1. 資料前處理

In [6]:
```
import pandas as pd
import numpy as np

df = pd.read_csv("./yelp.csv")
df = df[["stars","text"]]
df['stars'] = df['stars'].map(lambda x : 1 if x >=4 else 0)

from keras.datasets import imdb
from keras.preprocessing import sequence
from keras.preprocessing.text import Tokenizer
import numpy as np

split_ratio = 0.8
split_point = int(split_ratio*len(df["text"]))
train_text = df["text"].tolist()[:split_point]
y_train = np.array(df["stars"].tolist()[:split_point])
test_text = df["text"].tolist()[split_point:]
y_test = np.array(df["stars"].tolist()[split_point:])

token = Tokenizer(num_words=10000)
token.fit_on_texts(train_text)

token.word_index #可以看到它將英文字轉為數字的結果，例如:the轉換成1

x_train_seq = token.texts_to_sequences(train_text)
x_test_seq = token.texts_to_sequences(test_text)

x_train = sequence.pad_sequences(x_train_seq, maxlen=2000)
x_test = sequence.pad_sequences(x_test_seq, maxlen=2000)
```

2.1 CNN 建模

a. 用CNN對train的資料進行建模，可自行設計神經網路的架構

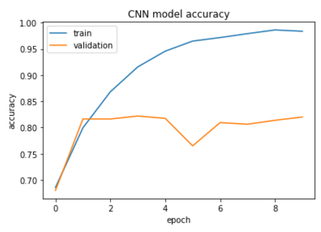
b. 加入Dropout Layer設定Dropout參數(建議0.7)

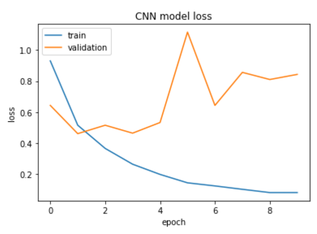
In [7]:
```
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.layers import Embedding
from keras.layers import Conv1D, MaxPooling1D, Flatten
from keras.regularizers import l2
import matplotlib.pyplot as plt

modelCNN = Sequential()
modelCNN.add(Embedding(output_dim=300, input_dim=10000, input_length=2000))
modelCNN.add(Conv1D(32, 3, activation='relu', padding="same", kernel_regularizer=l2(0.01), bias_regularizer=l2(0.01)))
modelCNN.add(MaxPooling1D(3))
modelCNN.add(Conv1D(32, 3, activation='relu', padding="same", kernel_regularizer=l2(0.01), bias_regularizer=l2(0.01)))
modelCNN.add(MaxPooling1D(3))
modelCNN.add(Flatten()) #需要
modelCNN.add(Dropout(0.7))
modelCNN.add(Dense(128, activation='relu'))
modelCNN.add(Dense(1, activation='sigmoid'))

print(modelCNN.summary())
modelCNN.compile(loss='binary_crossentropy',optimizer='Nadam',metrics=['accuracy'])
history = modelCNN.fit(x_train, y_train, epochs = 10, batch_size = 50, verbose = 1, validation_split = 0.2)
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 2000, 300)         3000000

 conv1d_2 (Conv1D)           (None, 2000, 32)          28832

 max_pooling1d_2 (MaxPooling  (None, 666, 32)          0
 1D)

 conv1d_3 (Conv1D)           (None, 666, 32)           3104

 max_pooling1d_3 (MaxPooling  (None, 222, 32)          0
 1D)

 flatten_1 (Flatten)         (None, 7104)              0

 dropout_3 (Dropout)         (None, 7104)              0

 dense_4 (Dense)             (None, 128)               909440

 dense_5 (Dense)             (None, 1)                 129

=================================================================
Total params: 3,941,505
Trainable params: 3,941,505
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/10
128/128 [==============================] - 8s 48ms/step - loss: 0.9306 - accuracy: 0.6856 - val_loss: 0.6438 - val_accuracy: 0.6800
Epoch 2/10
128/128 [==============================] - 6s 46ms/step - loss: 0.5161 - accuracy: 0.7994 - val_loss: 0.4618 - val_accuracy: 0.8163
Epoch 3/10
128/128 [==============================] - 6s 46ms/step - loss: 0.3667 - accuracy: 0.8681 - val_loss: 0.5161 - val_accuracy: 0.8163
Epoch 4/10
128/128 [==============================] - 6s 44ms/step - loss: 0.2650 - accuracy: 0.9156 - val_loss: 0.4650 - val_accuracy: 0.8219
Epoch 5/10
128/128 [==============================] - 6s 43ms/step - loss: 0.1993 - accuracy: 0.9456 - val_loss: 0.5336 - val_accuracy: 0.8175
Epoch 6/10
128/128 [==============================] - 6s 46ms/step - loss: 0.1451 - accuracy: 0.9648 - val_loss: 1.1158 - val_accuracy: 0.7650
Epoch 7/10
128/128 [==============================] - 6s 46ms/step - loss: 0.1251 - accuracy: 0.9716 - val_loss: 0.6437 - val_accuracy: 0.8094
Epoch 8/10
128/128 [==============================] - 6s 43ms/step - loss: 0.1034 - accuracy: 0.9791 - val_loss: 0.8566 - val_accuracy: 0.8062
Epoch 9/10
128/128 [==============================] - 6s 46ms/step - loss: 0.0823 - accuracy: 0.9861 - val_loss: 0.8105 - val_accuracy: 0.8138
Epoch 10/10
128/128 [==============================] - 6s 46ms/step - loss: 0.0824 - accuracy: 0.9836 - val_loss: 0.8435 - val_accuracy: 0.8200
```

c. plot出CNN訓練過程中的Accuracy與Loss值變化

In [8]:
```
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('CNN model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
#file_name = '/content/drive/MyDrive/DM HW4 picture/'+'CNN'+'accuracy.png'
#plt.savefig(file_name)
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('CNN model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
#file_name = '/content/drive/MyDrive/DM HW4 picture/'+'CNN'+'loss.png'
#plt.savefig(file_name)
plt.show()
```





2.2 LSTM 建模

a. 用LSTM 對train的資料進行建模，可自行設計神經網路的架構

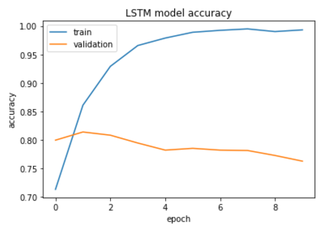
b. 加入Dropout Layer設定Dropout參數(建議0.7)

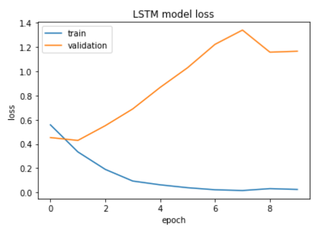
In [9]:
```
from keras.models import Sequential
from keras.layers.core import Dense,Dropout,Activation,Flatten
from keras.layers import CuDNNLSTM
from keras.layers.embeddings import Embedding
from keras.layers.recurrent import LSTM
from tensorflow import keras
from keras import optimizers
from tensorflow.keras import layers

modelLSTM = Sequential()
modelLSTM.add(Embedding(output_dim=300, input_dim=10000, input_length=2000))
modelLSTM.add(Dropout(0.2))
modelLSTM.add(CuDNNLSTM(32))
modelLSTM.add(Dense(units=256,activation='relu'))
modelLSTM.add(Dropout(0.2))
modelLSTM.add(Dense(units=1,activation='sigmoid'))

modelLSTM.summary()
adam = keras.optimizers.Adam() #default 0.001
modelLSTM.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'])
history = modelLSTM.fit(x_train, y_train, epochs = 10, batch_size = 50, verbose = 1, validation_split = 0.2)
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 2000, 300)         3000000

 dropout_4 (Dropout)         (None, 2000, 300)         0

 cu_dnnlstm_1 (CuDNNLSTM)    (None, 32)                42752

 dense_6 (Dense)             (None, 256)               8448

 dropout_5 (Dropout)         (None, 256)               0

 dense_7 (Dense)             (None, 1)                 257

=================================================================
Total params: 3,051,457
Trainable params: 3,051,457
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
128/128 [==============================] - 30s 227ms/step - loss: 0.5584 - accuracy: 0.7130 - val_loss: 0.4533 - val_accuracy: 0.7994
Epoch 2/10
128/128 [==============================] - 29s 224ms/step - loss: 0.3352 - accuracy: 0.8608 - val_loss: 0.4296 - val_accuracy: 0.8138
Epoch 3/10
128/128 [==============================] - 29s 225ms/step - loss: 0.1908 - accuracy: 0.9291 - val_loss: 0.5515 - val_accuracy: 0.8081
Epoch 4/10
128/128 [==============================] - 29s 225ms/step - loss: 0.0947 - accuracy: 0.9656 - val_loss: 0.6895 - val_accuracy: 0.7944
Epoch 5/10
128/128 [==============================] - 29s 225ms/step - loss: 0.0635 - accuracy: 0.9787 - val_loss: 0.8673 - val_accuracy: 0.7819
Epoch 6/10
128/128 [==============================] - 29s 223ms/step - loss: 0.0396 - accuracy: 0.9889 - val_loss: 1.0296 - val_accuracy: 0.7850
Epoch 7/10
128/128 [==============================] - 29s 224ms/step - loss: 0.0227 - accuracy: 0.9923 - val_loss: 1.2225 - val_accuracy: 0.7819
Epoch 8/10
128/128 [==============================] - 28s 223ms/step - loss: 0.0164 - accuracy: 0.9948 - val_loss: 1.3400 - val_accuracy: 0.7812
Epoch 9/10
128/128 [==============================] - 29s 225ms/step - loss: 0.0320 - accuracy: 0.9902 - val_loss: 1.1576 - val_accuracy: 0.7725
Epoch 10/10
128/128 [==============================] - 29s 223ms/step - loss: 0.0254 - accuracy: 0.9931 - val_loss: 1.1655 - val_accuracy: 0.7625
```

c. plot出LSTM訓練過程中的Accuracy與Loss值變化

In [10]:
```
import matplotlib.pyplot as plt
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('LSTM model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
#file_name = '/content/drive/MyDrive/DM HW4 picture/'+data_set_name+'accuracy.png'
#plt.savefig(file_name)
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('LSTM model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
#file_name = '/content/drive/MyDrive/DM HW4 picture/'+data_set_name+'loss.png'
#plt.savefig(file_name)
plt.show()
```





3.1 CNN模型評估(early stopping，訓練到epochs=3即停止，避免overfitting)

利用test的資料對建立的CNN模型進行測試，並計算Accuracy

In [12]:
```
modelCNN2 = Sequential()
modelCNN2.add(Embedding(output_dim=300, input_dim=10000, input_length=2000))
modelCNN2.add(Conv1D(32, 3, activation='relu', padding="same", kernel_regularizer=l2(0.01), bias_regularizer=l2(0.01)))
modelCNN2.add(MaxPooling1D(3))
modelCNN2.add(Conv1D(32, 3, activation='relu', padding="same", kernel_regularizer=l2(0.01), bias_regularizer=l2(0.01)))
modelCNN2.add(MaxPooling1D(3))
modelCNN2.add(Flatten()) #需要
modelCNN2.add(Dropout(0.7))
modelCNN2.add(Dense(128, activation='relu'))
modelCNN2.add(Dense(1, activation='sigmoid'))

print(modelCNN2.summary())
modelCNN2.compile(loss='binary_crossentropy',optimizer='Nadam',metrics=['accuracy'])
history = modelCNN2.fit(x_train, y_train, epochs = 3, batch_size = 50, verbose = 1, validation_split = 0.2)

CNN_scores = modelCNN2.evaluate(x_test, y_test,verbose=1)
print("CNN test set accuracy = ",round(CNN_scores[1],3))
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_5 (Embedding)     (None, 2000, 300)         3000000

 conv1d_6 (Conv1D)           (None, 2000, 32)          28832

 max_pooling1d_6 (MaxPooling  (None, 666, 32)          0
 1D)

 conv1d_7 (Conv1D)           (None, 666, 32)           3104

 max_pooling1d_7 (MaxPooling  (None, 222, 32)          0
 1D)

 flatten_3 (Flatten)         (None, 7104)              0

 dropout_7 (Dropout)         (None, 7104)              0

 dense_10 (Dense)            (None, 128)               909440

 dense_11 (Dense)            (None, 1)                 129

=================================================================
Total params: 3,941,505
Trainable params: 3,941,505
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/3
128/128 [==============================] - 7s 46ms/step - loss: 0.9086 - accuracy: 0.6866 - val_loss: 0.6598 - val_accuracy: 0.7350
Epoch 2/3
128/128 [==============================] - 6s 44ms/step - loss: 0.5003 - accuracy: 0.8055 - val_loss: 0.4465 - val_accuracy: 0.8344
Epoch 3/3
128/128 [==============================] - 6s 44ms/step - loss: 0.3439 - accuracy: 0.8831 - val_loss: 0.4546 - val_accuracy: 0.8381
63/63 [==============================] - 1s 10ms/step - loss: 0.4406 - accuracy: 0.8370
CNN test set accuracy =  0.837
```

3.2 LSTM模型評估(early stopping，訓練到epochs=2即停止，避免overfitting)

利用test的資料對建立的LSTM模型進行測試，並計算Accuracy

In [14]:
```
modelLSTM2 = Sequential()
modelLSTM2.add(Embedding(output_dim=300, input_dim=10000, input_length=2000))
modelLSTM2.add(Dropout(0.2))
modelLSTM2.add(CuDNNLSTM(32))
modelLSTM2.add(Dense(units=256,activation='relu'))
modelLSTM2.add(Dropout(0.2))
modelLSTM2.add(Dense(units=1,activation='sigmoid'))

modelLSTM2.summary()
adam = keras.optimizers.Adam() #default 0.001
modelLSTM2.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'])
history = modelLSTM2.fit(x_train, y_train, epochs = 2, batch_size = 50, verbose = 1, validation_split = 0.2)

LSTM_scores = modelLSTM2.evaluate(x_test, y_test,verbose=1)
print("LSTM test set accuracy = ",round(LSTM_scores[1],3))
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_7 (Embedding)     (None, 2000, 300)         3000000

 dropout_10 (Dropout)        (None, 2000, 300)         0

 cu_dnnlstm_2 (CuDNNLSTM)    (None, 32)                42752

 dense_14 (Dense)            (None, 256)               8448

 dropout_11 (Dropout)        (None, 256)               0

 dense_15 (Dense)            (None, 1)                 257

=================================================================
Total params: 3,051,457
Trainable params: 3,051,457
Non-trainable params: 0
_________________________________________________________________
Epoch 1/2
128/128 [==============================] - 30s 225ms/step - loss: 0.5567 - accuracy: 0.7248 - val_loss: 0.4537 - val_accuracy: 0.7944
Epoch 2/2
128/128 [==============================] - 28s 222ms/step - loss: 0.3068 - accuracy: 0.8766 - val_loss: 0.4426 - val_accuracy: 0.8250
63/63 [==============================] - 6s 81ms/step - loss: 0.4331 - accuracy: 0.8280
LSTM test set accuracy =  0.828
```