28 11  0  1  0  8 10  4  9 29 20 11 16  0  8  0  4 18 20
  6  2 11 18 10  6 18  1 11  1 22  1 17  2  0  8  0  3 16 11 17  6  4  1
  0  2  4 10  0  5  0 18  6  3  9 10  2  0 10 17 13  2  2  5  0  8 10  8
 12  0  9  5  0 10 21  1 19 13 12  1 24  0 29  9  2  1  1 13 23  7  0  5
 10 10  7 29  0 18  1  7  9  0 17  2  0  0  0  7  1  1  4  1  8  4 13 11
 14  2 12 22  7 21  4  1  1 15  1 24  0 28 12  0  4 22  1  0 27 23 14  3
  4  0  0  0  0 10  3  0 23 18 16  0 17  0  1  0  7  2  1  5  0  1 16  0
 18  0  0  0  6  0  2 15  2 27 15  5  1  3  1 28 10  7  5  2 24 22 10 12
  0  9  5  1  0  2  0 29  3 15  2 12 10  8 25  2  0  0  7 14 20 12  1  5
 30  3 15  0  1  0  6  3  5  2  3 24 11 17 16 18 14 15 14  0 25 11  0  5
  0 10  2  2  3  1 14 13 14  8  0  2  1  0  3  1 15 16 16  0 15  0 13  0
 13 21 19  2  0  1  1  0 10 19  0 15  1  5  8  6  3  9  4  1  2  3  5  8
 30  2 17 28  2  1  0 26 14  9  1  2  0 15  0  0 16 18  3 17  0  4  0  2
  0  1  1  2  6  2  8  0  3  2 13  0  6 25  1  0  4 22  2  0  3 15 11  2
  0 16]
              precision    recall  f1-score   support

         app     0.5909    0.7222    0.6500        18
         bus     1.0000    1.0000    1.0000         8
        calc     1.0000    0.7500    0.8571         8
        chat     0.9119    0.9416    0.9265       154
     cinemas     0.6250    0.6250    0.6250         8
    contacts     0.8333    1.0000    0.9091        10
    cookbook     0.9773    0.9663    0.9718        89
    datetime     0.5714    0.6667    0.6154         6
       email     1.0000    1.0000    1.0000         8
         epg     0.9259    0.6944    0.7937        36
      flight     1.0000    0.9524    0.9756        21
      health     0.7778    0.7368    0.7568        19
     lottery     0.8750    0.8750    0.8750         8
         map     0.8750    0.9130    0.8936        23
       match     1.0000    1.0000    1.0000         8
     message     1.0000    0.9524    0.9756        21
       music     0.9545    0.9545    0.9545        22
        news     0.8696    1.0000    0.9302        20
       novel     1.0000    0.8750    0.9333         8
      poetry     0.9444    1.0000    0.9714        34
       radio     1.0000    0.6250    0.7692         8
      riddle     1.0000    1.0000    1.0000        11
    schedule     0.8889    0.8889    0.8889         9
       stock     0.9091    0.8333    0.8696        24
   telephone     1.0000    0.9524    0.9756        21
       train     0.9583    0.9583    0.9583        24
 translation     1.0000    0.9048    0.9500        21
   tvchannel     0.7273    0.6957    0.7111        23
       video     0.6818    0.7500    0.7143        60
     weather     0.9524    0.9091    0.9302        22
     website     0.7000    0.7778    0.7368        18

    accuracy                         0.8870       770
   macro avg     0.8887    0.8684    0.8748       770
weighted avg     0.8929    0.8870    0.8877       770
```

In [48]:

```python
# Load saved model
BiLSTM_loaded = BiLSTM_Model.load_model('./BiLSTM')
BiLSTM_loaded.evaluate(test_x,test_y)
print(" ")
```

```
2021-04-22 02:33:02,071 [WARNING] kashgari - Sequence length is None, will use t
he max length of the samples, which is 67
2021-04-22 02:33:02,115 [DEBUG] kashgari - predict input shape (770, 67) x:
[[  2  66  20 ...   0   0   0]
 [  2 297   1 ...   0   0   0]
 [  2 374 764 ...   0   0   0]
 ...
 [  2  73 116 ...   0   0   0]
 [  2  66   5 ...   0   0   0]
 [  2 375 169 ...   0   0   0]]
2021-04-22 02:33:28,202 [DEBUG] kashgari - predict output shape (770, 31)
2021-04-22 02:33:28,207 [DEBUG] kashgari - predict output argmax: [ 0  0  1  1
0 13  3 13 14  2  4 10 13 19 21  4  5 10 14  0  2 11 14 14
  0  0  1 18 11  0  0 13  0  1  0  2  2  8  0 16  9 13 17  0  0  2 17  2
  7  0 19  0  0  0  1  6  3 12 17 27 22  1  0  0  1 30  0  6  0  1 11  8
  1 17 11  6 17  0 12  0  1 15  6  4  9 11  0  2 28  0  1  2 12  7  0  2
  2 15  8 27  9  9  3  0  1 15  2  2 19  4  5 12  9 14  6  0 20 16  1  8
  0  5  1  0 10  0 13  9 11 12 17  3  1 25  1  1 11 12  1 17 18  0  7  0
  0  2 14 14  2 19  2  3  1  2 28  7  4 17 30 14  4  0 27  3  8  7 20 16
 13 11 27 10 24  0 15  7 15  2 17  2  1 15  8  1  3  0  6  1 21  2  6 14
 12 19  5 19  1  0  0  3 13 17 19  0  9  3 30  9  0 15 14  0  0  6 16  3
  0  1  1 17  8  4  0  0 14 30  0  1  1  2 29 11 14 12  5  5 10 16  0  9
 16  6  4  6 10  9 11  0  8 15  0  0  4  1  4  1  2 14  7  1 11  1 25 16
 12  5 17 27  0  3 17 19 12  3  9 17  1  4 13 15  1 16  0  1  1 17  6  0
 11 15 22  3  4  0 23  1 17  0  0  2  2  1 21  0  0 11 12  2  3  0  0 16
  1 24  6 19 18 17  0 17  9  6  1 10  5 18  0  4  7  4 12  0 15  0  1  8
 24  1  3  7 14  2  7  1  0  4 27 17 13 17  1  2 18  7  4  2  5  1 16 28
  0 13  0 16  1  6 10 17 17 20  0 11  5 25  0  0  0  0 20  4  1 15 17  0
 12  0 20  1  0  3  0 21 30  1  8 27 30  6 14  3 11  0 24 20  1  7 23  0
 23 13 10  7  7 26 21  9  1  5  5  0 10 30  8  3  0 17  6  0  0 12  0 17
 19  9 15 18  0  4 11  0  1  0  8 10  4 10 29 12 11 17  0  8  0  4  9 20
  6 11 11  0 10  6  2  1 11  1 22  1 17  2  0  8  0  6 16 29 17  6  4  1
  0  0  4  9  7  5  1  2 12  3  9 10 11  0 24 17 30  5  6  5  0  8 10  8
 12  0  9  5  0 10 21 18 19 13 12  1 24  0  0  9  4  1  1 17 11  7  0  5
 10 10  7 30  0  2  1  7  9  0 17  2  0  2  0 16  1  1  4  1  8  2 13  2
 14  9 12 22  7 21  4  1  1 15  1 24  0 28 12  0  4 22  1  4 11 23 14  3
  4  0  2  0  0 10 15  0  2 18 16  0 17  0  1  0  7  2  1  5 19  1 16  0
 16  0  0  0  6  0  2 15  0 20 15  5  1 15  1 28 10  7  5  2 24 22 10 12
  0  2  5  1  0  2  0 29  3 15  2 12 10  8 25  2  0  0  7 14 21  0  1  5
 30  8 15  0  2  0  6  3  5  2  6 24 11 17 16 18 14 15 14  0  9 11  0  4
  0 10  5  5  3  1 14 13 14  8  9  4  1  0  3  1 15 17 16  3 15  0 13  0
 13 21 19  2  0  1  8  0 10 19  0 15  1  5  8  6  3  9  4  1  2  3  5  8
 30  2  2 28  2  1  0 26 14  9  1  2  0 30  0  0 16 18  3  0  4  4  0 28
  0  1  1 28  6  2 27  0  3  0  6  0  6 25  1  0  4 22  2  0  3 15 11  6
  0 16]
              precision    recall  f1-score   support

         app     0.4118    0.7778    0.5385        18
         bus     0.7778    0.8750    0.8235         8
        calc     0.5000    0.1250    0.2000         8
        chat     0.8600    0.8377    0.8487       154
     cinemas     0.7500    0.7500    0.7500         8
    contacts     0.7500    0.6000    0.6667        10
    cookbook     0.9765    0.9326    0.9540        89
    datetime     0.5000    0.3333    0.4000         6
       email     1.0000    0.8750    0.9333         8
         epg     0.7647    0.7222    0.7429        36
      flight     0.7826    0.8571    0.8182        21
      health     0.6000    0.3158    0.4138        19
     lottery     0.7500    0.7500    0.7500         8
         map     0.5600    0.6087    0.5833        23
       match     1.0000    0.6250    0.7692         8
     message     0.8182    0.8571    0.8372        21
       music     0.6207    0.8182    0.7059        22
        news     0.8000    1.0000    0.8889        20
       novel     0.5000    0.2500    0.3333         8
      poetry     0.8667    0.7647    0.8125        34
       radio     0.5000    0.7500    0.6000         8
      riddle     0.7857    1.0000    0.8800        11
    schedule     0.7778    0.7778    0.7778         9
```

```
       stock     0.7826    0.7500    0.7660        24
   telephone     0.7000    0.6667    0.6829        21
       train     0.8750    0.8750    0.8750        24
 translation     0.8636    0.9048    0.8837        21
   tvchannel     0.5556    0.6522    0.6000        23
       video     0.6111    0.5500    0.5789        60
     weather     0.6923    0.8182    0.7500        22
     website     0.7059    0.6667    0.6857        18

    accuracy                         0.7636       770
   macro avg     0.7238    0.7125    0.7048       770
weighted avg     0.7710    0.7636    0.7606       770
```

In [47]:
```python
# Load saved model
BERT_loaded = BiGRU_Model.load_model('./bert_model')
BERT_loaded.evaluate(test_x,test_y)
print(" ")
```

```
2021-04-22 02:28:26,719 [DEBUG] kashgari - -------------------------------------
-----------
2021-04-22 02:28:26,730 [DEBUG] kashgari - Loaded transformer model's vocab
2021-04-22 02:28:26,731 [DEBUG] kashgari - config_path       : ./BERT/bert_confi
g.json
2021-04-22 02:28:26,750 [DEBUG] kashgari - vocab_path      : ./BERT/vocab.txt
2021-04-22 02:28:26,775 [DEBUG] kashgari - checkpoint_path : ./BERT/bert_model.c
kpt
2021-04-22 02:28:26,790 [DEBUG] kashgari - Top 50 words    : ['[PAD]', '[unused
1]', '[unused2]', '[unused3]', '[unused4]', '[unused5]', '[unused6]', '[unused
7]', '[unused8]', '[unused9]', '[unused10]', '[unused11]', '[unused12]', '[unuse
d13]', '[unused14]', '[unused15]', '[unused16]', '[unused17]', '[unused18]', '[u
nused19]', '[unused20]', '[unused21]', '[unused22]', '[unused23]', '[unused24]',
'[unused25]', '[unused26]', '[unused27]', '[unused28]', '[unused29]', '[unused3
0]', '[unused31]', '[unused32]', '[unused33]', '[unused34]', '[unused35]', '[unu
sed36]', '[unused37]', '[unused38]', '[unused39]', '[unused40]', '[unused41]',
'[unused42]', '[unused43]', '[unused44]', '[unused45]', '[unused46]', '[unused4
7]', '[unused48]', '[unused49]']
2021-04-22 02:28:26,820 [DEBUG] kashgari - -------------------------------------
-----------
2021-04-22 02:28:54,749 [WARNING] kashgari - Sequence length is None, will use t
he max length of the samples, which is 67
2021-04-22 02:28:54,774 [DEBUG] kashgari - predict input shape (2, 770, 67) x:
(array([[ 101, 6821, 3300, ...,    0,    0,    0],
       [ 101, 7309, 7579, ...,    0,    0,    0],
       [ 101, 7213, 5455, ...,    0,    0,    0],
       ...,
       [ 101,  676, 2128, ...,    0,    0,    0],
       [ 101, 6821,  720, ...,    0,    0,    0],
       [ 101, 1461, 1373, ...,    0,    0,    0]], dtype=int32), array([[0, 0,
0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int32))
2021-04-22 02:32:51,594 [DEBUG] kashgari - predict output shape (770, 31)
2021-04-22 02:32:51,629 [DEBUG] kashgari - predict output argmax: [ 0  0  1  1
0 13  7 13 14  0  2 10 13  0 21 21 23 10 24  0  2 11 26 14
  0  0  1 18 11  0  0 13  0  1  0  2  2  8  0 20  9 13  2  0  0  2 13  2
  7  6 19  0  0  0  1  6 16 12 23 27 22  1  0  0  1 13  0  3  0 18  4  8
  1 16 11 18 16  0 12  0  1 15  6 29  9  0  0  2 28  0  1  2 12  7  0  0
```

```
 2  4  8 26  9  9  3  0  1 15  4  2 19  4  4 12  9 14  8  0 26 13  1  8
 0  2  1  0 27  0  2  9 11 12  5  3  1 25  1  1 11 12  1  5  1  0  0  0
 0  2 26 14  9 19  2  3  1  2  2  0  4  0 30 14  4  0 27  3  8  7 20 16
30 11 26 10 24  0 15  7 15  2 17  2  1 15  8  1  3  0  3  1 21  2  6 14
 0 20  5 19  1  0  2  3 13 16 19  0  0  3  5  9  0 15 14  0  0  6 16  3
 0  1  1  0  8  4  0  0 14  5  0  1  1  2 26 11 14 12  5  5 10 16  0 14
16  6  4  2 10  9 11  0  9 15  0  0  4  1  4  1 18  8  7  1 11  1 25 16
12  2 26 27  0  3 17 19 12  3  9  5  1 28 13 15  1 20  0  1  1 13  2  0
11 15 22  3  4  0  2  1 23  0  0  2  2  1 21  0 18 11 12  2  3  0  9 16
 1 24  3 19 18 22  0 17  9 17  1 10  5  0  0  4  7  4 12  0 15  0  1 18
25  1  3  7 28  9  0  1  0  4 27 17  2 17  1  2 18  7  4  2  5  1 16 28
 0 18  0 18  1  6 10 23 17 20  0 29 28 25  0  0  0  4 22  4  1 15 16  0
12  0 20  1  0  3  0  8 30  1  8 27 30  6 14  3 11  2 24 20  1  7 23  0
 2 13 10  7  7  3 21 25  1  2  2  0 10  9  8  3  0 17  6  0  2 12  0  2
19  0 15  0  0 28 11  0  1  0  8 10  4  9 29 17 11 16  0 14  0  4 18 20
 6  4 11  2 10  6 18  1 11  1 22  1  2  2  0  8  0  3 16 11 17  6  4  1
18  2  4 10  0  5  0 26  6  3  9 10  2  0 10 17 13 29  2  5  0  8 10  8
12  0  9  5  0 10 21  1 19 13 12  1 24  0 21  9 15  1  1 13 23  7  0  5
10 10  7 29  0 18  1  7  9  0 17  2  0  0  0  7  1  1  4  1  8  4 13 11
14 10 12 22  7 21  4  1  1 15  1 24  0 28 12  0  4 22  1  4  8 23 14  3
29  0  0  0  0 10  3  0 23 18 16  0  2  0  1 11  7  2  1  5  0  1 16  0
18  0  0  0  6  0  4 15  2 27 15  5  1  3  1 28 10  7  5  2 24  7 10 12
 0  9 13  1  0  2  0 29  3 15  2 12 10  8 25  2  0  0  7 14 20 12  1  2
30  3 15  0  1  0  6  3  5  2  3 24 11 17 16 18 14 15 14  0 25 11  0  5
 0 10  9  2  3  1 14 13 14  8  0  4  1  0  3  1 15 16 16  0 15  0 13  0
13 21 19  2 18  1  1  1 10 19  0 15  1  5  8  6  3  9 25  1  2  3  5  8
30  4 28 28  2  1  0 26 14  9  1  2  0  5  0  0 16 18  3  7  5  4  0 28
 0  1  1 28  6  2 27  0  3  2  7  0  6 25  1 29  4 22  2  0  3 15 11  8
 0 16]
             precision    recall  f1-score   support

         app     0.8462    0.6111    0.7097        18
         bus     1.0000    1.0000    1.0000         8
        calc     0.8889    1.0000    0.9412         8
        chat     0.9737    0.9610    0.9673       154
     cinemas     0.6667    1.0000    0.8000         8
    contacts     1.0000    0.9000    0.9474        10
    cookbook     1.0000    0.9888    0.9944        89
    datetime     0.7500    1.0000    0.8571         6
       email     0.8750    0.8750    0.8750         8
         epg     0.9091    0.8333    0.8696        36
      flight     1.0000    0.9524    0.9756        21
      health     1.0000    1.0000    1.0000        19
     lottery     1.0000    1.0000    1.0000         8
         map     0.8750    0.9130    0.8936        23
       match     0.8889    1.0000    0.9412         8
     message     0.9130    1.0000    0.9545        21
       music     1.0000    0.8182    0.9000        22
        news     0.8696    1.0000    0.9302        20
       novel     0.8750    0.8750    0.8750         8
      poetry     0.9444    1.0000    0.9714        34
       radio     1.0000    0.7500    0.8571         8
      riddle     1.0000    1.0000    1.0000        11
    schedule     0.7778    0.7778    0.7778         9
       stock     1.0000    0.9583    0.9787        24
   telephone     1.0000    1.0000    1.0000        21
       train     0.9600    1.0000    0.9796        24
 translation     1.0000    0.9524    0.9756        21
   tvchannel     0.8261    0.8261    0.8261        23
       video     0.8548    0.8833    0.8689        60
     weather     0.9545    0.9545    0.9545        22
     website     0.8000    0.8889    0.8421        18

    accuracy                         0.9351       770
   macro avg     0.9177    0.9264    0.9182       770
```

```
weighted avg     0.9383    0.9351    0.9348       770
```

Here, we see that the BERT model has an over 0.9 accuracy in its f1 score as well as a higher accuracy in the recall and precision metrics. For the CNN and BiLSTM models, it seems that BiLSTM performs the worst, possibly due to overfitting, compared to the CNN model where the BiLSTM has a 0.7 f1 score, while the CNN model has a 0.8 f1 score.

On the other hand, in terms of the training time, the BERT model takes the longest time to train (over an hour), while the two other models finished training within 5 minutes. This indicates for training a BERT model with a large dataset, a stronger computational power is needed in order to finish training in a reasonable time. If the classification problem does not require very high accuracy, then possibly the CNN model could be of use when large datasets were encountered.

In addition, we see the impact of overfitting in the training history graphs in the previous sections. Although the training data often has an above 0.9 accuracy, when inputted on test data or validation data, the accuracy drops down to 0.7-0.8 for the BiLSTM and CNN model.

With an assumption of high computation power, the BERT model should be the optimal choice for text classification since the labeling is the most accurate. In the following section, I will test the three models with real news articles from the THUC dataset in order to observe the prediction of the labeling of the text.

Classification on unlabeled data

I first loaded some random samples with the general category already shown in the dataset; however, this data is not labeled in the same way as the labels in `SMP2018ECDT`. Therefore, the resulting classification cannot be measured quantitatively.

In [44]:

```
# Import the unlabeled data
test_unlabeled_x = [storage[i][0] for i in range(len(storage))]
test_unlabeled_y = [storage[i][1] for i in range(len(storage))]
print("Actual classification: ", ['sports', 'entertain', 'land', 'education', 't
```

```
Actual classification:  ['sports', 'entertain', 'land', 'education', 'tech']
```

In [45]:

```
# Run the classification and print the result
print("CNN model: ", CNN_Model.predict(test_unlabeled_x))
print("BiLSTM: ", BiLSTM_model.predict(test_unlabeled_x))
print("BERT: ", bert_model.predict(test_unlabeled_x))
```

```
2021-04-22 02:06:36,391 [DEBUG] kashgari - predict input shape (5, 680) x:
[[2 1 1 ... 1 1 3]
 [2 1 1 ... 0 0 0]
 [2 1 1 ... 0 0 0]
 [2 1 1 ... 0 0 0]
 [2 1 1 ... 0 0 0]]
2021-04-22 02:06:36,485 [DEBUG] kashgari - predict output shape (5, 31)
2021-04-22 02:06:36,486 [DEBUG] kashgari - predict output argmax: [18 18  9 18 2
3]
2021-04-22 02:06:36,490 [DEBUG] kashgari - predict input shape (5, 680) x:
[[2 1 1 ... 1 1 3]
 [2 1 1 ... 0 0 0]
 [2 1 1 ... 0 0 0]
 [2 1 1 ... 0 0 0]
 [2 1 1 ... 0 0 0]]
CNN model:  ['health', 'health', 'map', 'health', 'novel']
2021-04-22 02:06:36,834 [DEBUG] kashgari - predict output shape (5, 31)
2021-04-22 02:06:36,835 [DEBUG] kashgari - predict output argmax: [2 1 2 2 3]
2021-04-22 02:06:36,840 [DEBUG] kashgari - predict input shape (2, 5, 512) x:
(array([[ 101,  100,  100, ..., 4638,  100, 1184],
       [ 101,  100, 2347, ...,    0,    0,    0],
       [ 101,  100, 8038, ..., 8024,  100,  100],
       [ 101, 8151,  100, ...,    0,    0,    0],
       [ 101,  100,  100, ...,  852,  100, 1400]], dtype=int32), array([[0, 0,
0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int32))
BiLSTM:  ['video', 'cookbook', 'video', 'video', 'poetry']
2021-04-22 02:06:44,702 [DEBUG] kashgari - predict output shape (5, 31)
2021-04-22 02:06:44,703 [DEBUG] kashgari - predict output argmax: [4 5 5 5 4]
BERT:  ['epg', 'tvchannel', 'tvchannel', 'tvchannel', 'epg']
```

The outputs of classfication are the following:

0:app, 1:bus, 2:calc, 3:chat, 4:cinemas, 5:contacts, 6:cookbook, 7:datetime, 8:email, 9:epg, 10:flight, 11:health, 12:lottery, 13:map, 14:match, 15:message, 16:music, 17:news, 18:novel, 19:poetry, 20:radio, 21:riddle, 22:schedule, 23:stock, 24:telephone, 25:train, 26:translation, 27:tvchannel, 28:video, 29:weather, 30:website

It seems like the models do not perform in sync when using the models. This could mean that the training data is insufficient since `SMP2018ECDT` dataset only has short sentences. From this example, we see that the performance of the models in real life might not match the theoretical computation, illustrated by classification result from the unlabeled test data.

Future Directions

1. **Tokenization**:

   Due to the writing system in Mandarin Chinese, start-ups and research have been trying to find a new approach to tackle the WS problem in Chinese. One interesting company, called Articut, uses linguistics principles rather than big data and machine learning to perform WS for Mandarin (https://github.com/Droidtown/ArticutAPI), which claims to have higher accuracy compared to `jieba` and `CKIPtagger` packages, demonstrating another approach in solving this problem.

1. **Models**:

   In order to receive an even better performance for the BERT model, it is necessary to fine-tune the model as well as increasing the data size to scale the model. This would require a higher computation power and a longer time to complete. For the other two models, I would expect the establishment of new layers or dropout layers to avoid overfitting and increase the accuracy.

4. Reference

Choubey, V. (2021). Text classification using CNN. Medium. Retrieved 21 April 2021, from https://medium.com/voice-tech-podcast/text-classification-using-cnn-9ade8155dfb9.

Droidtown. (2021). Droidtown/ArticutAPI. GitHub. Retrieved 21 April 2021, from https://github.com/Droidtown/ArticutAPI.

Galassi, A., Lippi, M., & Torroni, P. (2020). Attention in natural language processing. IEEE Transactions on Neural Networks and Learning Systems.

Horev, R. (2018). BERT Explained: State of the art language model for NLP. Medium. Retrieved 21 April 2021, from https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

jieba. (2021). fxsjy/jieba. GitHub. Retrieved 21 April 2021, from https://github.com/fxsjy/jieba.

Kashgari. (2021). Overview — Kashgari 2.0.1 documentation. Kashgari.readthedocs.io. Retrieved 21 April 2021, from https://kashgari.readthedocs.io/en/v2.0.1/.

Olah, C. (2015). Understanding LSTM Networks -- colah's blog. Colah.github.io. Retrieved 21 April 2021, from http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Ya, J. (2021). 中研院開源NLP套件「CKIPtagger」，繁中不結巴. Medium. Retrieved 21 April 2021, from
https://br19920702.medium.com/%E4%B8%AD%E7%A0%94%E9%99%A2%E9%96%8B%E6%BA%9
ckiptagger-%E7%B9%81%E4%B8%AD%E4%B8%8D%E7%B5%90%E5%B7%B4-7822fc4efbf.

5. Appendices

Appendix A. Code for `plot_train_hist` function.

In [3]:
```
def plot_train_hist(history, name):
    """
    This function plots the training history of the model.
    ---
    Input:
    history (object): history of the model according to fit() method.
    name (str): Name of the model

    Output:
    Two figures that show the history of the accuracy and history of the loss
    """
    # Summarize history for accuracy
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title(f'{name} Accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
    # Summarize history for loss
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title(f'{name} Loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
```

Appendix B. Code for training the CNN model.

In [4]:

```
# CNN Model
CNN_Model = CNN_Model()

# Build the model using the training data
CNN_Model.build_model(train_x, train_y)

# Compile model with custom loss, optimizer and metrics
CNN_Model.compile_model(optimizer='adam', metrics = ['accuracy'])

# Train the model
CNN_history = CNN_Model.fit(train_x, train_y, valid_x, valid_y, epochs = 20)

# Evaluate the model
CNN_Model.evaluate(test_x, test_y)

# Save model
CNN_Model.save('CNN')
```

```
Preparing text vocab dict: 100%|██████████| 1881/1881 [00:00<00:00, 272690.65it/
s]
2021-04-22 00:44:29,012 [DEBUG] kashgari - --- Build vocab dict finished, Total:
787 ---
2021-04-22 00:44:29,014 [DEBUG] kashgari - Top-10: ['[PAD]', '[UNK]', '[CLS]',
'[SEP]', '的', '么', '我', '。', '怎', '你']
Preparing classification label vocab dict: 100%|██████████| 1881/1881 [00:00<00:
00, 823537.14it/s]
Calculating sequence length: 100%|██████████| 1881/1881 [00:00<00:00, 1275996.41
it/s]
2021-04-22 00:44:29,044 [DEBUG] kashgari - Calculated sequence length = 15
2021-04-22 00:44:29,095 [DEBUG] kashgari - Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input (InputLayer)           [(None, None)]            0
_________________________________________________________________
layer_embedding (Embedding)  (None, None, 100)         78700
_________________________________________________________________
conv1d (Conv1D)              (None, None, 128)         64128
_________________________________________________________________
global_max_pooling1d (Global (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 64)                8256
_________________________________________________________________
dense_1 (Dense)              (None, 31)                2015
_________________________________________________________________
activation (Activation)      (None, 31)                0
=================================================================
Total params: 153,099
Trainable params: 153,099
```

```
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
29/29 [==============================] - 1s 31ms/step - loss: 3.3346 - accuracy:
0.1672 - val_loss: 2.9447 - val_accuracy: 0.1979
Epoch 2/20
29/29 [==============================] - 0s 9ms/step - loss: 2.7454 - accuracy:
0.2769 - val_loss: 2.4298 - val_accuracy: 0.3307
Epoch 3/20
29/29 [==============================] - 0s 9ms/step - loss: 2.1597 - accuracy:
0.4127 - val_loss: 1.9284 - val_accuracy: 0.5052
Epoch 4/20
29/29 [==============================] - 0s 7ms/step - loss: 1.6452 - accuracy:
0.5762 - val_loss: 1.4194 - val_accuracy: 0.6641
Epoch 5/20
29/29 [==============================] - 0s 8ms/step - loss: 1.0096 - accuracy:
0.7706 - val_loss: 1.0394 - val_accuracy: 0.7396
Epoch 6/20
29/29 [==============================] - 0s 7ms/step - loss: 0.6540 - accuracy:
0.8395 - val_loss: 0.7781 - val_accuracy: 0.8047
Epoch 7/20
29/29 [==============================] - 0s 7ms/step - loss: 0.4059 - accuracy:
0.9090 - val_loss: 0.6576 - val_accuracy: 0.8229
Epoch 8/20
29/29 [==============================] - 0s 7ms/step - loss: 0.2504 - accuracy:
0.9550 - val_loss: 0.5891 - val_accuracy: 0.8177
Epoch 9/20
29/29 [==============================] - 0s 7ms/step - loss: 0.1464 - accuracy:
0.9754 - val_loss: 0.5632 - val_accuracy: 0.8411
Epoch 10/20
29/29 [==============================] - 0s 7ms/step - loss: 0.1094 - accuracy:
0.9834 - val_loss: 0.5088 - val_accuracy: 0.8568
Epoch 11/20
29/29 [==============================] - 0s 9ms/step - loss: 0.0687 - accuracy:
0.9938 - val_loss: 0.5225 - val_accuracy: 0.8516
Epoch 12/20
29/29 [==============================] - 0s 8ms/step - loss: 0.0500 - accuracy:
0.9978 - val_loss: 0.5350 - val_accuracy: 0.8464
Epoch 13/20
29/29 [==============================] - 0s 7ms/step - loss: 0.0371 - accuracy:
0.9994 - val_loss: 0.5264 - val_accuracy: 0.8464
Epoch 14/20
29/29 [==============================] - 0s 7ms/step - loss: 0.0257 - accuracy:
1.0000 - val_loss: 0.5489 - val_accuracy: 0.8438
Epoch 15/20
29/29 [==============================] - 0s 7ms/step - loss: 0.0207 - accuracy:
0.9999 - val_loss: 0.5404 - val_accuracy: 0.8490
Epoch 16/20
29/29 [==============================] - 0s 7ms/step - loss: 0.0160 - accuracy:
1.0000 - val_loss: 0.5140 - val_accuracy: 0.8568
Epoch 17/20
29/29 [==============================] - 0s 7ms/step - loss: 0.0139 - accuracy:
1.0000 - val_loss: 0.5499 - val_accuracy: 0.8359
Epoch 18/20
29/29 [==============================] - 0s 7ms/step - loss: 0.0102 - accuracy:
1.0000 - val_loss: 0.5671 - val_accuracy: 0.8411
Epoch 19/20
29/29 [==============================] - 0s 8ms/step - loss: 0.0086 - accuracy:
1.0000 - val_loss: 0.5279 - val_accuracy: 0.8516
Epoch 20/20
29/29 [==============================] - 0s 8ms/step - loss: 0.0075 - accuracy:
1.0000 - val_loss: 0.5340 - val_accuracy: 0.8490
2021-04-22 00:44:34,799 [WARNING] kashgari - Sequence length is None, will use t
he max length of the samples, which is 67
2021-04-22 00:44:34,813 [DEBUG] kashgari - predict input shape (770, 67) x:
```

```
[[   2  66  20 ...   0   0   0]
 [   2 297   1 ...   0   0   0]
 [   2 374 764 ...   0   0   0]
 ...
 [   2  73 116 ...   0   0   0]
 [   2  66   5 ...   0   0   0]
 [   2 375 169 ...   0   0   0]]
2021-04-22 00:44:35,006 [DEBUG] kashgari - predict output shape (770, 31)
2021-04-22 00:44:35,011 [DEBUG] kashgari - predict output argmax: [ 0  0  1  1
0 13  7 13 14  0  2 10 13  0 21  4  2 10 24  0  2 11 26 14
  0  0  1 18 11  0  0 13  0  1  0  2  2  8  0 20  9 13  2  0  0  2  2  2
  7  6 19  0  0  0  1  6 16 12 23 27 22  1  0  0  1 13  0  3  0  1  0  8
  1 16 11 18 17  0 12  0  1 15  6 13 14  0  0  2 28  0  1  2 12  7  0  2
  2 15  8 26  9  9  3  0  1 17  2  2 19  4  4 12  9 14  8  0 20 17  1  8
  0  2  1  0 27  0 13  9 11 12  5  3  1 25  1  1 11 12  1  5 18  0 20  0
  0  2 26 14  9 19  5  3  1  2 28  0  0  2 30 14  4  0 27  3  8  7 20 16
 17 11 26 10 24  0 15  7 15  2 17  2  1 15  8  1  3  9  3  1 21 17  6 14
  0 20  5 19  1  0  0  3 13 16 19  0  0  3 21  9  0 15 14  0  9  6 16  3
  0  1  1  0  8  4  0  0 14  5  0  1  1  2 26 11 14 12  5  5 10 16  0 14
 16  6  4  6 10  9 11  0  9 15  0  0  4  1  4  1  1  8  7  1 11  1 25 16
 12  0 29 27  0  3 17 19 12  3  9 17  1  4 13 15  1 20  0  1  1 17 13  0
 11 15 22  3  4  0  6  1 23  0  0  2  2  1 21  0  0 11 12  2  3  0  9 16
  1 24  6 19 18 17  0 17  9  9  1 10  2  0  0  2  7  4 12  0 15  0  1 18
 25  1  3  7 28  9  0  1  0  4 27 17  2 17  1  2 18  7  4  2  5  1 16 28
  0 18  0 18  1  6 10 23 17 20  0 29  0 25  0  0  0  4 22  4  1 15 16  0
 12  0 20  1 18  3  0 29 30  1  8 27 30  6 14  3 11  6 24 20  1  7 23  0
  2  2 10  7  7  3 21 25  1  2  5  0 10 21  8  3  0 13  6  0  2  0  0  2
 19  0 15  3  0 28 11  0  1  0  8 10  4  9 29 20 11 16  0  8  0  4 18 20
  6  2 11 18 10  6 18  1 11  1 22  1 17  2  0  8  0  3 16 11 17  6  4  1
  0  2  4 10  0  5  0 18  6  3  9 10  2  0 10 17 13  2  2  5  0  8 10  8
 12  0  9  5  0 10 21  1 19 13 12  1 24  0 29  9  2  1  1 13 23  7  0  5
 10 10  7 29  0 18  1  7  9  0 17  2  0  0  0  7  1  1  4  1  8  4 13 11
 14  2 12 22  7 21  4  1  1 15  1 24  0 28 12  0  4 22  1  0 27 23 14  3
  4  0  0  0  0 10  3  0 23 18 16  0 17  0  1  0  7  2  1  5  0  1 16  0
 18  0  0  0  6  0  2 15  2 27 15  5  1  3  1 28 10  7  5  2 24 22 10 12
  0  9  5  1  0  2  0 29  3 15  2 12 10  8 25  2  0  0  7 14 20 12  1  5
 30  3 15  0  1  0  6  3  5  2  3 24 11 17 16 18 14 15 14  0 25 11  0  5
  0 10  2  2  3  1 14 13 14  8  0  2  1  0  3  1 15 16 16  0 15  0 13  0
 13 21 19  2  0  1  1  0 10 19  0 15  1  5  8  6  3  9  4  1  2  3  5  8
 30  2 17 28  2  1  0 26 14  9  1  2  0 15  0  0 16 18  3 17  0  4  0  2
  0  1  1  2  6  2  8  0  3  2 13  0  6 25  1  0  4 22  2  0  3 15 11  2
  0 16]
2021-04-22 00:44:35,067 [INFO] kashgari - model saved to /Users/enjuichang/Deskt
op/Minerva/Spring 2020/CS156/Final/CNN
              precision    recall  f1-score   support

         app     0.5909    0.7222    0.6500        18
         bus     1.0000    1.0000    1.0000         8
        calc     1.0000    0.7500    0.8571         8
        chat     0.9119    0.9416    0.9265       154
     cinemas     0.6250    0.6250    0.6250         8
    contacts     0.8333    1.0000    0.9091        10
    cookbook     0.9773    0.9663    0.9718        89
    datetime     0.5714    0.6667    0.6154         6
       email     1.0000    1.0000    1.0000         8
         epg     0.9259    0.6944    0.7937        36
      flight     1.0000    0.9524    0.9756        21
      health     0.7778    0.7368    0.7568        19
     lottery     0.8750    0.8750    0.8750         8
         map     0.8750    0.9130    0.8936        23
       match     1.0000    1.0000    1.0000         8
     message     1.0000    0.9524    0.9756        21
       music     0.9545    0.9545    0.9545        22
        news     0.8696    1.0000    0.9302        20
       novel     1.0000    0.8750    0.9333         8
      poetry     0.9444    1.0000    0.9714        34
       radio     1.0000    0.6250    0.7692         8
      riddle     1.0000    1.0000    1.0000        11
    schedule     0.8889    0.8889    0.8889         9
       stock     0.9091    0.8333    0.8696        24
   telephone     1.0000    0.9524    0.9756        21
       train     0.9583    0.9583    0.9583        24
 translation     1.0000    0.9048    0.9500        21
   tvchannel     0.7273    0.6957    0.7111        23
       video     0.6818    0.7500    0.7143        60
     weather     0.9524    0.9091    0.9302        22
     website     0.7000    0.7778    0.7368        18

    accuracy                         0.8870       770
   macro avg     0.8887    0.8684    0.8748       770
weighted avg     0.8929    0.8870    0.8877       770
```

```
Out[4]: '/Users/enjuichang/Desktop/Minerva/Spring 2020/CS156/Final/CNN'
```

Appendix C. Code for building the BiLSTM model

```
In [6]:
from typing import Dict, Any
from tensorflow import keras
from kashgari.tasks.classification.abc_model import ABCClassificationModel
from kashgari.layers import L

# Create my own BiLSTM model based on the Kashgari package
class DoubleBiLSTMModel(ABCClassificationModel):
    """
    Bidirectional LSTM Sequence Labeling Model
    """

    @classmethod
    def default_hyper_parameters(cls) -> Dict[str, Dict[str, Any]]:
        """
        Get hyper parameters of model
        Returns:
            hyper parameters dict
        """
        return {
            'layer_blstm1': {
                'units': 128,
                'return_sequences': True
            },
            'layer_dropout1': {
                'rate': 0.2
            },
            'layer_blstm2': {
                'units': 128,
                'return_sequences': False
            },
            'layer_dropout2': {
                'rate': 0.2
            },
            'layer_dense': {
                'units': 64,
                'activation': 'relu'
            },
            'layer_time_distributed': {},
            'layer_output': {
            }
        }

    def build_model_arc(self):
        """
        build model architectural
        """
        output_dim = self.label_processor.vocab_size
        config = self.hyper_parameters
        embed_model = self.embedding.embed_model

        # Define your layers
        layer_blstm1 = L.Bidirectional(L.LSTM(**config['layer_blstm1']),
                                       name='layer_blstm1')
        layer_dropout1 = L.Dropout(**config['layer_dropout1'],
                                   name='layer_dropout1')
        layer_blstm2 = L.Bidirectional(L.LSTM(**config['layer_blstm2']),
                                       name='layer_blstm2')
        layer_dropout2 = L.Dropout(**config['layer_dropout2'],
                                   name='layer_dropout2')
        layer_dense = L.Dense(**config['layer_dense'],
                                   name='layer_dense')
        layer_time_distributed = L.Dense(output_dim, **config['layer_output'])

        # You need to use this actiovation layer as final activation
        # to suppor multi-label classification
        layer_activation = self._activation_layer()

        # Define tensor flow
        tensor = layer_blstm1(embed_model.output)
        tensor = layer_blstm2(tensor)
        tensor = layer_dropout2(tensor)
        tensor = layer_dense(tensor)
        tensor = layer_time_distributed(tensor)
        output_tensor = layer_activation(tensor)

        # Init model
        self.tf_model = keras.Model(embed_model.inputs, output_tensor)
```

Appendix D. Code for training the BiLSTM model

In [7]:

```
# BiLSTM model
BiLSTM_model = DoubleBiLSTMModel()

# Build the model using the training data
BiLSTM_model.build_model(train_x, train_y)

# Compile model with custom loss, optimizer and metrics
BiLSTM_model.compile_model(optimizer='adam', metrics = ['accuracy'])

# Train the model
BiLSTM_history = BiLSTM_model.fit(train_x, train_y, valid_x, valid_y, epochs= 20

# Evaluate the model
BiLSTM_model.evaluate(test_x, test_y)

# Save model
BiLSTM_model.save('BiLSTM')
```

```
Preparing text vocab dict: 100%|██████████| 1881/1881 [00:00<00:00, 299706.95it/
s]
2021-04-22 00:44:35,532 [DEBUG] kashgari - --- Build vocab dict finished, Total:
787 ---
2021-04-22 00:44:35,534 [DEBUG] kashgari - Top-10: ['[PAD]', '[UNK]', '[CLS]',
'[SEP]', '的', '么', '我', '。', '怎', '你']
Preparing classification label vocab dict: 100%|██████████| 1881/1881 [00:00<00:
00, 727141.55it/s]
Calculating sequence length: 100%|██████████| 1881/1881 [00:00<00:00, 285706.01i
t/s]
2021-04-22 00:44:35,573 [DEBUG] kashgari - Calculated sequence length = 15
2021-04-22 00:44:38,839 [DEBUG] kashgari - Model: "model_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input (InputLayer)           [(None, None)]            0         
_________________________________________________________________
layer_embedding (Embedding)  (None, None, 100)         78700     
_________________________________________________________________
layer_blstm1 (Bidirectional) (None, None, 256)         234496    
_________________________________________________________________
layer_blstm2 (Bidirectional) (None, 256)               394240    
_________________________________________________________________
layer_dropout2 (Dropout)     (None, 256)               0         
_________________________________________________________________
layer_dense (Dense)          (None, 64)                16448     
_________________________________________________________________
dense_2 (Dense)              (None, 31)                2015      
_________________________________________________________________
activation_1 (Activation)    (None, 31)                0         
=================================================================
Total params: 725,899
Trainable params: 725,899
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
29/29 [==============================] - 21s 239ms/step - loss: 3.2907 - accurac
y: 0.1921 - val_loss: 2.7252 - val_accuracy: 0.2188
Epoch 2/20
29/29 [==============================] - 3s 90ms/step - loss: 2.5571 - accuracy:
0.2863 - val_loss: 2.3019 - val_accuracy: 0.3411
Epoch 3/20
29/29 [==============================] - 2s 76ms/step - loss: 2.1133 - accuracy:
0.3894 - val_loss: 1.9795 - val_accuracy: 0.4401
Epoch 4/20
29/29 [==============================] - 3s 98ms/step - loss: 1.7739 - accuracy:
0.4968 - val_loss: 1.8028 - val_accuracy: 0.5078
Epoch 5/20
29/29 [==============================] - 3s 93ms/step - loss: 1.3035 - accuracy:
0.6411 - val_loss: 1.4274 - val_accuracy: 0.6198
Epoch 6/20
29/29 [==============================] - 2s 78ms/step - loss: 0.9002 - accuracy:
0.7425 - val_loss: 1.2346 - val_accuracy: 0.6823
Epoch 7/20
29/29 [==============================] - 2s 83ms/step - loss: 0.6330 - accuracy:
0.8240 - val_loss: 1.1561 - val_accuracy: 0.7318
Epoch 8/20
29/29 [==============================] - 2s 84ms/step - loss: 0.4679 - accuracy:
0.8674 - val_loss: 1.2344 - val_accuracy: 0.7318
Epoch 9/20
29/29 [==============================] - 2s 80ms/step - loss: 0.3048 - accuracy:
0.9196 - val_loss: 1.1838 - val_accuracy: 0.7526
Epoch 10/20
29/29 [==============================] - 2s 79ms/step - loss: 0.2551 - accuracy:
```

```
0.9255 - val_loss: 1.0742 - val_accuracy: 0.7760
Epoch 11/20
29/29 [==============================] - 2s 81ms/step - loss: 0.2298 - accuracy:
0.9390 - val_loss: 1.1108 - val_accuracy: 0.7552
Epoch 12/20
29/29 [==============================] - 2s 77ms/step - loss: 0.1388 - accuracy:
0.9673 - val_loss: 1.2051 - val_accuracy: 0.7630
Epoch 13/20
29/29 [==============================] - 3s 96ms/step - loss: 0.1095 - accuracy:
0.9772 - val_loss: 1.1242 - val_accuracy: 0.7812
Epoch 14/20
29/29 [==============================] - 3s 93ms/step - loss: 0.0932 - accuracy:
0.9755 - val_loss: 1.2452 - val_accuracy: 0.7656
Epoch 15/20
29/29 [==============================] - 3s 90ms/step - loss: 0.0588 - accuracy:
0.9896 - val_loss: 1.3821 - val_accuracy: 0.7526
Epoch 16/20
29/29 [==============================] - 4s 124ms/step - loss: 0.0398 - accurac
y: 0.9947 - val_loss: 1.3896 - val_accuracy: 0.7578
Epoch 17/20
29/29 [==============================] - 4s 127ms/step - loss: 0.0473 - accurac
y: 0.9908 - val_loss: 1.4192 - val_accuracy: 0.7708
Epoch 18/20
29/29 [==============================] - 3s 120ms/step - loss: 0.0502 - accurac
y: 0.9860 - val_loss: 1.4429 - val_accuracy: 0.7604
Epoch 19/20
29/29 [==============================] - 3s 118ms/step - loss: 0.0852 - accurac
y: 0.9755 - val_loss: 1.3349 - val_accuracy: 0.7839
Epoch 20/20
29/29 [==============================] - 3s 97ms/step - loss: 0.0716 - accuracy:
0.9834 - val_loss: 1.5663 - val_accuracy: 0.7656
2021-04-22 00:45:51,390 [WARNING] kashgari - Sequence length is None, will use t
he max length of the samples, which is 67
2021-04-22 00:45:51,401 [DEBUG] kashgari - predict input shape (770, 67) x:
[[  2  66  20 ...   0   0   0]
 [  2 297   1 ...   0   0   0]
 [  2 374 764 ...   0   0   0]
 ...
 [  2  73 116 ...   0   0   0]
 [  2  66   5 ...   0   0   0]
 [  2 375 169 ...   0   0   0]]
2021-04-22 00:45:57,168 [DEBUG] kashgari - predict output shape (770, 31)
2021-04-22 00:45:57,170 [DEBUG] kashgari - predict output argmax: [ 0  0  1  1
0 13  3 13 14  2  4 10 13 19 21  4  5 10 14  0  2 11 14 14
  0  0  1 18 11  0  0 13  0  1  0  2  2  8  0 16  9 13 17  0  0  2 17  2
  7  0 19  0  0  0  1  6  3 12 17 27 22  1  0  0  1 30  0  6  0  1 11  8
  1 17 11  6 17  0 12  0  1 15  6  4  9 11  0  2 28  0  1  2 12  7  0  2
  2 15  8 27  9  9  3  0  1 15  2  2 19  4  5 12  9 14  6  0 20 16  1  8
  0  5  1  0 10  0 13  9 11 12 17  3  1 25  1  1 11 12  1 17 18  0  7  0
  0  2 14 14  2 19  2  3  1  2 28  7  4 17 30 14  4  0 27  3  8  7 20 16
 13 11 27 10 24  0 15  7 15  2 17  2  1 15  8  1  3  0  6  1 21  2  6 14
 12 19  5 19  1  0  0  3 13 17 19  0  9  3 30  9  0 15 14  0  0  6 16  3
  0  1  1 17  8  4  0  0 14 30  0  1  1  2 29 11 14 12  5  5 10 16  0  9
 16  6  4  6 10  9 11  0  8 15  0  0  4  1  4  1  2 14  7  1 11  1 25 16
 12  5 17 27  0  3 17 19 12  3  9 17  1  4 13 15  1 16  0  1  1 17  6  0
 11 15 22  3  4  0 23  1 17  0  0  2  2  1 21  0  0 11 12  2  3  0  0 16
  1 24  6 19 18 17  0 17  9  6  1 10  5 18  0  4  7  4 12  0 15  0  1  8
 24  1  3  7 14  2  7  1  0  4 27 17 13 17  1  2 18  7  4  2  5  1 16 28
  0 13  0 16  1  6 10 17 17 20  0 11  5 25  0  0  0  0 20  4  1 15 17  0
 12  0 20  1  0  3  0 21 30  1  8 27 30  6 14  3 11  0 24 20  1  7 23  0
 23 13 10  7  7 26 21  9  1  5  5  0 10 30  8  3  0 17  6  0  0 12  0 17
 19  9 15 18  0  4 11  0  1  0  8 10  4 10 29 12 11 17  0  8  0  4  9 20
  6 11 11  0 10  6  2  1 11  1 22  1 17  2  0  8  0  6 16 29 17  6  4  1
  0  0  4  9  7  5  1  2 12  3  9 10 11  0 24 17 30  5  6  5  0  8 10  8
 12  0  9  5  0 10 21 18 19 13 12  1 24  0  0  9  4  1  1 17 11  7  0  5
 10 10  7 30  0  2  1  7  9  0 17  2  0  2  0 16  1  1  4  1  8  2 13  2
 14  9 12 22  7 21  4  1  1 15  1 24  0 28 12  0  4 22  1  4 11 23 14  3
  4  0  2  0  0 10 15  0  2 18 16  0 17  0  1  0  7  2  1  5 19  1 16  0
 16  0  0  0  6  0  2 15  0 20 15  5  1 15  1 28 10  7  5  2 24 22 10 12
  0  2  5  1  0  2  0 29  3 15  2 12 10  8 25  2  0  0  7 14 21  0  1  5
 30  8 15  0  2  0  6  3  5  2  6 24 11 17 16 18 14 15 14  0  9 11  0  4
  0 10  5  5  3  1 14 13 14  8  9  4  1  0  3  1 15 17 16  3 15  0 13  0
 13 21 19  2  0  1  8  0 10 19  0 15  1  5  8  6  3  9  4  1  2  3  5  8
 30  2  2 28  2  1  0 26 14  9  1  2  0 30  0  0 16 18  3  0  4  4  0 28
  0  1  1 28  6  2 27  0  3  0  6  0  6 25  1  0  4 22  2  0  3 15 11  6
  0 16]
2021-04-22 00:45:57,236 [INFO] kashgari - model saved to /Users/enjuichang/Deskt
op/Minerva/Spring 2020/CS156/Final/BiLSTM
              precision    recall  f1-score   support

         app     0.4118    0.7778    0.5385        18
         bus     0.7778    0.8750    0.8235         8
        calc     0.5000    0.1250    0.2000         8
        chat     0.8600    0.8377    0.8487       154
     cinemas     0.7500    0.7500    0.7500         8
    contacts     0.7500    0.6000    0.6667        10
    cookbook     0.9765    0.9326    0.9540        89
    datetime     0.5000    0.3333    0.4000         6
       email     1.0000    0.8750    0.9333         8
         epg     0.7647    0.7222    0.7429        36
      flight     0.7826    0.8571    0.8182        21
      health     0.6000    0.3158    0.4138        19
     lottery     0.7500    0.7500    0.7500         8
         map     0.5600    0.6087    0.5833        23
       match     1.0000    0.6250    0.7692         8
     message     0.8182    0.8571    0.8372        21
       music     0.6207    0.8182    0.7059        22
        news     0.8000    1.0000    0.8889        20
       novel     0.5000    0.2500    0.3333         8
      poetry     0.8667    0.7647    0.8125        34
       radio     0.5000    0.7500    0.6000         8
      riddle     0.7857    1.0000    0.8800        11
    schedule     0.7778    0.7778    0.7778         9
       stock     0.7826    0.7500    0.7660        24
   telephone     0.7000    0.6667    0.6829        21
       train     0.8750    0.8750    0.8750        24
 translation     0.8636    0.9048    0.8837        21
   tvchannel     0.5556    0.6522    0.6000        23
       video     0.6111    0.5500    0.5789        60
     weather     0.6923    0.8182    0.7500        22
     website     0.7059    0.6667    0.6857        18

    accuracy                         0.7636       770
   macro avg     0.7238    0.7125    0.7048       770
weighted avg     0.7710    0.7636    0.7606       770
```

Out[7]:
```
'/Users/enjuichang/Desktop/Minerva/Spring 2020/CS156/Final/BiLSTM'
```

Appendix E. Code for training the BERT model

In [9]:
```
# BERT Model
bert_embed = BertEmbedding('./BERT')
bert_model = BiGRU_Model(bert_embed, sequence_length=100)

# Build the model using the training data
bert_model.build_model(train_x, train_y)
```

```python
# Compile model with custom loss, optimizer and metrics
bert_model.compile_model(optimizer='adam', metrics = ['accuracy'])

# Train the model
bert_model_history = bert_model.fit(train_x, train_y, valid_x, valid_y, epochs=

# Save model
bert_model.save('bert_model')
```

```
2021-04-22 00:45:57,658 [DEBUG] kashgari - ---------------------------------------
-----------
2021-04-22 00:45:57,659 [DEBUG] kashgari - Loaded transformer model's vocab
2021-04-22 00:45:57,660 [DEBUG] kashgari - config_path       : ./BERT/bert_confi
g.json
2021-04-22 00:45:57,661 [DEBUG] kashgari - vocab_path      : ./BERT/vocab.txt
2021-04-22 00:45:57,662 [DEBUG] kashgari - checkpoint_path : ./BERT/bert_model.c
kpt
2021-04-22 00:45:57,663 [DEBUG] kashgari - Top 50 words    : ['[PAD]', '[unused
1]', '[unused2]', '[unused3]', '[unused4]', '[unused5]', '[unused6]', '[unused
7]', '[unused8]', '[unused9]', '[unused10]', '[unused11]', '[unused12]', '[unuse
d13]', '[unused14]', '[unused15]', '[unused16]', '[unused17]', '[unused18]', '[u
nused19]', '[unused20]', '[unused21]', '[unused22]', '[unused23]', '[unused24]',
'[unused25]', '[unused26]', '[unused27]', '[unused28]', '[unused29]', '[unused3
0]', '[unused31]', '[unused32]', '[unused33]', '[unused34]', '[unused35]', '[unu
sed36]', '[unused37]', '[unused38]', '[unused39]', '[unused40]', '[unused41]',
'[unused42]', '[unused43]', '[unused44]', '[unused45]', '[unused46]', '[unused4
7]', '[unused48]', '[unused49]']
2021-04-22 00:45:57,663 [DEBUG] kashgari - ---------------------------------------
-----------
Preparing text vocab dict: 100%|██████████| 1881/1881 [00:00<00:00, 254335.46it/
s]
2021-04-22 00:45:57,675 [DEBUG] kashgari - --- Build vocab dict finished, Total:
787 ---
2021-04-22 00:45:57,677 [DEBUG] kashgari - Top-10: ['[PAD]', '[UNK]', '[CLS]',
'[SEP]', '的', '么', '我', '。', '怎', '你']
Preparing classification label vocab dict: 100%|██████████| 1881/1881 [00:00<00:
00, 886060.85it/s]
2021-04-22 00:46:05,381 [DEBUG] kashgari - Model: "model_5"
________________________________________________________________________________
__________________
Layer (type)                     Output Shape          Param #     Connected to
================================================================================
==================
Input-Token (InputLayer)         [(None, None)]        0

________________________________________________________________________________
__________________
Input-Segment (InputLayer)       [(None, None)]        0

________________________________________________________________________________
__________________
Embedding-Token (Embedding)      (None, None, 768)     16226304    Input-Token[0]
[0]

________________________________________________________________________________
__________________
Embedding-Segment (Embedding)    (None, None, 768)     1536        Input-Segment
[0][0]

________________________________________________________________________________
__________________
Embedding-Token-Segment (Add)    (None, None, 768)     0           Embedding-Token
[0][0]
                                                                   Embedding-Segme
nt[0][0]

________________________________________________________________________________
__________________
Embedding-Position (PositionEmb (None, None, 768)     393216      Embedding-Token
-Segment[0][0]
__________________________________________________________________________________
__________________
Embedding-Norm (LayerNormalizat (None, None, 768)    1536        Embedding-Posit
ion[0][0]
__________________________________________________________________________________
__________________
Embedding-Dropout (Dropout)     (None, None, 768)    0           Embedding-Norm
[0][0]
__________________________________________________________________________________
__________________
Transformer-0-MultiHeadSelfAtte (None, None, 768)    2362368     Embedding-Dropo
ut[0][0]
                                                                 Embedding-Dropo
ut[0][0]
                                                                 Embedding-Dropo
ut[0][0]
__________________________________________________________________________________
__________________
Transformer-0-MultiHeadSelfAtte (None, None, 768)    0           Transformer-0-M
ultiHeadSelfAttent
__________________________________________________________________________________
__________________
Transformer-0-MultiHeadSelfAtte (None, None, 768)    0           Embedding-Dropo
ut[0][0]
                                                                 Transformer-0-M
ultiHeadSelfAttent
__________________________________________________________________________________
__________________
Transformer-0-MultiHeadSelfAtte (None, None, 768)    1536        Transformer-0-M
ultiHeadSelfAttent
__________________________________________________________________________________
__________________
Transformer-0-FeedForward (Feed (None, None, 768)    4722432     Transformer-0-M
ultiHeadSelfAttent
__________________________________________________________________________________
__________________
Transformer-0-FeedForward-Dropo (None, None, 768)    0           Transformer-0-F
eedForward[0][0]
__________________________________________________________________________________
__________________
Transformer-0-FeedForward-Add ( (None, None, 768)    0           Transformer-0-M
ultiHeadSelfAttent
                                                                 Transformer-0-F
eedForward-Dropout
__________________________________________________________________________________
__________________
Transformer-0-FeedForward-Norm  (None, None, 768)    1536        Transformer-0-F
eedForward-Add[0][
__________________________________________________________________________________
__________________
Transformer-1-MultiHeadSelfAtte (None, None, 768)    2362368     Transformer-0-F
eedForward-Norm[0]
                                                                 Transformer-0-F
eedForward-Norm[0]
                                                                 Transformer-0-F
eedForward-Norm[0]
__________________________________________________________________________________
__________________
Transformer-1-MultiHeadSelfAtte (None, None, 768)    0           Transformer-1-M
ultiHeadSelfAttent
__________________________________________________________________________________
__________________
Transformer-1-MultiHeadSelfAtte (None, None, 768)    0           Transformer-0-F
eedForward-Norm[0]
```

```
                                                                 Transformer-1-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-1-MultiHeadSelfAtte (None, None, 768)    1536        Transformer-1-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-1-FeedForward (Feed (None, None, 768)    4722432     Transformer-1-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-1-FeedForward-Dropo (None, None, 768)    0           Transformer-1-F
eedForward[0][0]
________________________________________________________________________________
__________________
Transformer-1-FeedForward-Add ( (None, None, 768)    0           Transformer-1-M
ultiHeadSelfAttent
                                                                 Transformer-1-F
eedForward-Dropout
________________________________________________________________________________
__________________
Transformer-1-FeedForward-Norm  (None, None, 768)    1536        Transformer-1-F
eedForward-Add[0][
________________________________________________________________________________
__________________
Transformer-2-MultiHeadSelfAtte (None, None, 768)    2362368     Transformer-1-F
eedForward-Norm[0]
                                                                 Transformer-1-F
eedForward-Norm[0]
                                                                 Transformer-1-F
eedForward-Norm[0]
________________________________________________________________________________
__________________
Transformer-2-MultiHeadSelfAtte (None, None, 768)    0           Transformer-2-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-2-MultiHeadSelfAtte (None, None, 768)    0           Transformer-1-F
eedForward-Norm[0]
                                                                 Transformer-2-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-2-MultiHeadSelfAtte (None, None, 768)    1536        Transformer-2-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-2-FeedForward (Feed (None, None, 768)    4722432     Transformer-2-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-2-FeedForward-Dropo (None, None, 768)    0           Transformer-2-F
eedForward[0][0]
________________________________________________________________________________
__________________
Transformer-2-FeedForward-Add ( (None, None, 768)    0           Transformer-2-M
ultiHeadSelfAttent
                                                                 Transformer-2-F
eedForward-Dropout
________________________________________________________________________________
__________________
Transformer-2-FeedForward-Norm  (None, None, 768)    1536        Transformer-2-F
eedForward-Add[0][
________________________________________________________________________________
```

```
__________________
Transformer-3-MultiHeadSelfAtte (None, None, 768)    2362368     Transformer-2-F
eedForward-Norm[0]
                                                                 Transformer-2-F
eedForward-Norm[0]
                                                                 Transformer-2-F
eedForward-Norm[0]
________________________________________________________________________________
__________________
Transformer-3-MultiHeadSelfAtte (None, None, 768)    0           Transformer-3-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-3-MultiHeadSelfAtte (None, None, 768)    0           Transformer-2-F
eedForward-Norm[0]
                                                                 Transformer-3-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-3-MultiHeadSelfAtte (None, None, 768)    1536        Transformer-3-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-3-FeedForward (Feed (None, None, 768)    4722432     Transformer-3-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-3-FeedForward-Dropo (None, None, 768)    0           Transformer-3-F
eedForward[0][0]
________________________________________________________________________________
__________________
Transformer-3-FeedForward-Add ( (None, None, 768)    0           Transformer-3-M
ultiHeadSelfAttent
                                                                 Transformer-3-F
eedForward-Dropout
________________________________________________________________________________
__________________
Transformer-3-FeedForward-Norm  (None, None, 768)    1536        Transformer-3-F
eedForward-Add[0][
________________________________________________________________________________
__________________
Transformer-4-MultiHeadSelfAtte (None, None, 768)    2362368     Transformer-3-F
eedForward-Norm[0]
                                                                 Transformer-3-F
eedForward-Norm[0]
                                                                 Transformer-3-F
eedForward-Norm[0]
________________________________________________________________________________
__________________
Transformer-4-MultiHeadSelfAtte (None, None, 768)    0           Transformer-4-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-4-MultiHeadSelfAtte (None, None, 768)    0           Transformer-3-F
eedForward-Norm[0]
                                                                 Transformer-4-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-4-MultiHeadSelfAtte (None, None, 768)    1536        Transformer-4-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-4-FeedForward (Feed (None, None, 768)    4722432     Transformer-4-M
ultiHeadSelfAttent
```

```
__________________________________________________________________________________________________
Transformer-4-FeedForward-Dropo (None, None, 768)    0           Transformer-4-F
eedForward[0][0]
__________________________________________________________________________________________________
Transformer-4-FeedForward-Add ( (None, None, 768)    0           Transformer-4-M
ultiHeadSelfAttent
                                                                 Transformer-4-F
eedForward-Dropout
__________________________________________________________________________________________________
Transformer-4-FeedForward-Norm  (None, None, 768)    1536        Transformer-4-F
eedForward-Add[0][
__________________________________________________________________________________________________
Transformer-5-MultiHeadSelfAtte (None, None, 768)    2362368     Transformer-4-F
eedForward-Norm[0]
                                                                 Transformer-4-F
eedForward-Norm[0]
                                                                 Transformer-4-F
eedForward-Norm[0]
__________________________________________________________________________________________________
Transformer-5-MultiHeadSelfAtte (None, None, 768)    0           Transformer-5-M
ultiHeadSelfAttent
__________________________________________________________________________________________________
Transformer-5-MultiHeadSelfAtte (None, None, 768)    0           Transformer-4-F
eedForward-Norm[0]
                                                                 Transformer-5-M
ultiHeadSelfAttent
__________________________________________________________________________________________________
Transformer-5-MultiHeadSelfAtte (None, None, 768)    1536        Transformer-5-M
ultiHeadSelfAttent
__________________________________________________________________________________________________
Transformer-5-FeedForward (Feed (None, None, 768)    4722432     Transformer-5-M
ultiHeadSelfAttent
__________________________________________________________________________________________________
Transformer-5-FeedForward-Dropo (None, None, 768)    0           Transformer-5-F
eedForward[0][0]
__________________________________________________________________________________________________
Transformer-5-FeedForward-Add ( (None, None, 768)    0           Transformer-5-M
ultiHeadSelfAttent
                                                                 Transformer-5-F
eedForward-Dropout
__________________________________________________________________________________________________
Transformer-5-FeedForward-Norm  (None, None, 768)    1536        Transformer-5-F
eedForward-Add[0][
__________________________________________________________________________________________________
Transformer-6-MultiHeadSelfAtte (None, None, 768)    2362368     Transformer-5-F
eedForward-Norm[0]
                                                                 Transformer-5-F
eedForward-Norm[0]
                                                                 Transformer-5-F
eedForward-Norm[0]
__________________________________________________________________________________________________
Transformer-6-MultiHeadSelfAtte (None, None, 768)    0           Transformer-6-M
```

```
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-6-MultiHeadSelfAtte (None, None, 768)    0           Transformer-5-F
eedForward-Norm[0]
                                                                 Transformer-6-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-6-MultiHeadSelfAtte (None, None, 768)    1536        Transformer-6-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-6-FeedForward (Feed (None, None, 768)    4722432     Transformer-6-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-6-FeedForward-Dropo (None, None, 768)    0           Transformer-6-F
eedForward[0][0]
________________________________________________________________________________
__________________
Transformer-6-FeedForward-Add ( (None, None, 768)    0           Transformer-6-M
ultiHeadSelfAttent
                                                                 Transformer-6-F
eedForward-Dropout
________________________________________________________________________________
__________________
Transformer-6-FeedForward-Norm  (None, None, 768)    1536        Transformer-6-F
eedForward-Add[0][
________________________________________________________________________________
__________________
Transformer-7-MultiHeadSelfAtte (None, None, 768)    2362368     Transformer-6-F
eedForward-Norm[0]
                                                                 Transformer-6-F
eedForward-Norm[0]
                                                                 Transformer-6-F
eedForward-Norm[0]
________________________________________________________________________________
__________________
Transformer-7-MultiHeadSelfAtte (None, None, 768)    0           Transformer-7-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-7-MultiHeadSelfAtte (None, None, 768)    0           Transformer-6-F
eedForward-Norm[0]
                                                                 Transformer-7-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-7-MultiHeadSelfAtte (None, None, 768)    1536        Transformer-7-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-7-FeedForward (Feed (None, None, 768)    4722432     Transformer-7-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-7-FeedForward-Dropo (None, None, 768)    0           Transformer-7-F
eedForward[0][0]
________________________________________________________________________________
__________________
Transformer-7-FeedForward-Add ( (None, None, 768)    0           Transformer-7-M
ultiHeadSelfAttent
                                                                 Transformer-7-F
eedForward-Dropout
```

```
________________________________________________________________________________
__________________
Transformer-7-FeedForward-Norm  (None, None, 768)    1536        Transformer-7-F
eedForward-Add[0][
________________________________________________________________________________
__________________
Transformer-8-MultiHeadSelfAtte (None, None, 768)    2362368     Transformer-7-F
eedForward-Norm[0]
                                                                 Transformer-7-F
eedForward-Norm[0]
                                                                 Transformer-7-F
eedForward-Norm[0]
________________________________________________________________________________
__________________
Transformer-8-MultiHeadSelfAtte (None, None, 768)    0           Transformer-8-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-8-MultiHeadSelfAtte (None, None, 768)    0           Transformer-7-F
eedForward-Norm[0]
                                                                 Transformer-8-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-8-MultiHeadSelfAtte (None, None, 768)    1536        Transformer-8-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-8-FeedForward (Feed (None, None, 768)    4722432     Transformer-8-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-8-FeedForward-Dropo (None, None, 768)    0           Transformer-8-F
eedForward[0][0]
________________________________________________________________________________
__________________
Transformer-8-FeedForward-Add ( (None, None, 768)    0           Transformer-8-M
ultiHeadSelfAttent
                                                                 Transformer-8-F
eedForward-Dropout
________________________________________________________________________________
__________________
Transformer-8-FeedForward-Norm  (None, None, 768)    1536        Transformer-8-F
eedForward-Add[0][
________________________________________________________________________________
__________________
Transformer-9-MultiHeadSelfAtte (None, None, 768)    2362368     Transformer-8-F
eedForward-Norm[0]
                                                                 Transformer-8-F
eedForward-Norm[0]
                                                                 Transformer-8-F
eedForward-Norm[0]
________________________________________________________________________________
__________________
Transformer-9-MultiHeadSelfAtte (None, None, 768)    0           Transformer-9-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-9-MultiHeadSelfAtte (None, None, 768)    0           Transformer-8-F
eedForward-Norm[0]
                                                                 Transformer-9-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-9-MultiHeadSelfAtte (None, None, 768)    1536        Transformer-9-M
```

```
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-9-FeedForward (Feed (None, None, 768)    4722432     Transformer-9-M
ultiHeadSelfAttent
________________________________________________________________________________
__________________
Transformer-9-FeedForward-Dropo (None, None, 768)    0           Transformer-9-F
eedForward[0][0]
________________________________________________________________________________
__________________
Transformer-9-FeedForward-Add ( (None, None, 768)    0           Transformer-9-M
ultiHeadSelfAttent
                                                                 Transformer-9-F
eedForward-Dropout
________________________________________________________________________________
__________________
Transformer-9-FeedForward-Norm  (None, None, 768)    1536        Transformer-9-F
eedForward-Add[0][
________________________________________________________________________________
__________________
Transformer-10-MultiHeadSelfAtt (None, None, 768)    2362368     Transformer-9-F
eedForward-Norm[0]
                                                                 Transformer-9-F
eedForward-Norm[0]
                                                                 Transformer-9-F
eedForward-Norm[0]
________________________________________________________________________________
__________________
Transformer-10-MultiHeadSelfAtt (None, None, 768)    0           Transformer-10-
MultiHeadSelfAtten
________________________________________________________________________________
__________________
Transformer-10-MultiHeadSelfAtt (None, None, 768)    0           Transformer-9-F
eedForward-Norm[0]
                                                                 Transformer-10-
MultiHeadSelfAtten
________________________________________________________________________________
__________________
Transformer-10-MultiHeadSelfAtt (None, None, 768)    1536        Transformer-10-
MultiHeadSelfAtten
________________________________________________________________________________
__________________
Transformer-10-FeedForward (Fee (None, None, 768)    4722432     Transformer-10-
MultiHeadSelfAtten
________________________________________________________________________________
__________________
Transformer-10-FeedForward-Drop (None, None, 768)    0           Transformer-10-
FeedForward[0][0]
________________________________________________________________________________
__________________
Transformer-10-FeedForward-Add  (None, None, 768)    0           Transformer-10-
MultiHeadSelfAtten
                                                                 Transformer-10-
FeedForward-Dropou
________________________________________________________________________________
__________________
Transformer-10-FeedForward-Norm (None, None, 768)    1536        Transformer-10-
FeedForward-Add[0]
________________________________________________________________________________
__________________
Transformer-11-MultiHeadSelfAtt (None, None, 768)    2362368     Transformer-10-
FeedForward-Norm[0
                                                                 Transformer-10-
FeedForward-Norm[0
```

```
                                                                 Transformer-10-
FeedForward-Norm[0
__________________________________________________________________________________
__________________
Transformer-11-MultiHeadSelfAtt (None, None, 768)    0           Transformer-11-
MultiHeadSelfAtten
__________________________________________________________________________________
__________________
Transformer-11-MultiHeadSelfAtt (None, None, 768)    0           Transformer-10-
FeedForward-Norm[0
                                                                 Transformer-11-
MultiHeadSelfAtten
__________________________________________________________________________________
__________________
Transformer-11-MultiHeadSelfAtt (None, None, 768)    1536        Transformer-11-
MultiHeadSelfAtten
__________________________________________________________________________________
__________________
Transformer-11-FeedForward (Fee (None, None, 768)    4722432     Transformer-11-
MultiHeadSelfAtten
__________________________________________________________________________________
__________________
Transformer-11-FeedForward-Drop (None, None, 768)    0           Transformer-11-
FeedForward[0][0]
__________________________________________________________________________________
__________________
Transformer-11-FeedForward-Add  (None, None, 768)    0           Transformer-11-
MultiHeadSelfAtten
                                                                 Transformer-11-
FeedForward-Dropou
__________________________________________________________________________________
__________________
Transformer-11-FeedForward-Norm (None, None, 768)    1536        Transformer-11-
FeedForward-Add[0]
__________________________________________________________________________________
__________________
bidirectional (Bidirectional)   (None, 256)          689664      Transformer-11-
FeedForward-Norm[0
__________________________________________________________________________________
__________________
dense_3 (Dense)                 (None, 31)           7967        bidirectional
[0][0]
__________________________________________________________________________________
__________________
activation_2 (Activation)       (None, 31)           0           dense_3[0][0]
==================================================================================
==================
Total params: 102,374,687
Trainable params: 697,631
Non-trainable params: 101,677,056
__________________________________________________________________________________
__________________
Epoch 1/8
29/29 [==============================] - 475s 16s/step - loss: 2.1958 - accurac
y: 0.4512 - val_loss: 0.6283 - val_accuracy: 0.8359
Epoch 2/8
29/29 [==============================] - 417s 15s/step - loss: 0.5559 - accurac
y: 0.8640 - val_loss: 0.3582 - val_accuracy: 0.8906
Epoch 3/8
29/29 [==============================] - 413s 14s/step - loss: 0.2481 - accurac
y: 0.9285 - val_loss: 0.2847 - val_accuracy: 0.9167
Epoch 4/8
29/29 [==============================] - 492s 17s/step - loss: 0.1762 - accurac
y: 0.9611 - val_loss: 0.2560 - val_accuracy: 0.9297
Epoch 5/8
```

```
29/29 [==============================] - 492s 17s/step - loss: 0.1118 - accurac
y: 0.9810 - val_loss: 0.2529 - val_accuracy: 0.9141
Epoch 6/8
29/29 [==============================] - 506s 18s/step - loss: 0.0884 - accurac
y: 0.9868 - val_loss: 0.2489 - val_accuracy: 0.9062
Epoch 7/8
29/29 [==============================] - 491s 17s/step - loss: 0.0560 - accurac
y: 0.9948 - val_loss: 0.2510 - val_accuracy: 0.9089
Epoch 8/8
29/29 [==============================] - 762s 26s/step - loss: 0.0379 - accurac
y: 0.9959 - val_loss: 0.2432 - val_accuracy: 0.9141
2021-04-22 01:53:38,169 [INFO] kashgari - model saved to /Users/enjuichang/Deskt
op/Minerva/Spring 2020/CS156/Final/bert_model
```

```
Out[9]: '/Users/enjuichang/Desktop/Minerva/Spring 2020/CS156/Final/bert_model'
```

Appendix F. Code for generating the unlabeled news data

```
In [42]:
from glob import glob
import tqdm
import jieba
import numpy as np

# Set the file path
sports = glob('./News data/THUCNews/体育/*')
entertain = glob('./News data/THUCNews/娱乐/*')
land = glob('./News data/THUCNews/房产/*')
edu = glob('./News data/THUCNews/教育/*')
tech = glob('./News data/THUCNews/科技/*')

files = [sports, entertain, land, edu, tech]

storage = []

# Use preprocessing techniques to scrape and label the data
for label in range(len(files)):
    # Open the files
    text = files[label][np.random.randint(len(files[label]))]
    lines = open(text, 'r', encoding='utf-8').read().splitlines()

    # Run through lines and use jieba to perform word separation
    ls = []
    for line in tqdm.tqdm(lines):
        rows = line.split('\t')
        ls.append(list(jieba.cut('\t'.join(rows))))

    # Concatenate the WS-transformd data and label the news
    data = []
    ls = np.concatenate(np.asarray(ls))
    for i in range(len(ls)):
        if ls[i] != '\u3000' and ls[i] != '\xa0':
            data.append(ls[i])
    storage.append([data,label])
```

```
100%|██████████| 17/17 [00:00<00:00, 715.67it/s]
100%|██████████| 8/8 [00:00<00:00, 260.32it/s]
100%|██████████| 13/13 [00:00<00:00, 422.57it/s]
100%|██████████| 7/7 [00:00<00:00, 924.20it/s]
100%|██████████| 7/7 [00:00<00:00, 673.55it/s]
```