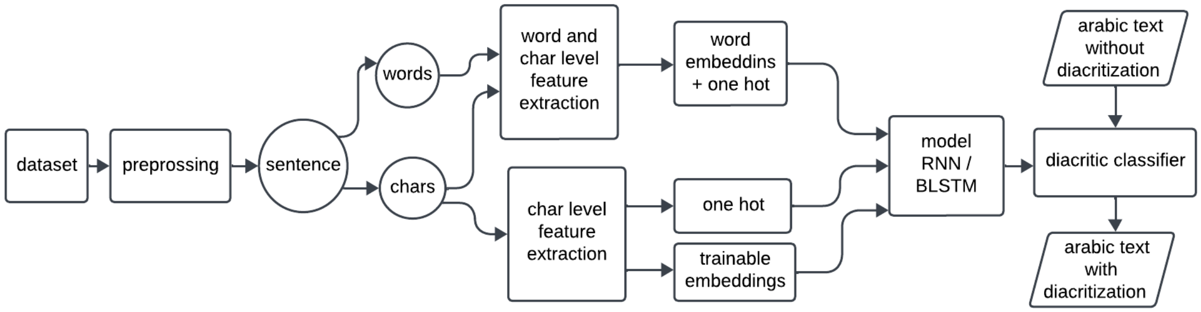
NLP project
(Arabic-Text-Diacritization)

Team: 7

Name	ID	SEC	BN
Asmaa Adel Abdelhamed kawashty	9202285	1	13
Samaa Hazem Mohamed Abdel-latif	9202660	1	31
Norhan Reda Abdelwahed Ahmed	9203639	2	31
Hoda Gamal Hamouda Ismail	9203673	2	33

Supervisor: Eng. Omar Samir galal

a. Project Pipeline


dataset
preprossing
sentence
words
chars
word and char level feature extraction
char level feature extraction
word embeddins + one hot
one hot
trainable embeddings
model RNN / BLSTM
arabic text without diacritization
diacritic classifier
arabic text with diacritization

b. A detailed description of each phase in your pipeline

i. Data preprocessing

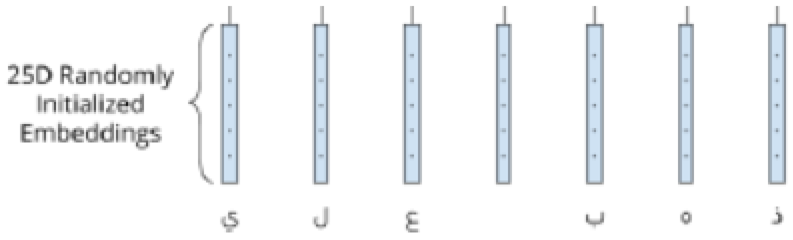
- Split the sentences with punctuations.
- Split into smaller sentences of length no more than 500 characters (without counting diacritics).
  → This step is necessary for the training phase to limit memory usage within a single batch.
- Remove all the non-Arabic characters.
- Remove diacritics.
- Start each sentence with **<s>** and end it with **</s>**
  (both will have a corresponding class **‘no diacritics’ ‘’**)

ii. Feature extraction

We have used 3 approaches for feature extraction:

1. **One Hot encoding (char level)**
   Here we represent the input as a vector of length of the characters that we have and set the value that represent that character with one and the other with zero
   Ex. "أ" = $[1, 0, 0, 0, 0, 0, \ldots\ldots, 0]$
   And the output is also a on hot vector of length of the class diacritization which is 15
   Ex " ً" $= [\,0, 1, 0, 0, 0, \ldots\ldots, 0]$

2. **Trainable embeddings (char level)**
   Here we depend on an embedding layer to learn feature vectors for each character through the training process.



This layer has an `input_dim=len(char_mapping),output_dim=25`
And we used `embeddings_initializer=glorot_normal`
We have tried another types of `embeddings_initializer`
Like `RandomNormal, HeUniform, HeNormal, GlorotUniform`
But the `glorot_normal` was the one that has the max accuracy among all of them

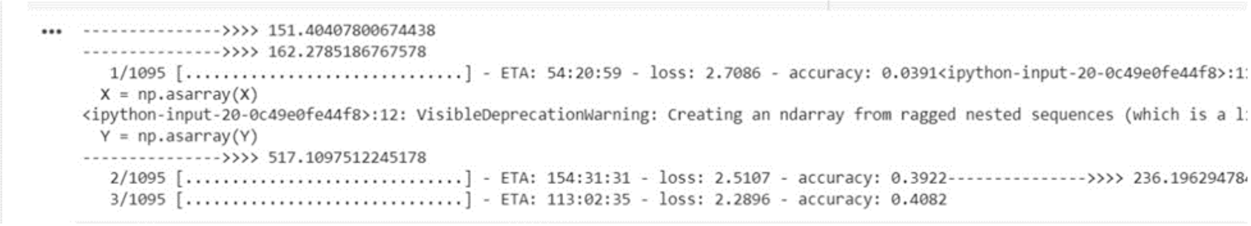
3. **Word2vec embeddings + oneHot (word and char level)**

   We used word2vec for getting the word embedding.
   At first, we trained the word2vec model with training and validation words.
   And then we use this model to get the word embedding.

   For each char, we concatenate the char's one hot vector to its word embedding vector which leads to a vector of size 397.

Another approach for word embeddings:
(unsuccessful trail / not complete)
We have tried to use **Arabert** model for getting word embeddings with model
CAMeL-Lab/bert-base-arabic-camelbert-ca

But it takes much time to get the word embedding so it was taking too long to train even for 1 epoch.
This is an example for feature extraction time for each batch using Arabert word embeddings and char one hot vector:



```
•••  --------------->>>> 151.40407800674438
     --------------->>>> 162.2785186767578
        1/1095 [..............................] - ETA: 54:20:59 - loss: 2.7086 - accuracy: 0.0391<ipython-input-20-0c49e0fe44f8>:1
      X = np.asarray(X)
     <ipython-input-20-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a l
      Y = np.asarray(Y)
     --------------->>>> 517.1097512245178
        2/1095 [..............................] - ETA: 154:31:31 - loss: 2.5107 - accuracy: 0.3922--------------->>>> 236.19629478
        3/1095 [..............................] - ETA: 113:02:35 - loss: 2.2896 - accuracy: 0.4082
```

iii. Model training

Before fitting the model, we do the following:

- Random shuffle the data (for more randomness).
- Sort the data based on length of sentences (to have sentences with closer length together and reduce the overhead of the padding).

In fitting the model, we use:

- **batch_size** = 256

which is not very small because of limited hardware and memory resources and not very high to introduce more generalization.

- **epochs** = 50

to ensure that the model converges and learns enough
but more than 50 epochs, the validation accuracy doesn't significantly improve.

First Model:

- First layer:
  Input Layer with dimension of feature vector length

  Feature vector in case of:
  One hot → one hot vector with length of number of chars we have 97
  Word2vec + One hot → one hot vector with length of number of chars we have 97 + word embedding vector with length of 300 (397)
  (it is replaced in the case of using Trainable embeddings with the **Embedding** layer)

- **3 Bidirectional LSTM layers**
  With 256 units and `kernel_initializer=glorot_normal`

  → 256 hidden units per layer because using a smaller number of units will decrease the accuracy and using a larger number of units doesn't significantly improve it.

- **2 Dropout layers** With dropout rate = 0.5
  (after 1st and 2nd Bidirectional LSTM layers)
  → These Dropout layers help to prevent overfitting.

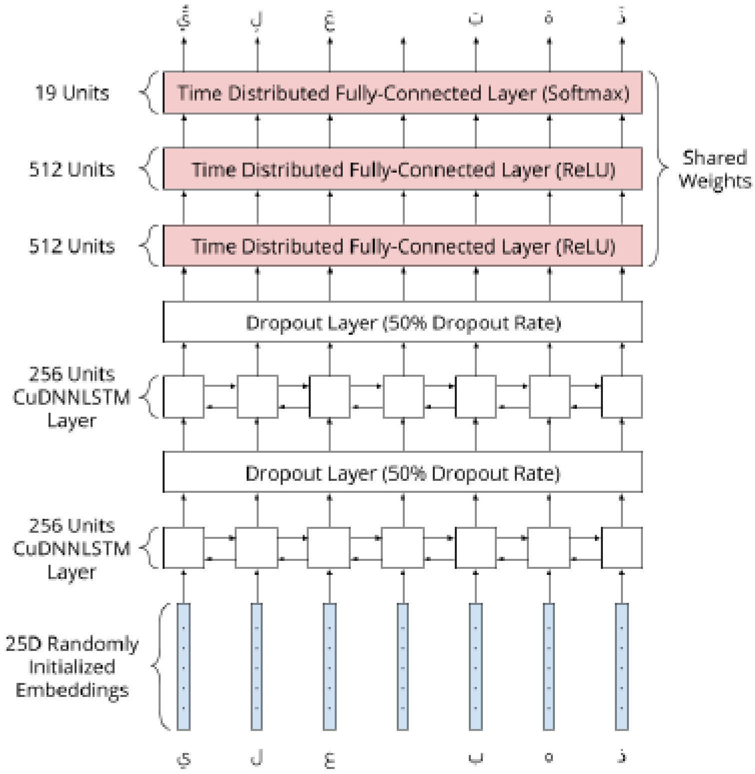
- **3 Dense layers**
  For the first 2 layers of them: **512** units, **Relu** activation function
  For the last one: number of units equal to the number of possible classes or diacritics we have (**15**), **Softmax** activation function
  And for all `kernel_initializer=glorot_normal`

  We have tried another types of `kernel_initializer`
  Like `RandomNormal, HeUniform, HeNormal, GlorotUniform`
  But the `glorot_normal` was the one that has the max accuracy among all of them.

- We use `'categorical_crossentropy'` as the loss function,
  `Adam` as the optimizer for updating the weights,
  And `'accuracy'` as the metric for evaluating the model.

Each dense layer is wrapped with a **TimeDistributed** wrapper which applies the Dense layer independently to each time step.

This helps the model capture temporal patterns and dependencies in the sequence, potentially improving its ability to understand the sequential nature of the data and enhance accuracy.


ذَهَبَ عَلِيٌّ
19 Units
Time Distributed Fully-Connected Layer (Softmax)
512 Units
Time Distributed Fully-Connected Layer (ReLU)
512 Units
Time Distributed Fully-Connected Layer (ReLU)
Shared Weights
Dropout Layer (50% Dropout Rate)
256 Units CuDNNLSTM Layer
Dropout Layer (50% Dropout Rate)
256 Units CuDNNLSTM Layer
25D Randomly Initialized Embeddings
ذهب علي

Second Model:

- The same as the first one but with **RNN layers** instead of Bidirectional LSTM.

Third Model:

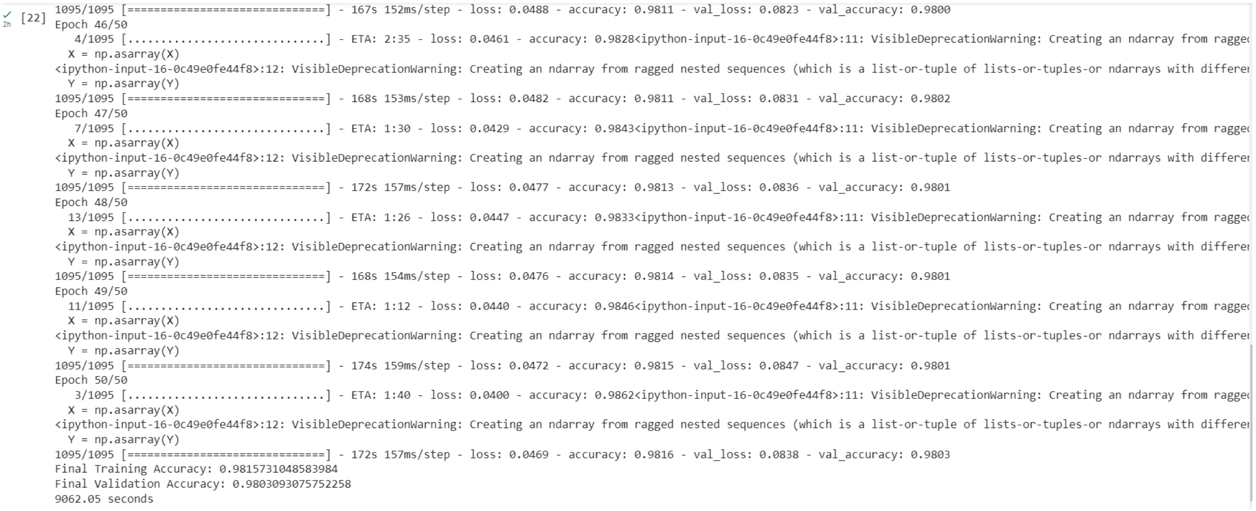
- The same as the first one but with **GRU layers** instead of Bidirectional LSTM.

c. Evaluation: Report the DER for all trials you did.

DER = 1 - accuracy

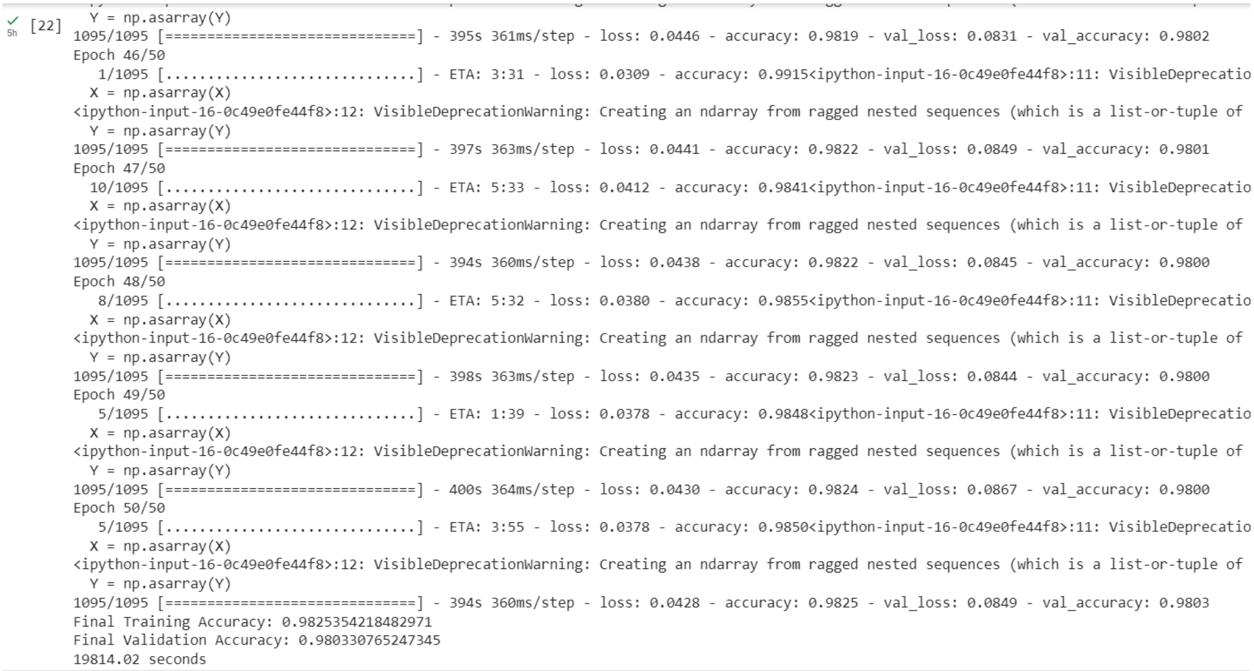
model	Feature extraction	Validation accuracy	Validation DER
BLSTM	oneHot	98.033077 %	1.966923
BLSTM	Trainable embeddings	98.030931%	1.969069
BLSTM	Word2vec +oneHot	94.686555%	5.313444
RNN	Trainable embeddings	87.37234%	12.62766
RNN	oneHot	87.27 %	12.73
GRU	Trainable embeddings	81.6587 %	18.3413

Blstm Embedding



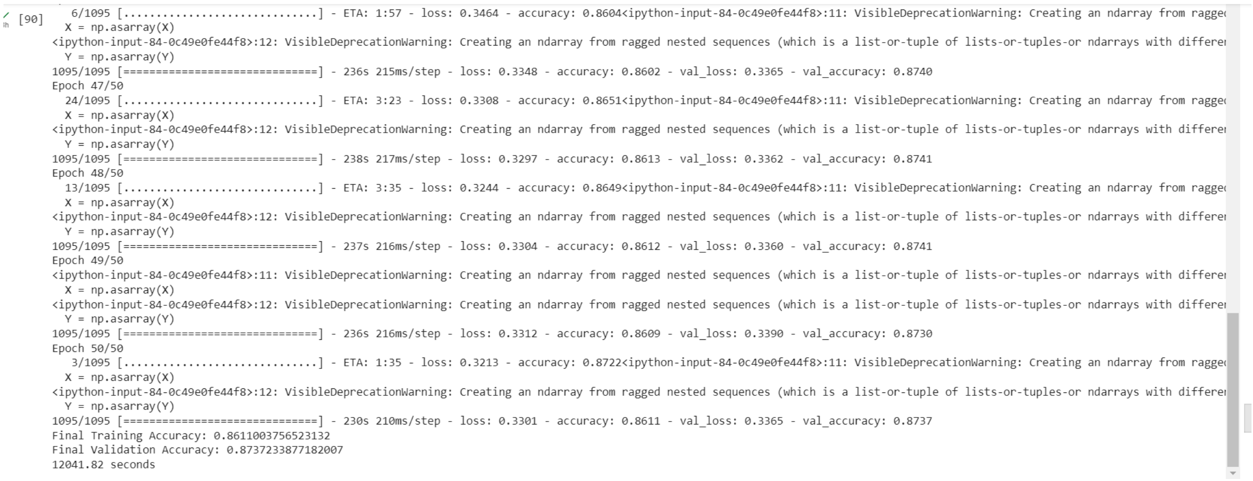
```
1095/1095 [==============================] - 167s 152ms/step - loss: 0.0488 - accuracy: 0.9811 - val_loss: 0.0823 - val_accuracy: 0.9800
Epoch 46/50
   4/1095 [..............................] - ETA: 2:35 - loss: 0.0461 - accuracy: 0.9828<ipython-input-16-0c49e0fe44f8>:11: VisibleDeprecationWarning: Creating an ndarray from ragge
  X = np.asarray(X)
<ipython-input-16-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with differe
  Y = np.asarray(Y)
1095/1095 [==============================] - 168s 153ms/step - loss: 0.0482 - accuracy: 0.9811 - val_loss: 0.0831 - val_accuracy: 0.9802
Epoch 47/50
   7/1095 [..............................] - ETA: 1:30 - loss: 0.0429 - accuracy: 0.9843<ipython-input-16-0c49e0fe44f8>:11: VisibleDeprecationWarning: Creating an ndarray from ragge
  X = np.asarray(X)
<ipython-input-16-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with differe
  Y = np.asarray(Y)
1095/1095 [==============================] - 172s 157ms/step - loss: 0.0477 - accuracy: 0.9813 - val_loss: 0.0836 - val_accuracy: 0.9801
Epoch 48/50
  13/1095 [..............................] - ETA: 1:26 - loss: 0.0447 - accuracy: 0.9833<ipython-input-16-0c49e0fe44f8>:11: VisibleDeprecationWarning: Creating an ndarray from ragge
  X = np.asarray(X)
<ipython-input-16-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with differe
  Y = np.asarray(Y)
1095/1095 [==============================] - 168s 154ms/step - loss: 0.0476 - accuracy: 0.9814 - val_loss: 0.0835 - val_accuracy: 0.9801
Epoch 49/50
  11/1095 [..............................] - ETA: 1:12 - loss: 0.0440 - accuracy: 0.9846<ipython-input-16-0c49e0fe44f8>:11: VisibleDeprecationWarning: Creating an ndarray from ragge
  X = np.asarray(X)
<ipython-input-16-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with differe
  Y = np.asarray(Y)
1095/1095 [==============================] - 174s 159ms/step - loss: 0.0472 - accuracy: 0.9815 - val_loss: 0.0847 - val_accuracy: 0.9801
Epoch 50/50
   3/1095 [..............................] - ETA: 1:40 - loss: 0.0400 - accuracy: 0.9862<ipython-input-16-0c49e0fe44f8>:11: VisibleDeprecationWarning: Creating an ndarray from ragge
  X = np.asarray(X)
<ipython-input-16-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with differe
  Y = np.asarray(Y)
1095/1095 [==============================] - 172s 157ms/step - loss: 0.0469 - accuracy: 0.9816 - val_loss: 0.0838 - val_accuracy: 0.9803
Final Training Accuracy: 0.9815731048583984
Final Validation Accuracy: 0.9803093075752258
9062.05 seconds
```

Blstm One Hot



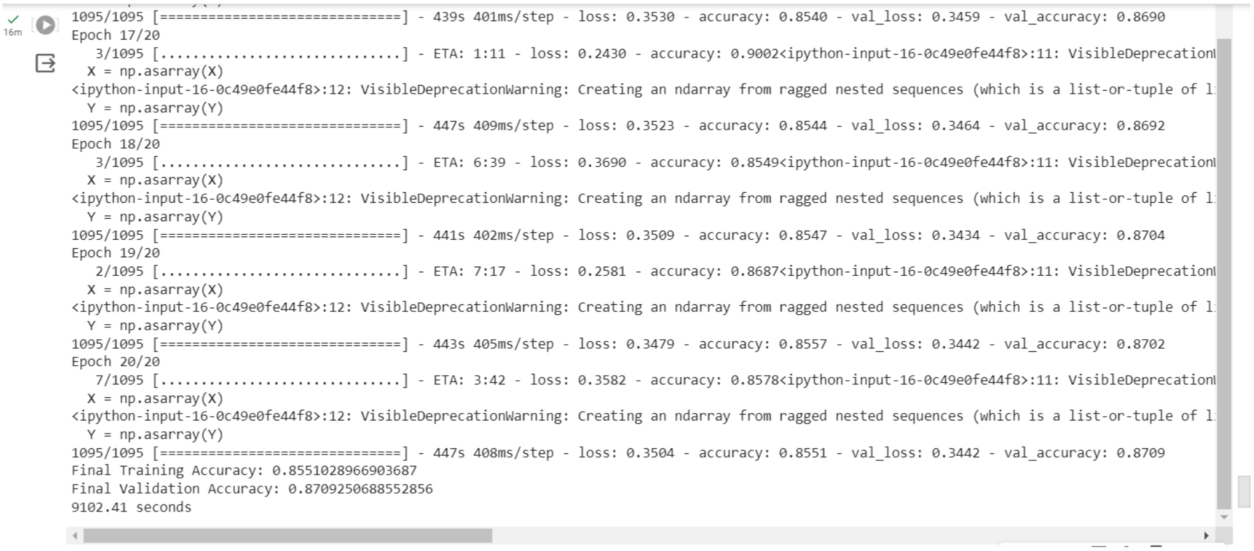
```
  Y = np.asarray(Y)
1095/1095 [==============================] - 395s 361ms/step - loss: 0.0446 - accuracy: 0.9819 - val_loss: 0.0831 - val_accuracy: 0.9802
Epoch 46/50
   1/1095 [..............................] - ETA: 3:31 - loss: 0.0309 - accuracy: 0.9915<ipython-input-16-0c49e0fe44f8>:11: VisibleDeprecatio
  X = np.asarray(X)
<ipython-input-16-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of
  Y = np.asarray(Y)
1095/1095 [==============================] - 397s 363ms/step - loss: 0.0441 - accuracy: 0.9822 - val_loss: 0.0849 - val_accuracy: 0.9801
Epoch 47/50
  10/1095 [..............................] - ETA: 5:33 - loss: 0.0412 - accuracy: 0.9841<ipython-input-16-0c49e0fe44f8>:11: VisibleDeprecatio
  X = np.asarray(X)
<ipython-input-16-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of
  Y = np.asarray(Y)
1095/1095 [==============================] - 394s 360ms/step - loss: 0.0438 - accuracy: 0.9822 - val_loss: 0.0845 - val_accuracy: 0.9800
Epoch 48/50
   8/1095 [..............................] - ETA: 5:32 - loss: 0.0380 - accuracy: 0.9855<ipython-input-16-0c49e0fe44f8>:11: VisibleDeprecatio
  X = np.asarray(X)
<ipython-input-16-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of
  Y = np.asarray(Y)
1095/1095 [==============================] - 398s 363ms/step - loss: 0.0435 - accuracy: 0.9823 - val_loss: 0.0844 - val_accuracy: 0.9800
Epoch 49/50
   5/1095 [..............................] - ETA: 1:39 - loss: 0.0378 - accuracy: 0.9848<ipython-input-16-0c49e0fe44f8>:11: VisibleDeprecatio
  X = np.asarray(X)
<ipython-input-16-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of
  Y = np.asarray(Y)
1095/1095 [==============================] - 400s 364ms/step - loss: 0.0430 - accuracy: 0.9824 - val_loss: 0.0867 - val_accuracy: 0.9800
Epoch 50/50
   5/1095 [..............................] - ETA: 3:55 - loss: 0.0378 - accuracy: 0.9850<ipython-input-16-0c49e0fe44f8>:11: VisibleDeprecatio
  X = np.asarray(X)
<ipython-input-16-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of
  Y = np.asarray(Y)
1095/1095 [==============================] - 394s 360ms/step - loss: 0.0428 - accuracy: 0.9825 - val_loss: 0.0849 - val_accuracy: 0.9803
Final Training Accuracy: 0.9825354218482971
Final Validation Accuracy: 0.980330765247345
19814.02 seconds
```

Rnn embeddings



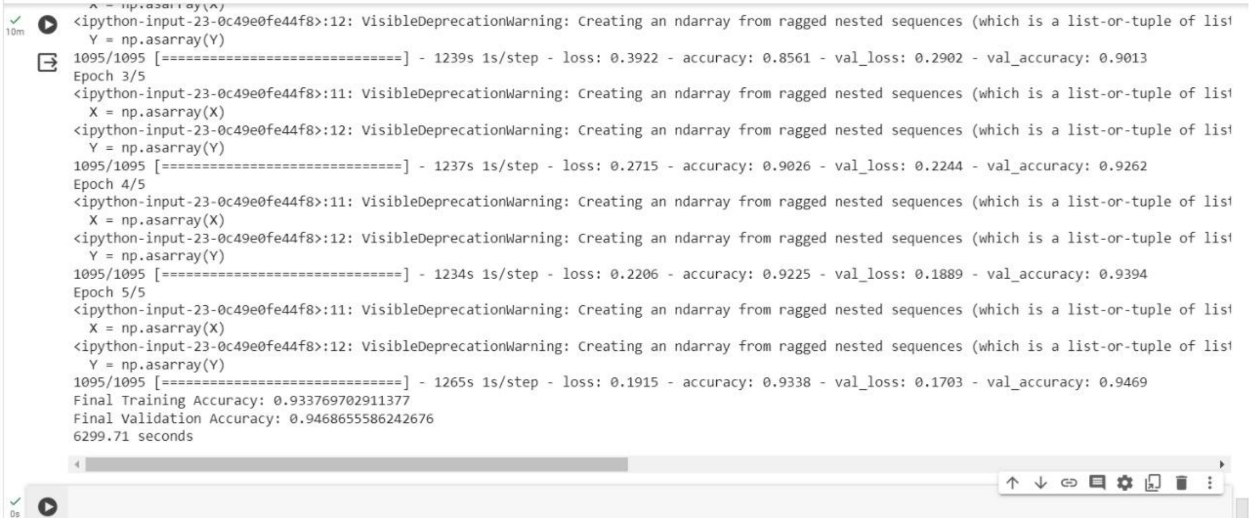
```
  6/1095 [..............................] - ETA: 1:57 - loss: 0.3464 - accuracy: 0.8604<ipython-input-84-0c49e0fe44f8>:11: VisibleDeprecationWarning: Creating an ndarray from ragge
  X = np.asarray(X)
<ipython-input-84-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with differe
  Y = np.asarray(Y)
1095/1095 [==============================] - 236s 215ms/step - loss: 0.3348 - accuracy: 0.8602 - val_loss: 0.3365 - val_accuracy: 0.8740
Epoch 47/50
  24/1095 [..............................] - ETA: 3:23 - loss: 0.3308 - accuracy: 0.8651<ipython-input-84-0c49e0fe44f8>:11: VisibleDeprecationWarning: Creating an ndarray from ragge
  X = np.asarray(X)
<ipython-input-84-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with differe
  Y = np.asarray(Y)
1095/1095 [==============================] - 238s 217ms/step - loss: 0.3297 - accuracy: 0.8613 - val_loss: 0.3362 - val_accuracy: 0.8741
Epoch 48/50
  13/1095 [..............................] - ETA: 3:35 - loss: 0.3244 - accuracy: 0.8649<ipython-input-84-0c49e0fe44f8>:11: VisibleDeprecationWarning: Creating an ndarray from ragge
  X = np.asarray(X)
<ipython-input-84-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with differe
  Y = np.asarray(Y)
1095/1095 [==============================] - 237s 216ms/step - loss: 0.3304 - accuracy: 0.8612 - val_loss: 0.3360 - val_accuracy: 0.8741
Epoch 49/50
<ipython-input-84-0c49e0fe44f8>:11: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with differe
  X = np.asarray(X)
<ipython-input-84-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with differe
  Y = np.asarray(Y)
1095/1095 [==============================] - 236s 216ms/step - loss: 0.3312 - accuracy: 0.8609 - val_loss: 0.3390 - val_accuracy: 0.8730
Epoch 50/50
   3/1095 [..............................] - ETA: 1:35 - loss: 0.3213 - accuracy: 0.8722<ipython-input-84-0c49e0fe44f8>:11: VisibleDeprecationWarning: Creating an ndarray from ragge
  X = np.asarray(X)
<ipython-input-84-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with differe
  Y = np.asarray(Y)
1095/1095 [==============================] - 230s 210ms/step - loss: 0.3301 - accuracy: 0.8611 - val_loss: 0.3365 - val_accuracy: 0.8737
Final Training Accuracy: 0.8611003756523132
Final Validation Accuracy: 0.8737233877182007
12041.82 seconds
```

Rnn one hot



```
1095/1095 [==============================] - 439s 401ms/step - loss: 0.3530 - accuracy: 0.8540 - val_loss: 0.3459 - val_accuracy: 0.8690
Epoch 17/20
   3/1095 [..............................] - ETA: 1:11 - loss: 0.2430 - accuracy: 0.9002<ipython-input-16-0c49e0fe44f8>:11: VisibleDeprecation
  X = np.asarray(X)
<ipython-input-16-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of l
  Y = np.asarray(Y)
1095/1095 [==============================] - 447s 409ms/step - loss: 0.3523 - accuracy: 0.8544 - val_loss: 0.3464 - val_accuracy: 0.8692
Epoch 18/20
   3/1095 [..............................] - ETA: 6:39 - loss: 0.3690 - accuracy: 0.8549<ipython-input-16-0c49e0fe44f8>:11: VisibleDeprecation
  X = np.asarray(X)
<ipython-input-16-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of l
  Y = np.asarray(Y)
1095/1095 [==============================] - 441s 402ms/step - loss: 0.3509 - accuracy: 0.8547 - val_loss: 0.3434 - val_accuracy: 0.8704
Epoch 19/20
   2/1095 [..............................] - ETA: 7:17 - loss: 0.2581 - accuracy: 0.8687<ipython-input-16-0c49e0fe44f8>:11: VisibleDeprecation
  X = np.asarray(X)
<ipython-input-16-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of l
  Y = np.asarray(Y)
1095/1095 [==============================] - 443s 405ms/step - loss: 0.3479 - accuracy: 0.8557 - val_loss: 0.3442 - val_accuracy: 0.8702
Epoch 20/20
   7/1095 [..............................] - ETA: 3:42 - loss: 0.3582 - accuracy: 0.8578<ipython-input-16-0c49e0fe44f8>:11: VisibleDeprecation
  X = np.asarray(X)
<ipython-input-16-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of l
  Y = np.asarray(Y)
1095/1095 [==============================] - 447s 408ms/step - loss: 0.3504 - accuracy: 0.8551 - val_loss: 0.3442 - val_accuracy: 0.8709
Final Training Accuracy: 0.8551028966903687
Final Validation Accuracy: 0.8709250688552856
9102.41 seconds
```

BLSTM word2vec



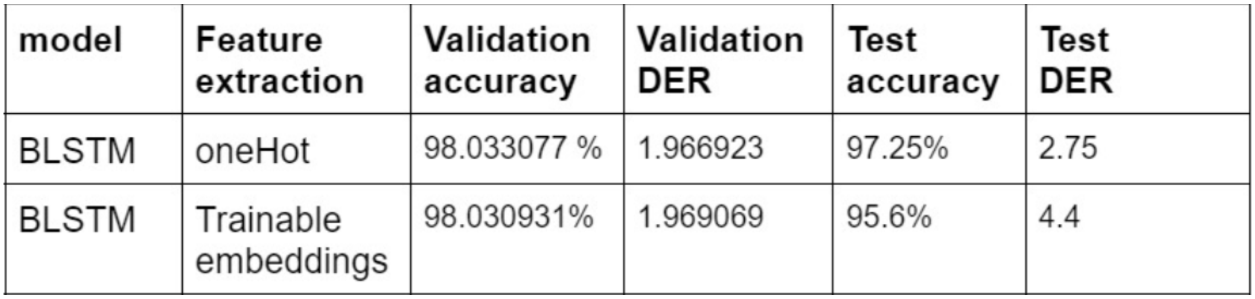
```
  X = np.asarray(X)
<ipython-input-23-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of list
  Y = np.asarray(Y)
1095/1095 [==============================] - 1239s 1s/step - loss: 0.3922 - accuracy: 0.8561 - val_loss: 0.2902 - val_accuracy: 0.9013
Epoch 3/5
<ipython-input-23-0c49e0fe44f8>:11: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of list
  X = np.asarray(X)
<ipython-input-23-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of list
  Y = np.asarray(Y)
1095/1095 [==============================] - 1237s 1s/step - loss: 0.2715 - accuracy: 0.9026 - val_loss: 0.2244 - val_accuracy: 0.9262
Epoch 4/5
<ipython-input-23-0c49e0fe44f8>:11: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of list
  X = np.asarray(X)
<ipython-input-23-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of list
  Y = np.asarray(Y)
1095/1095 [==============================] - 1234s 1s/step - loss: 0.2206 - accuracy: 0.9225 - val_loss: 0.1889 - val_accuracy: 0.9394
Epoch 5/5
<ipython-input-23-0c49e0fe44f8>:11: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of list
  X = np.asarray(X)
<ipython-input-23-0c49e0fe44f8>:12: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of list
  Y = np.asarray(Y)
1095/1095 [==============================] - 1265s 1s/step - loss: 0.1915 - accuracy: 0.9338 - val_loss: 0.1703 - val_accuracy: 0.9469
Final Training Accuracy: 0.933769702911377
Final Validation Accuracy: 0.9468655586242676
6299.71 seconds
```

d. Specify what model you used for the test set submission on Kaggle and the reason for choosing it.

BLSTM model with char embedding layer

It’s the model which gives us the highest accuracy in the submission.

We expected the BLSTM model with char one hot vector to be the highest as it was a little bit higher in the validation accuracy, and higher in the accuracy when testing on the sample test set given in the first (even though it is only one line).



model	Feature extraction	Validation accuracy	Validation DER	Test accuracy	Test DER
BLSTM	oneHot	98.033077 %	1.966923	97.25%	2.75
BLSTM	Trainable embeddings	98.030931%	1.969069	95.6%	4.4

We also tried many trials such as:

- changing in model layers such as increasing BLSTM layers.
- Increasing output dimension of the embedding layer.
- Increasing number of epochs in training.

But no improvement.