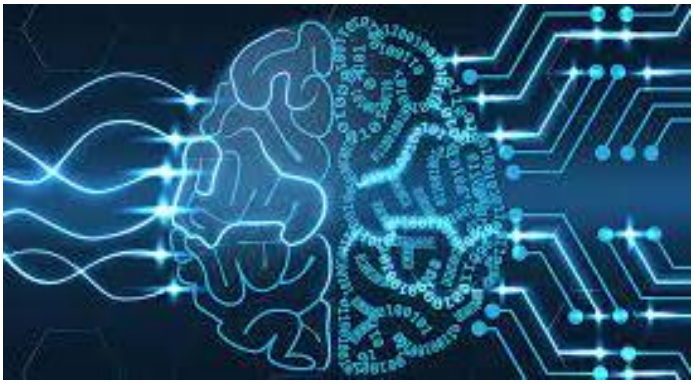
جامعة عبد المالك السعدي
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵏ ⵄⴰⴱⴷⵍⵎⴰⵍⵉⴽ ⵙⵙⴰⵄⴷⵉ
Université Abdelmalek Essaadi

FACULTE DES SCIENCES ET TECHNIQUES
TANGER
كلية العلوم والتقنيات طنجة

Data analytics / Deep Learning

Arabic Automated short answers grading system and Smart Assistance for Islamic Education for scholars



Réalisé par :

Hajar hbibiali

Chaimae Aboulouafa

Encadré par :

Prof.El AACHAK lotfi

Table of contents

1-Introduction :

Deep learning is a subset of machine learning that involves neural networks with multiple layers (deep neural networks). These networks are capable of learning intricate hierarchical representations of data through the training process. Deep learning has shown significant success in various tasks such as image and speech recognition, natural language processing, and even playing games. The key components of deep learning include:

-Neural Networks: Deep learning models are often based on artificial neural networks that mimic the structure and function of the human brain.

- Layers: Deep neural networks consist of multiple layers, including input, hidden, and output layers. Each layer contains nodes (neurons) that process and transform the input data.

- Training: Deep learning models are trained using large datasets to learn patterns and relationships within the data. This process involves adjusting the model's parameters through backpropagation and optimization algorithms.

Generative AI: Generative AI refers to the development of algorithms and models that can generate new content or data that is similar to existing examples. This involves creating something new, whether it's images, text, music, or other forms of data. Generative models have gained popularity in recent years, and they are often powered by deep learning techniques. Types of generative models include:

- Generative Adversarial Networks (GANs): GANs consist of a generator and a discriminator network that are trained together. The generator aims to

create realistic data, while the discriminator tries to distinguish between real and generated data. This adversarial training process leads to the generation of high-quality content.

-Variational Autoencoders (VAEs): VAEs are another type of generative model that focuses on learning the underlying distribution of the data. They map input data to a latent space, allowing for the generation of new samples by sampling from this space. Applications of generative AI include image synthesis, text generation, style transfer, and even creating realistic deepfakes.

2-Understanding of the Business Problem:

In the context of this project, our objective is to develop an Arabic Automated short answers grading system and Smart Assistance for islamic education for schoolers. The main goal is to provide accurate and appropriate assessments to students based on their responses. The idea behind this system is to give the adequate grade to the students according to their answers (grades from 0 to 20), the system should be in Arabic, We aim to evaluate their comprehension of Islamic Kbnowledge, and assit them during this test , their ability to analyze relevant information, and formulate coherent arguments. Through this system, we aim to improve the grading process for short answers by offering an automated solution that will help teachers save time and provide more precise and constructive feedback to students. We seek to create a more efficient and interactive learning environment where students can receive quick feedback on their

performance and have a tool that will help them understand their strengths and weaknesses in the subject. By fully understanding the business problem, we will be able to develop a system that meets the specific needs of teachers and students. We will strive to provide an objective and consistent assessment while taking into account the linguistic and cultural peculiarities inherent in Arabic and Islamic education. In summary, our goal is to develop an automated system for grading short answers in Arabic related to Islamic Education that will provide accurate and fair assessments to students, while offering teachers an effective tool to evaluate student performance and provide constructive feedback.

3-Understanding of data:

In this project, we have implemented a process to assess students through 10 questions related Islamic Education. These questions have been carefully selected by a former an Evaluation of Islamic knowledge webSite to ensure their relevance and alignment with the curriculum. Each answer receives a score according to its accuracy, relevance and and the quality of its argumentation. A score of 0 is given if the answer is answer is incorrect, a score of 1 is given if the answer is correct but lacking in solid argumentation, and a score of 2 is given if the answer is perfect, with a clear and convincing.

	A	B
1	id_question	question
2	1	من هو خاتم الأنبياء والمرسلين؟
3	2	كم استمرت الدعوة السرية؟
4	3	ما هي السور التي بدأت بالحمد ؟
5	4	ما هو اسم والدة نبي الله عيسى عليه السلام
6	5	من هي أول من دخل في الإسلام من النساء؟
7	6	ما هي السورة التي لا تبتدأ بالبسملة؟
8	7	ما هو اسم مرضعة الرسول صلى الله عليه وسلم ؟
9	8	من هو النبي الذي التقمه الحوت؟
10	9	ما هي السورة التي تحدثت عن تقسيم الغنائم؟
11	10	ما هي أقصر سورة في القرآن الكريم ؟

Fig1 . questions

Then, we have gathered all the possible answers to each question, we have made sure that the answers with good quality variant, real, true, false, and semi true ans repeated answers , so we have collected 100 answer for each question so in total a CSV file with 1000 answer.

	A	B	C
1	question_id	answer	grade
2	1	محمد صلى الله عليه وسلم	2
3	1	خاتم الأنبياء	2
4	1	الرسول الأعظم	2
5	1	سيد المرسلين	2
6	1	صاحب الرسالة الخاتمة	2
7	1	النبي الذي لا يبعث بعده نبي	2
8	1	صاحب الشريعة الخاتمة	2
9	1	نبي الله الذي أرسله الله إلى جميع الناس، وختم به رسالته إلى البشرية	2
10	1	النبي الذي أكرمه الله بآيات كثيرة في القرآن الكريم	2
11	1	نبي الله الذي أقام العدل بين الناس	2
12	1	نبي الله الذي أحبه الناس	2
13	1	نبي الله الذي كان نجاة للناس	2
14	1	لنبي محمد	2
15	1	سيدنا محمد صلى الله عليه وسلم	2
16	1	الرسول محمد	2
17	1	النبي الأكرم محمد صلى الله عليه وسلم	2

.....

	A	B	C
987	10	سورة الكوثر هي أقصر سورة في القرآن الكريم، وهي مكية عدد آياتها ثلاث آيات، وعدد كلماتها عشر كل	2
988	10	الكهف	0
989	10	الفرقان	0
990	10	آل عمران	0
991	10	الإسراء	0
992	10	مريم	0
993	10	نعم، سورة الكوثر تحتوي على ثلاث آيات	2
994	10	الكوثر هي سورة مكية	2
995	10	سورة الكوثر تتميز بقلة عدد آياتها	2
996	10	الكوثر تعتبر هي أقصر سورة	2
997	10	سورة الكوثر تتألف من ثلاث آيات فقط	2
998	10	سورة الكوثر هي أقصر سورة في القرآن الكريم	2
999	10	سورة الكوثر تحتوي على ثلاث آيات، وهو ما قد يعتبر نسبيًا قليلًا مقارنة بسور أخر	1
1000	10	سورة الكوثر هي سورة مكية عدد كلماتها عشر كلمات	1

	A	B	C
379	4	سارة بنت إبراهيم	0
380	4	هاجر أم إسماعيل	0
381	4	ملكة سبأ	0
382	4	القديسة مريم	2
383	4	والدة النبي عيسى تُلقب بـ "فاطمة الطاهرة"،	0
384	4	أمه كانت تُسمى "مريم النبية"	0
385	4	اسم أم النبي عيسى هو "ليلى"،	0
386	4	للرسول عيسى والدتين: والدة جسدية وأخرى سماوية، واسم والدته السماوية هو "صوفيا"	0
387	4	سارة العذراء	0
388	4	هاجر العذراء	0
389	4	اسم والدة نبي الله عيسى عليه السلام هو مريم عليها السلام	2
390	4	اسم والدة نبي الله عيسى عليه السلام هو مريم عليها السلام	2
391	4	اسم والدة نبي الله عيسى عليه السلام هو مريم عليها السلام	2
392	4	والدة نبي الله عيسى عليه السلام : مريم	2
393	4	مريم بنت عمران.	2

Fig2 .answers

4-DATA HUB CONSTRUCTION (DATA PREPARATION)

```
answers = pd.read_csv("../datasets/answers.csv", encoding='utf-8')
answers
```

✓ 0.0s

	question_id	answer	grade
0	1	محمد صلى الله عليه وسلم	2.0
1	1	خاتم الأنبياء	2.0
2	1	الرسول الأعظم	2.0
3	1	سيد المرسلين	2.0
4	1	صاحب الرسالة الخاتمة	2.0
...	...	...	...
986	10	الكهف	0.0
987	10	الفرقان	0.0
988	10	آل عمران	0.0
989	10	الإسراء	0.0
990	10	مريم	0.0

Fig3 .dataset

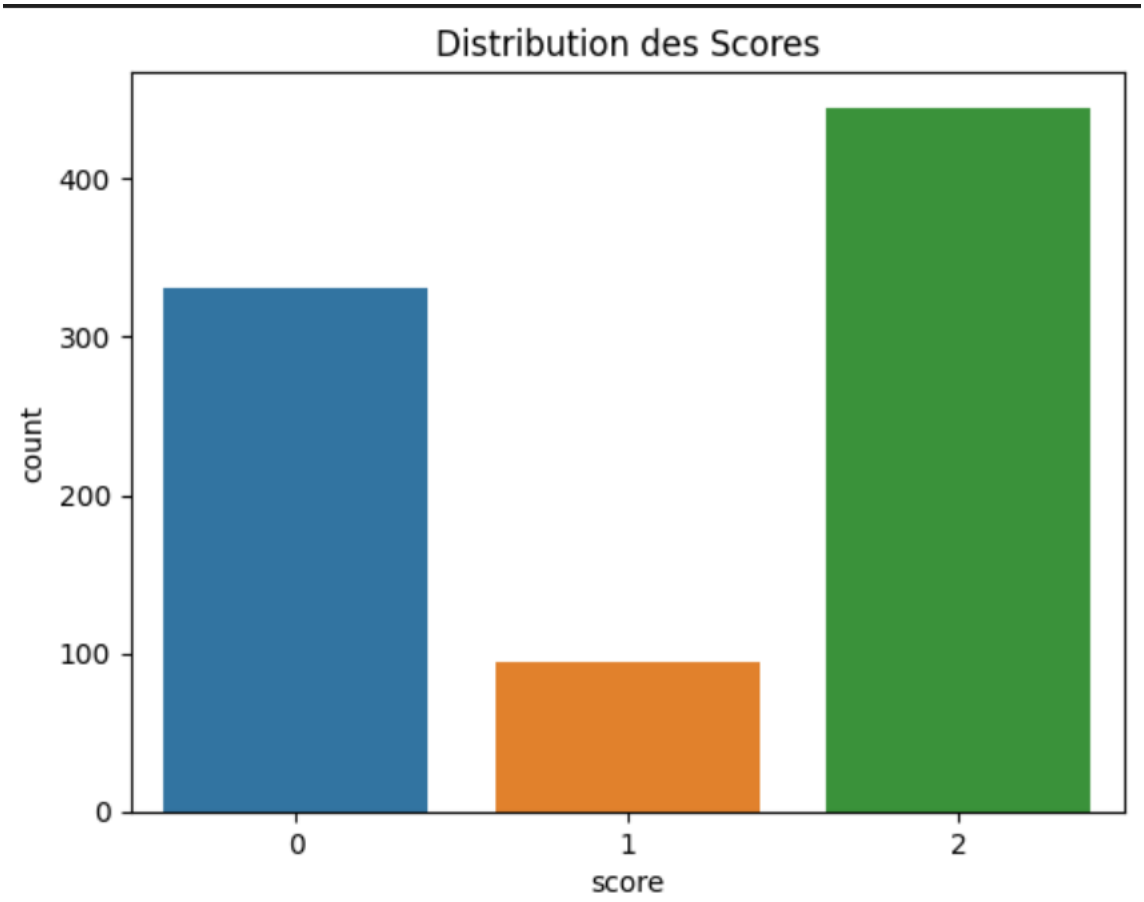
```python
# Supprimer les lignes avec des valeurs manquantes
answers['answer'].isnull().sum()
answers.dropna(inplace=True)
```

```python
# Convertir 'grade' en int64
answers['grade'] = answers['grade'].astype(int)
# Conversion 'question' et 'answer' en type 'str'
answers['answer'] = answers['answer'].astype(str)
```

```python
# Suppression des doublons :
answers.drop_duplicates(inplace=True)
#définir 'id_question' comme index
answers.set_index('question_id', inplace=True)
```

[8]

```python
# Distribution des scores
sns.countplot(x='grade', data=answers)
plt.title('Distribution des Scores')
plt.show()
```




```python
# Function to clean text
def clean_text(text):
    # Define the characters you want to remove
    characters_to_remove = ["'", ':', '؟']

    # Remove specified characters
    for char in characters_to_remove:
        text = text.replace(char, '')

    # Remove `` and ''
    text = text.replace('``', '').replace("''", '')

    return text


# Apply the clean_text function to the 'answer' column in the 'answers' DataFrame
answers['answer'] = answers['answer'].apply(clean_text)
```

```
answers.head(10)
```

question_id	answer	grade
1	محمد صلى الله عليه وسلم	2
1	خاتم الأنبياء	2
1	الرسول الأعظم	2
1	سيد المرسلين	2
1	صاحب الرسالة الخاتمة	2
1	النبي الذي لا يبعث بعده نبي	2
1	صاحب الشريعة الخاتمة	2
1	نبي الله الذي أرسله الله إلى جميع الناس، وختم ...	2
1	النبي الذي أكرمه الله بآيات كثيرة في القرآن ال...	2
1	نبي الله الذي أقام العدل بين الناس	2

```
stu_answers = answers['answer']
scores = answers['grade']
scores = tf.keras.utils.to_categorical(
    scores, num_classes=3, dtype='float32'
)
```

```python
from nltk.stem.arlstem import ARLSTem
stemmer = ARLSTem()
# Function to clean and preprocess text
def preprocess_text(text):
    # Tokenization
    tokens = word_tokenize(text.strip())

    # Removal of stop words
    tokens = [word for word in tokens if word not in arb_stopwords]

    # Stemming
    tokens = [stemmer.stem(word) for word in tokens]

    # Lemmatization
    lemmatized_tokens = [WordNetLemmatizer().lemmatize(word) for word in tokens]

    # Concatenate tokens into a single string
    preprocessed_text = ' '.join(lemmatized_tokens)

    return preprocessed_text, lemmatized_tokens

answers['answer'], answers['answer'] = zip(*answers['answer'].apply(preprocess_text))
```

```
answers
```

question_id	answer	grade
1	[محمد, صلي, الل, سلم]	2
1	[خاتم, نبيء]	2
1	[رسول, اعظم]	2
1	[سيد, مرسل]	2
1	[صاحب, رسال, خاتم]	2
...	...	...
10	[كهف]	0
10	[فرق]	0
10	[ال, عمر]	0
10	[سرء]	0
10	[مريم]	0

870 rows × 2 columns

```python
from gensim.models import KeyedVectors
absolute_path = "./wiki.ar.vec"  # Replace with the actual absolute path
fasttext_model = KeyedVectors.load_word2vec_format(absolute_path)
```

```python
from keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

#train tokenization
tokenizer = Tokenizer(filters='''!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n`÷×_–“…”!|+¦~{}',.؟":/،_][%^&*()_
combined_texts = answers['answer']
tokenizer.fit_on_texts(combined_texts)
sequences = tokenizer.texts_to_sequences(combined_texts)
max_sequence_length = max(len(s) for s in sequences)
sequences = pad_sequences(sequences, max_sequence_length)
word2idx = tokenizer.word_index
vocab_size = len(word2idx) + 1
```

```python
from keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

#train tokenization
tokenizer = Tokenizer(filters='''!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n`÷×_–“…”!|+¦~{}',.؟":/،_][%^&*()_<>:''''' )
combined_texts = answers['answer']
tokenizer.fit_on_texts(combined_texts)
sequences = tokenizer.text
max_sequence_length = max(
sequences = pad_sequences(sequences, max_sequence_length)
word2idx = tokenizer.word_index
vocab_size = len(word2idx) + 1
```

(variable) sequences: list[list]

```
    # Word Embedding
    from keras.layers import Embedding
    import numpy as np

    EMBEDDING_DIM = 300
    num_words = len(word2idx) + 1

    # Prepare embedding matrix
    embedding_matrix = np.zeros((num_words, EMBEDDING_DIM))
    count = 0  # Initialize count here
    for word, idx in word2idx.items():
        if word in fasttext_model:
            embedding_matrix[idx] = fasttext_model.get_vector(word)
        else:
            count += 1
            print("Word not exist in vocab: " + word)
            print(count)
```


```
# Word Embedding
from keras.layers import Embedding
import numpy as np

EMBEDDING_DIM = 300
num_words = len(word2idx) + 1

# Prepare embedding matrix
embedding_matrix = np.zeros((num_words, EMBEDDING_DIM))
count = 0  # Initialize count here
for word, idx in word2idx.items():
    if word in fasttext_model:
        embedding_matrix[idx] = fasttext_model.get_vector(word)
    else:
        count += 1
        print("Word not exist in vocab: " + word)
```

```
Word not exist in vocab: 3
Word not exist in vocab: ''
Word not exist in vocab: ``
Word not exist in vocab: ارضع
Word not exist in vocab: مرضع
Word not exist in vocab: بتلع
Word not exist in vocab: فتتح
Word not exist in vocab: نبيء
Word not exist in vocab: اسلام،
Word not exist in vocab: ربم
Word not exist in vocab: حتوي
Word not exist in vocab: خري
Word not exist in vocab: اربخ
Word not exist in vocab: قران،
Word not exist in vocab: يتامي
Word not exist in vocab: شريع
Word not exist in vocab: عقوب
Word not exist in vocab: 5
Word not exist in vocab: فاطرهي
Word not exist in vocab: بالل
Word not exist in vocab: وزيع
Word not exist in vocab: 10
Word not exist in vocab: عالمين،
Word not exist in vocab: بتدئ
...
Word not exist in vocab: مطفف
```

```
embedding_matrix
```

```
array([[ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.30983999, -0.13467   , -0.73664999, ..., -0.32602999,
        -0.21698999, -0.29073   ],
       [ 0.21615   ,  0.21889   , -0.049481  , ..., -0.13906001,
         0.12826   ,  0.36548001],
       ...,
       [-0.37913001,  0.90614003, -0.30304   , ..., -0.21646   ,
         0.085595  ,  0.34847   ],
       [ 0.024534  ,  0.40884   , -0.45969999, ..., -0.34301001,
         0.17645   , -0.40713999],
       [-0.27902001, -0.022783  , -0.23921999, ..., -0.31531999,
         0.037558  ,  0.30046999]])
```

```
from sklearn.model_selection import train_test_split
# concatenate question number with
X_train, X_test, y_train, y_test = train_test_split(sequences, scores, test_size=0.2)
```

1-LSTM

```
# train model
from keras.regularizers import l1
from keras.models import Sequential
from keras.layers import Dense, Embedding,Input,Dropout,Flatten
from keras.layers import LSTM
from keras.models import Model

print('Build model...')

inp = Input(shape=(max_sequence_length,))
model = Embedding(num_words,
                  EMBEDDING_DIM,
                  weights=[embedding_matrix],
                  input_length=max_sequence_length,
                  trainable=False)(inp)
model = LSTM(units=64, return_sequences=True, return_state=False ,  activation= tf.keras.activations.relu)(model)
model = Dropout(0.2)(model)
model = Flatten()(model)
model = Dense(32, activation=tf.keras.activations.relu)(model)
model = Dropout(0.2)(model)
model = Dense(3, activation=tf.keras.activations.softmax)(model)
model = Model(inputs=inp, outputs=model)
```

```
from keras import backend as K

def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

```
import tensorflow_addons as tfa
#https://neptune.ai/blog/tensorboard-tutorial
tensorboard_callback = TensorBoard(log_dir="./logs")
#Early stopping
es_callback = tf.keras.callbacks.EarlyStopping(monitor='loss',patience=3,verbose=1,mode='min')
keras_callbacks = [
  tensorboard_callback ,es_callback
]
#metrics=['accuracy',tf.keras.metrics.Precision(), tf.keras.metrics.Recall(),f1_m,tfa.metrics.CohenKappa(num_classes=3)]
# try using different optimizers and different optimizer configs
model.compile(loss='categorical_crossentropy',
              optimizer=tf.keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999),
              metrics=['accuracy',tf.keras.metrics.Precision(), tf.keras.metrics.Recall(),f1_m,tfa.metrics.CohenKappa(num_classes=3)])

model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32)]              0

 embedding (Embedding)       (None, 32, 300)           323700

 lstm (LSTM)                 (None, 32, 64)            93440

 dropout (Dropout)           (None, 32, 64)            0

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 32)                65568

 dropout_1 (Dropout)         (None, 32)                0

 dense_1 (Dense)             (None, 3)                 99

=================================================================
Total params: 482807 (1.84 MB)
Trainable params: 159107 (621.51 KB)
Non-trainable params: 323700 (1.23 MB)
```

```
%reload_ext tensorboard
log_folder = 'logs'

print('Train...')
time_start = time()
hisotry = model.fit(X_train, y_train,batch_size=256,epochs=150,callbacks=keras_callbacks,validation_data=(X_test, y_test))
time_start = time() - time_start

print("Took : "+str(np.round(time_start, 2))+" (s)")

#model.save('asag_lstm_model.h5')
```

```
3/3 [==============================] - 11s 928ms/step - loss: 1.0788 - accuracy: 0.3764 - precision: 1.0000 - recall: 0.0057 - f1_m: 0.0142
Epoch 2/150
3/3 [==============================] - 0s 145ms/step - loss: 0.9780 - accuracy: 0.5259 - precision: 0.7326 - recall: 0.1810 - f1_m: 0.2948
Epoch 3/150
3/3 [==============================] - 0s 141ms/step - loss: 0.9336 - accuracy: 0.5216 - precision: 0.6842 - recall: 0.3736 - f1_m: 0.4862
Epoch 4/150
3/3 [==============================] - 0s 159ms/step - loss: 0.9100 - accuracy: 0.5374 - precision: 0.6819 - recall: 0.4066 - f1_m: 0.5097
Epoch 5/150
3/3 [==============================] - 0s 180ms/step - loss: 0.8856 - accuracy: 0.6049 - precision: 0.6841 - recall: 0.3764 - f1_m: 0.4826
Epoch 6/150
3/3 [==============================] - 0s 177ms/step - loss: 0.8669 - accuracy: 0.6221 - precision: 0.7112 - recall: 0.3290 - f1_m: 0.4510
Epoch 7/150
3/3 [==============================] - 0s 157ms/step - loss: 0.8680 - accuracy: 0.6164 - precision: 0.7179 - recall: 0.3290 - f1_m: 0.4481
Epoch 8/150
3/3 [==============================] - 0s 161ms/step - loss: 0.8289 - accuracy: 0.6379 - precision: 0.6944 - recall: 0.3951 - f1_m: 0.5022
Epoch 9/150
3/3 [==============================] - 0s 168ms/step - loss: 0.8205 - accuracy: 0.6365 - precision: 0.7044 - recall: 0.4382 - f1_m: 0.5402
Epoch 10/150
3/3 [==============================] - 1s 268ms/step - loss: 0.8006 - accuracy: 0.6624 - precision: 0.7289 - recall: 0.4828 - f1_m: 0.5861
...
Epoch 58/150
3/3 [==============================] - 0s 120ms/step - loss: 0.1716 - accuracy: 0.9210 - precision: 0.9309 - recall: 0.9095 - f1_m: 0.9222
Epoch 58: early stopping
Took : 32.25 (s)
```

```python
scores_trainig = model.evaluate(X_train, y_train, verbose=1)
print("Training Loss: %f%%" % (scores_trainig[0]))
print("Training Accuracy: %.2f%%" % (scores_trainig[1]*100))
print("Training Precision: %.2f%%" % (scores_trainig[2]*100))
print("Training Recall: %.2f%%" % (scores_trainig[3]*100))
print("Training F1 Score: %.2f%%" % (scores_trainig[4]*100))
print("Training Cohen Kappa: %.2f%%" % (scores_trainig[5]*100))
```

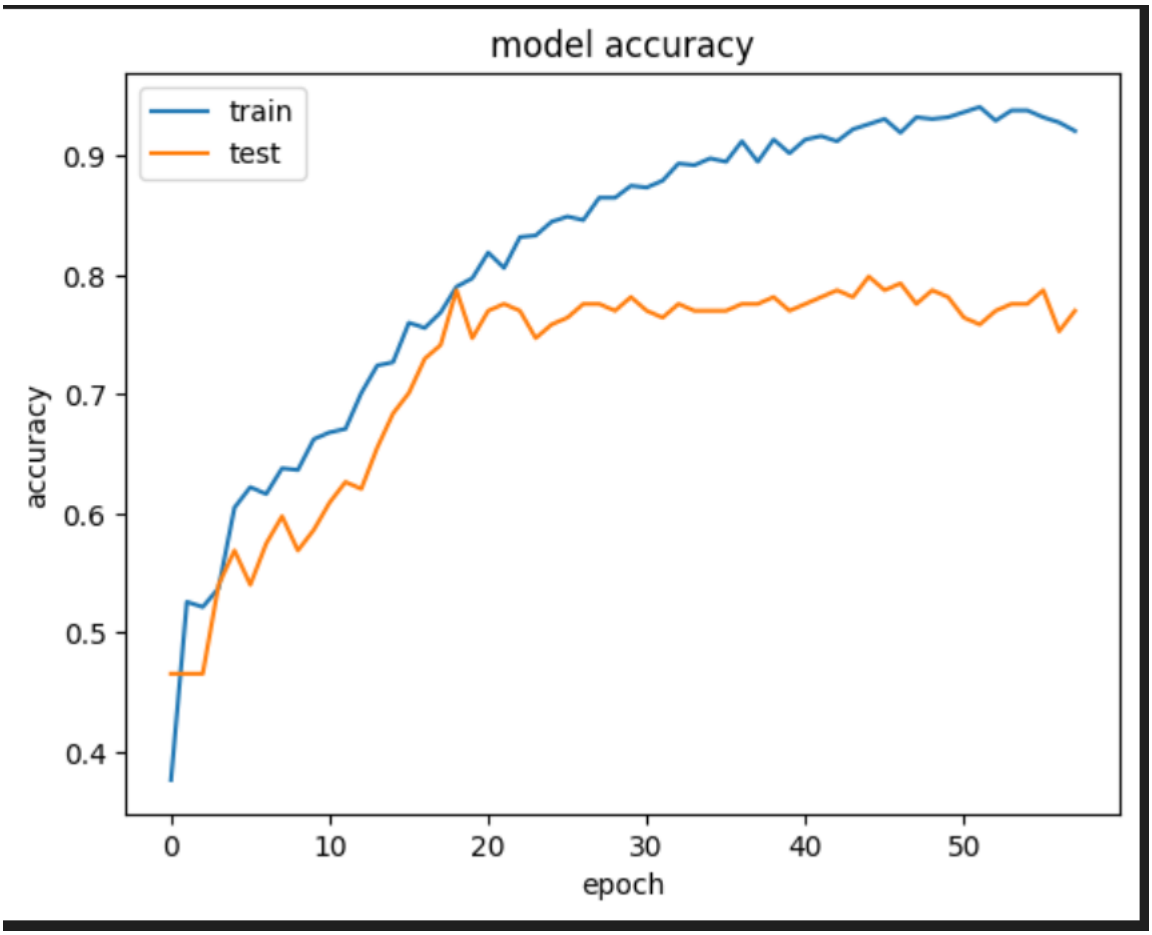
Python

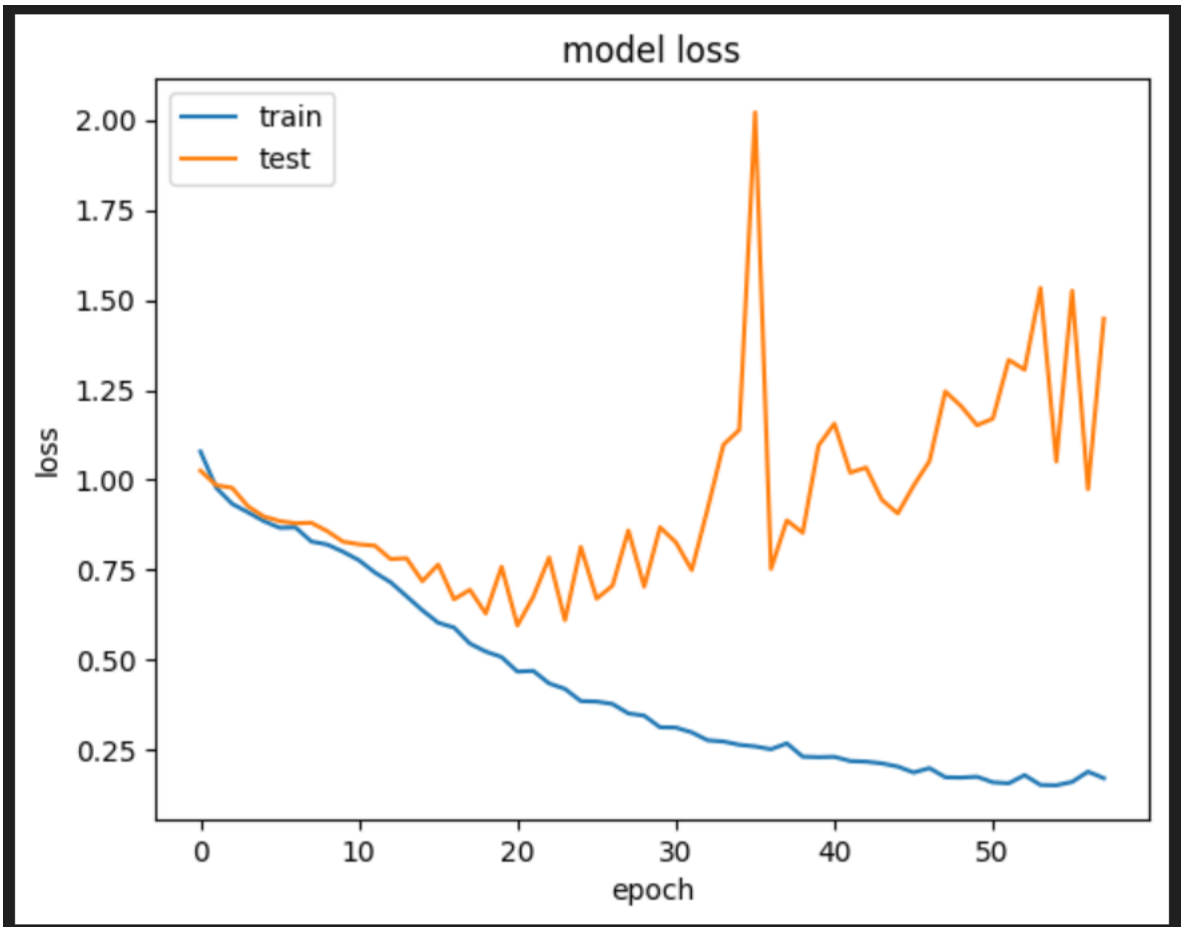
```
22/22 [==============================] - 0s 8ms/step - loss: 0.1356 - accuracy: 0.9397 - precision: 0.9460 - recall: 0.9310 - f1_m: 0.9379 - cohen_kapp
Training Loss: 0.135574%
Training Accuracy: 93.97%
Training Precision: 94.60%
Training Recall: 93.10%
Training F1 Score: 93.79%
Training Cohen Kappa: 89.30%
```

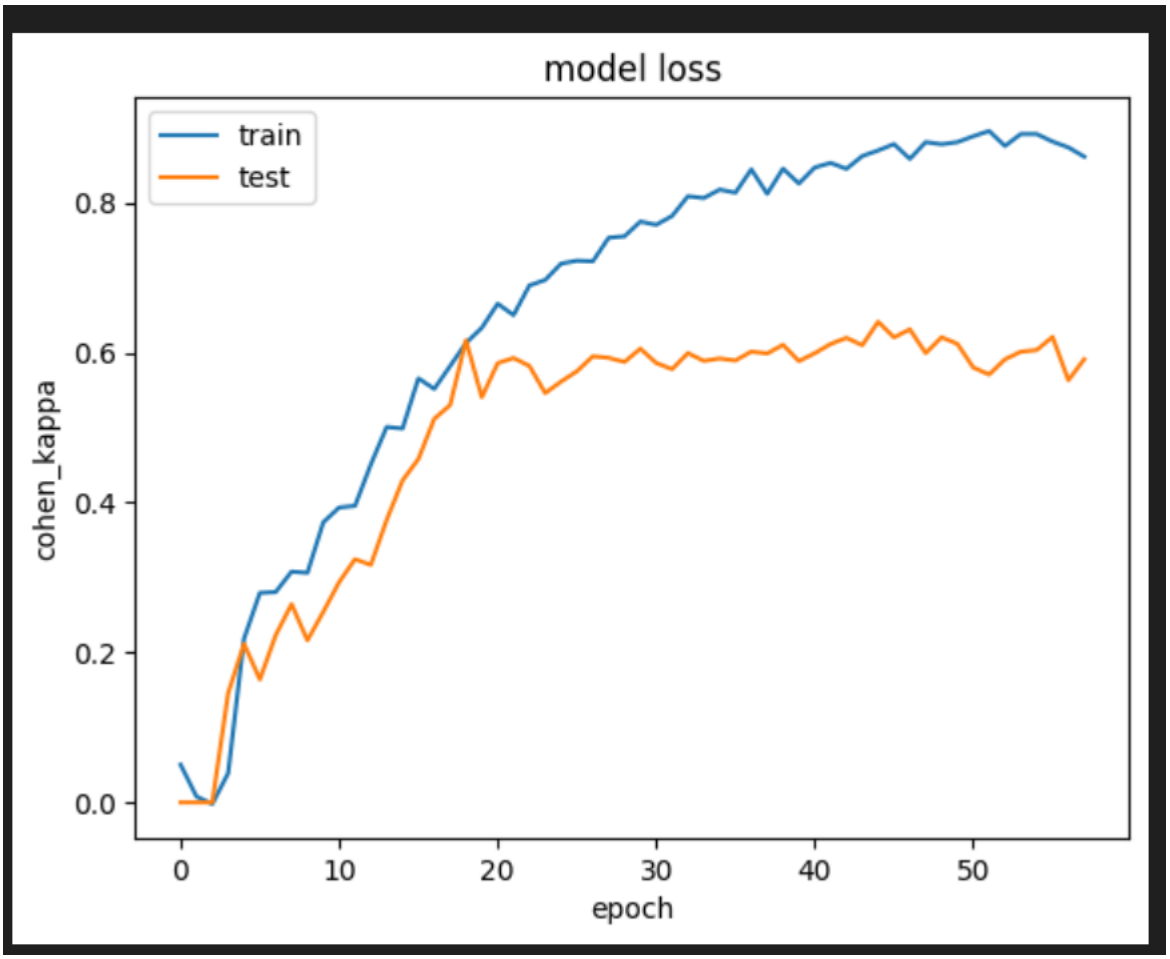
```python
scores_test = model.evaluate(X_test, y_test, verbose=1)
print("Test Loss: %f%%" % (scores_test[0]))
print("Test Accuracy: %.2f%%" % (scores_test[1]*100))
print("Test Precision: %.2f%%" % (scores_test[2]*100))
print("Test Recall: %.2f%%" % (scores_test[3]*100))
print("Test F1 Score: %.2f%%" % (scores_test[4]*100))
print("Test Cohen Kappa: %.2f%%" % (scores_test[5]*100))
```

Python

```
6/6 [==============================] - 0s 10ms/step - loss: 1.4482 - accuracy: 0.7701 - precision: 0.7778 - recall: 0.7644 - f1_m: 0.7724 - cohen_kappa
Test Loss: 1.448241%
Test Accuracy: 77.01%
Test Precision: 77.78%
Test Recall: 76.44%
Test F1 Score: 77.24%
Test Cohen Kappa: 59.13%
```


model accuracy
train
test
accuracy
epoch
0.4
0.5
0.6
0.7
0.8
0.9
0
10
20
30
40
50


model loss
train
test
loss
epoch
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
10
20
30
40
50



1-Transformer

```python
class TokenAndPositionEmbedding(layers.Layer):
    def __init__(self, maxlen, vocab_size, embed_dim ,embedding_matrix):
        super(TokenAndPositionEmbedding, self).__init__()
        self.token_emb = layers.Embedding(input_dim=vocab_size,weights=[embedding_matrix], output_dim=embed_dim)
        self.pos_emb = layers.Embedding(input_dim=maxlen, output_dim=embed_dim)

    def call(self, x):
        maxlen = tf.shape(x)[-1]
        positions = tf.range(start=0, limit=maxlen, delta=1)
        positions = self.pos_emb(positions)
        x = self.token_emb(x)
        return x + positions
```

```
class TransformerBlock(layers.Layer):
    def __init__(self, embed_dim, num_heads, ff_dim, rate=0.1):
        super(TransformerBlock, self).__init__()
        self.att = layers.MultiHeadAttention(num_heads=num_heads, key_dim=embed_dim)
        self.ffn = keras.Sequential(
            [layers.Dense(ff_dim, activation="relu"), layers.Dense(embed_dim),]
        )
        self.layernorm1 = layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = layers.LayerNormalization(epsilon=1e-6)
        self.dropout1 = layers.Dropout(rate)
        self.dropout2 = layers.Dropout(rate)


    def call(self, inputs, training):
        attn_output = self.att(inputs, inputs)
        attn_output = self.dropout1(attn_output, training=training)
        out1 = self.layernorm1(inputs + attn_output)
        ffn_output = self.ffn(out1)
        ffn_output = self.dropout2(ffn_output, training=training)
        return self.layernorm2(out1 + ffn_output)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(sequences, scores, test_size=0.2)
```

```
num_heads = 1  # Number of attention heads
ff_dim = 8  # Hidden layer size in feed forward network inside transformer

inputs = layers.Input(shape=(max_sequence_length,))
embedding_layer = TokenAndPositionEmbedding(max_sequence_length, vocab_size, EMBEDDING_DIM,embedding_matrix)
x = embedding_layer(inputs)
transformer_block = TransformerBlock(EMBEDDING_DIM, num_heads, ff_dim,0.3)
x = transformer_block(x)
x = layers.GlobalMaxPooling1D()(x)
x = layers.Dropout(0.1)(x)
#x = layers.Dense(5, activation="relu")(x)
#x = layers.Dropout(0.1)(x)
outputs = layers.Dense(3, activation="softmax")(x)

model = keras.Model(inputs=inputs, outputs=outputs)
```

```python
from keras import backend as K

def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

```python
import tensorflow_addons as tfa
#https://neptune.ai/blog/tensorboard-tutorial
tensorboard_callback = TensorBoard(log_dir="./logstr")
#Early stopping
es_callback = tf.keras.callbacks.EarlyStopping(monitor='loss',patience=3,verbose=1,mode='min')
keras_callbacks = [
  tensorboard_callback,es_callback
]


model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999), loss="categorical_crossentropy",  metrics=['accurac
#model.save('transformer_model.h5')
model.summary()
```

Python

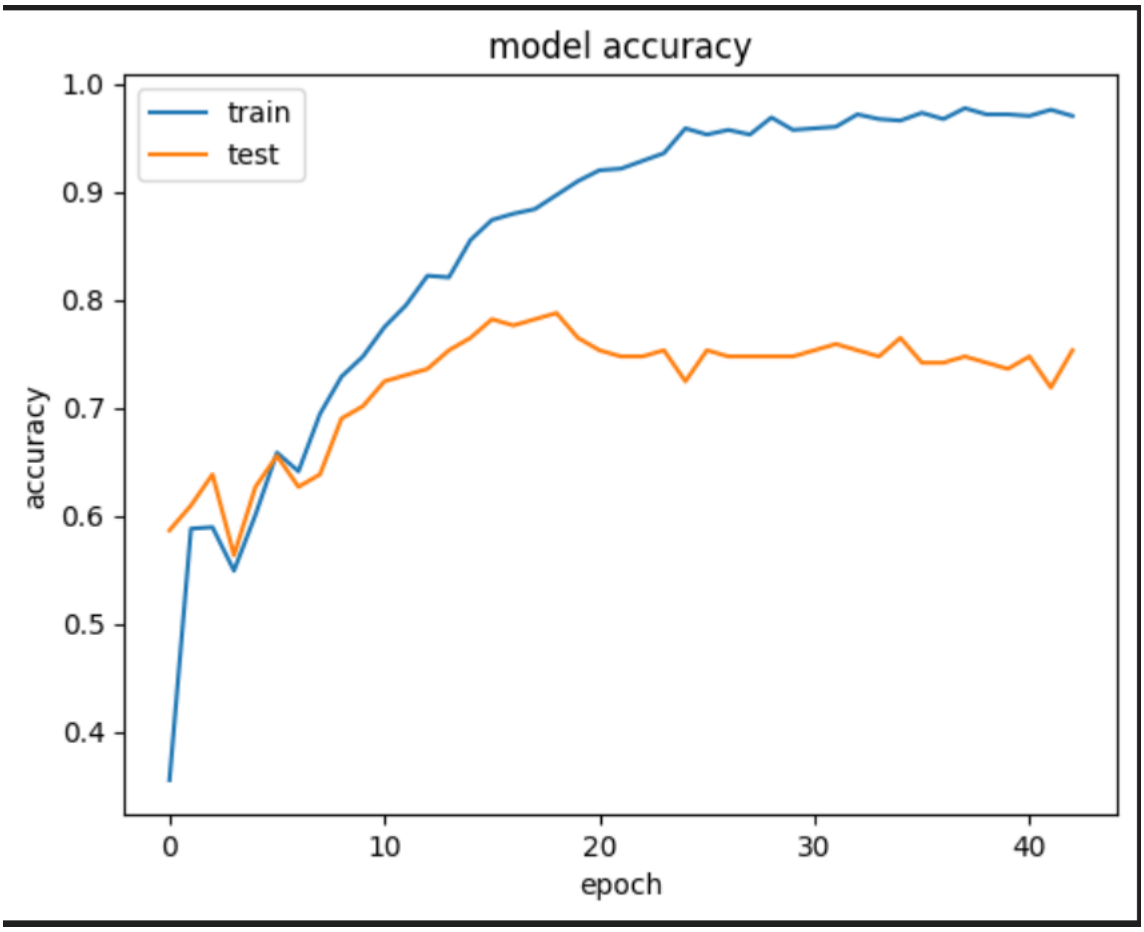
```python
%reload_ext tensorboard
log_folder = 'logstr'
time_start = time()
hisotry = model.fit(
    X_train, y_train, batch_size=256, epochs=100,callbacks=keras_callbacks,validation_data=(X_test, y_test))
time_start = time() - time_start

print("Took : "+str(np.round(time_start, 2))+" (s)")
```

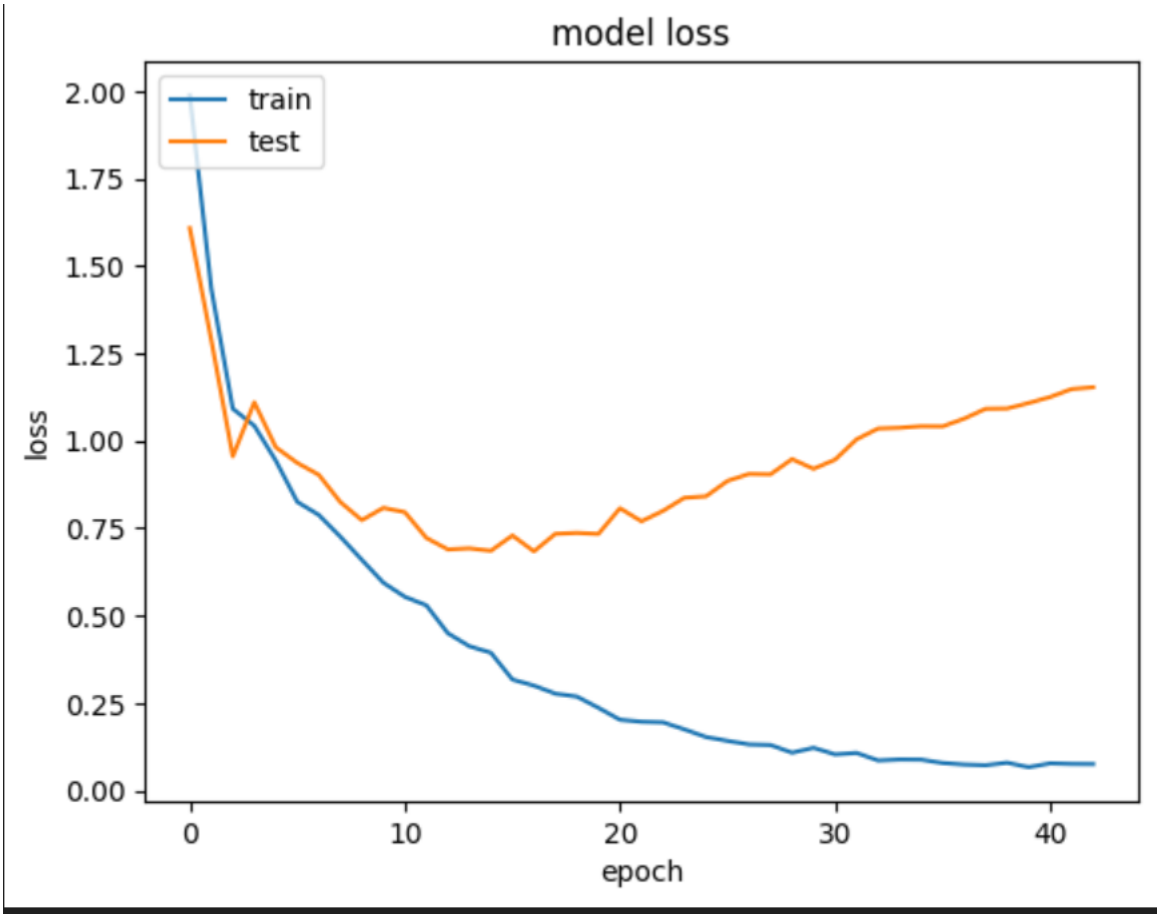
```python
scores_trainig = model.evaluate(X_train, y_train, verbose=1)
print("Training Loss: %f%%" % (scores_trainig[0]))
print("Training Accuracy: %.2f%%" % (scores_trainig[1]*100))
print("Training Precision: %.2f%%" % (scores_trainig[2]*100))
print("Training Recall: %.2f%%" % (scores_trainig[3]*100))
print("Training F1 Score: %.2f%%" % (scores_trainig[4]*100))
print("Training Cohen Kappa: %.2f%%" % (scores_trainig[5]*100))
```

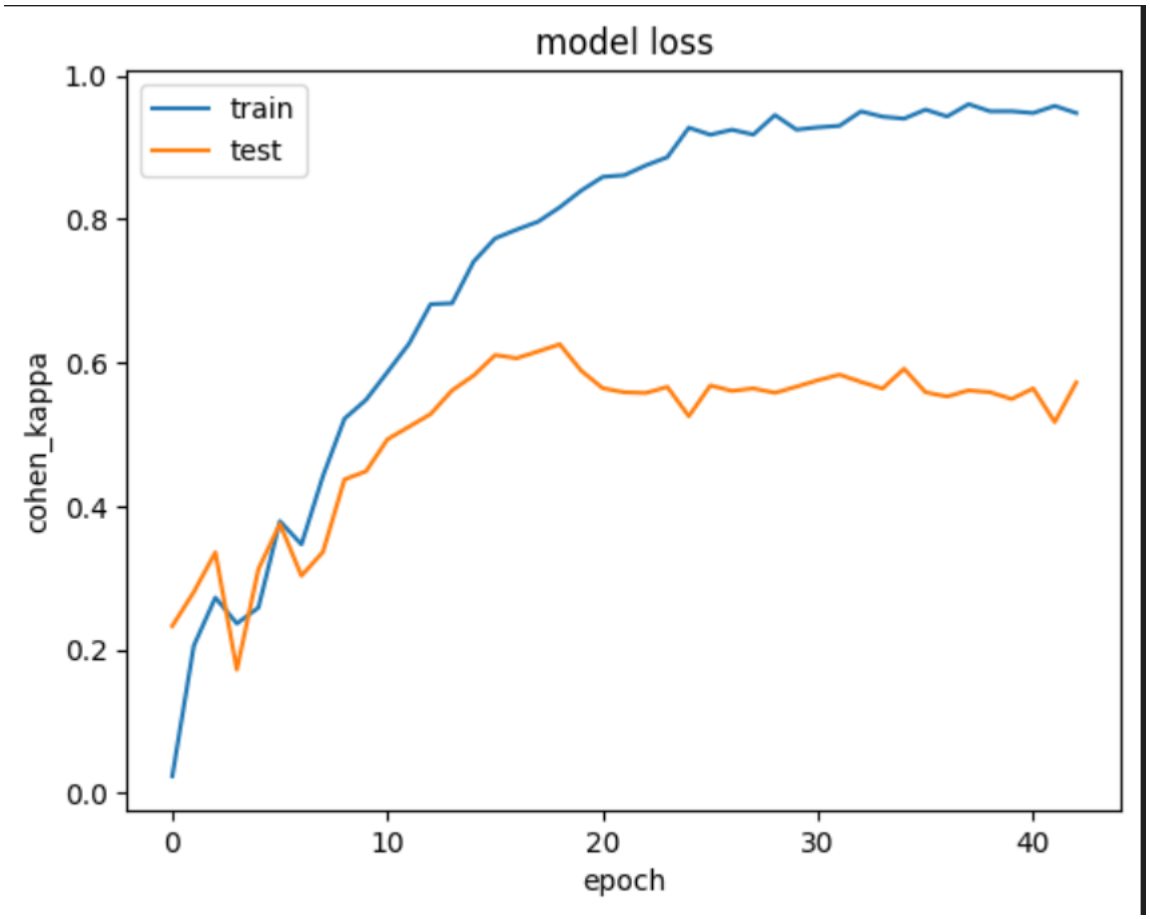
```
22/22 [==============================] - 1s 30ms/step - loss: 0.0577 - accuracy: 0.9784 - precision: 0.9784
Training Loss: 0.057733%
Training Accuracy: 97.84%
Training Precision: 97.84%
Training Recall: 97.70%
Training F1 Score: 97.79%
Training Cohen Kappa: 96.24%
```

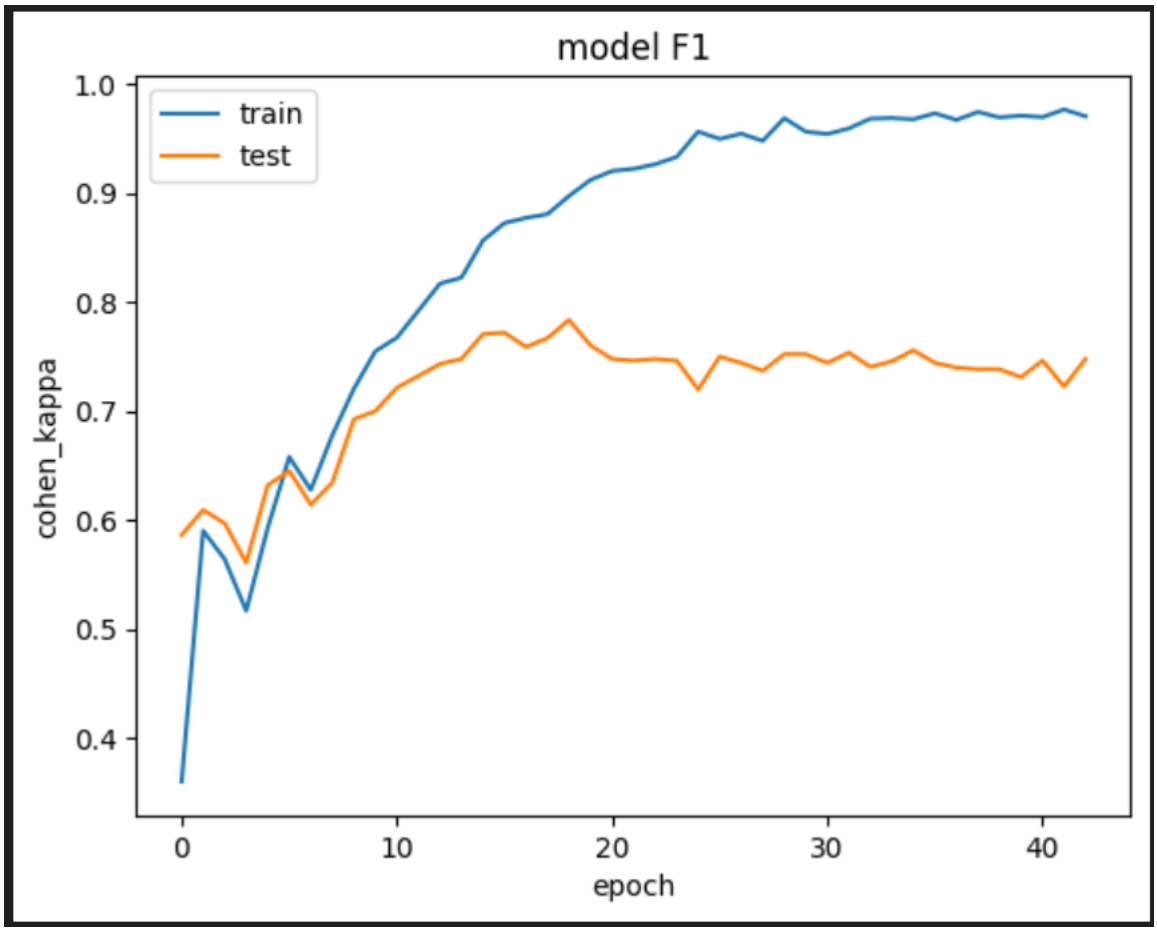
```python
scores_test = model.evaluate(X_test, y_test, verbose=1)
print("Test Loss: %f%%" % (scores_test[0]))
print("Test Accuracy: %.2f%%" % (scores_test[1]*100))
print("Test Precision: %.2f%%" % (scores_test[2]*100))
print("Test Recall: %.2f%%" % (scores_test[3]*100))
print("Test F1 Score: %.2f%%" % (scores_test[4]*100))
print("Test Cohen Kappa: %.2f%%" % (scores_test[5]*100))
```

```
6/6 [==============================] - 0s 21ms/step - loss: 1.1529 - accuracy: 0.7529 -
Test Loss: 1.152943%
Test Accuracy: 75.29%
Test Precision: 75.44%
Test Recall: 74.14%
Test F1 Score: 75.14%
Test Cohen Kappa: 57.23%
```


model accuracy
train
test
accuracy
epoch
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
10
20
30
40


model loss
train
test
loss
epoch
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
10
20
30
40


model loss
train
test
cohen_kappa
epoch
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40


model F1
train
test
cohen_kappa
epoch
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
10
20
30
40

Model	Train_Accuracy	Train_Loss	Test_Accuracy	Test_Loss
LSTM	93.97%	0.135574%	1.448241%	77.01%
Transformer	0.057733%	97.84%	1.152943%	75.29%

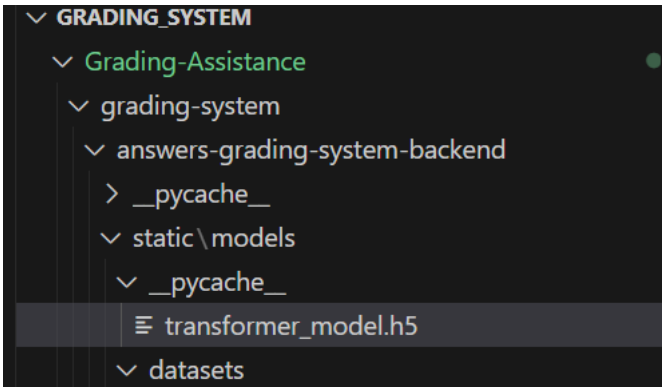
Deploying the model:

In this project, we used FastAPI to develop our application's backendand Angular for the frontend. We chose FastAPI because of its high performance and integrated support for GraphQL, which enabled us to set up a GraphQL API for communication between frontend and backend.

- **FAST.API**

```python
from fastapi import FastAPI
from strawberry.asgi import GraphQL
import strawberry
from fastapi.middleware.cors import CORSMiddleware
from typing import List
import models.ready_model as ready_model


app = FastAPI()
```

Importation of our Transformer model:



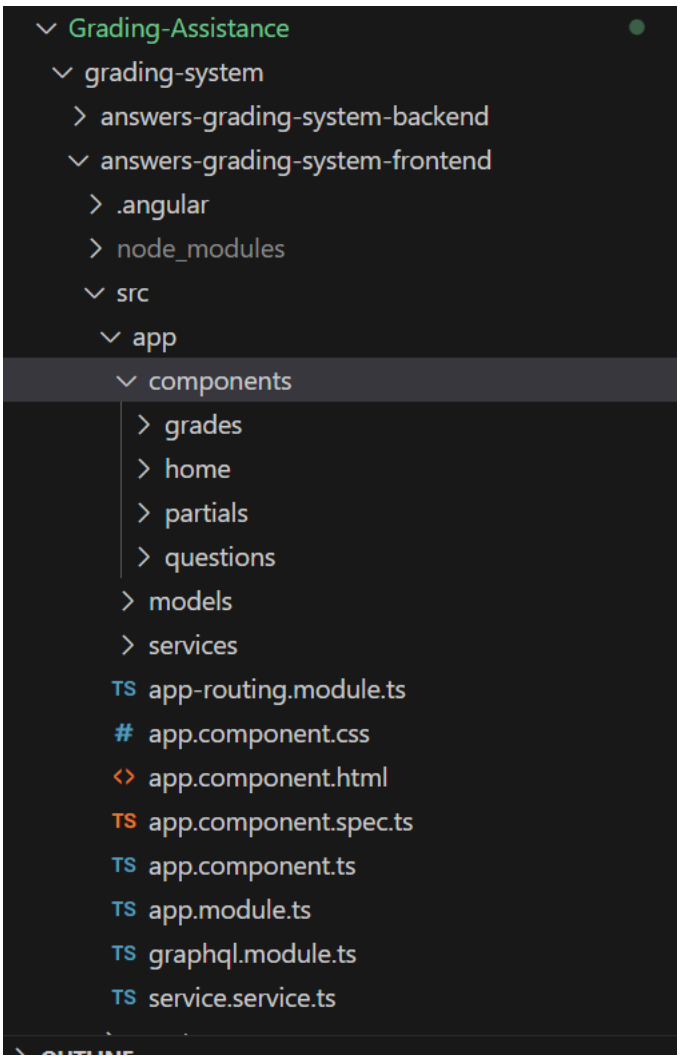
```python
def predict(input):
    model_path = (
        "./static/models/trained_models/transformer_model" + ".h5"
    )
    with open(model_path, "rb") as file:
        model, model_word2vec = pickle.load(file)
    input = preprocces_input(input, model_word2vec)
    input = input.reshape(1, -1)
    pred = model.predict(input)
    result = pred[0]
    return result
```

Predicting the score of each answer and returning it to the front :

```python
@strawberry.type
class Mutation:
    @strawberry.mutation
    def create_question(self, id: str, answer: str) -> List[Question]:
        if(id=="1"):
            questions.clear()
        score=ready_model.predict(answer, id)
        questions.append(Question(id=id, answer=answer, score=score))
        return questions
```

- ## Angular:

  The structure of the Angular:



  In App, four components are created: main page, questions page
  answers page and the navbar :

- ## GRAPHQL:

  To facilitate communication between the frontend and backend, we've GraphQL.
  FastApi integration (backend side):
  We used Strawberry, a Python library that makes it
  create GraphQL servers simply and intuitively.
  intuitive way. It offers declarative syntax for defining GraphQL schemas
  GraphQL schemas, data types, queries and resolvers.

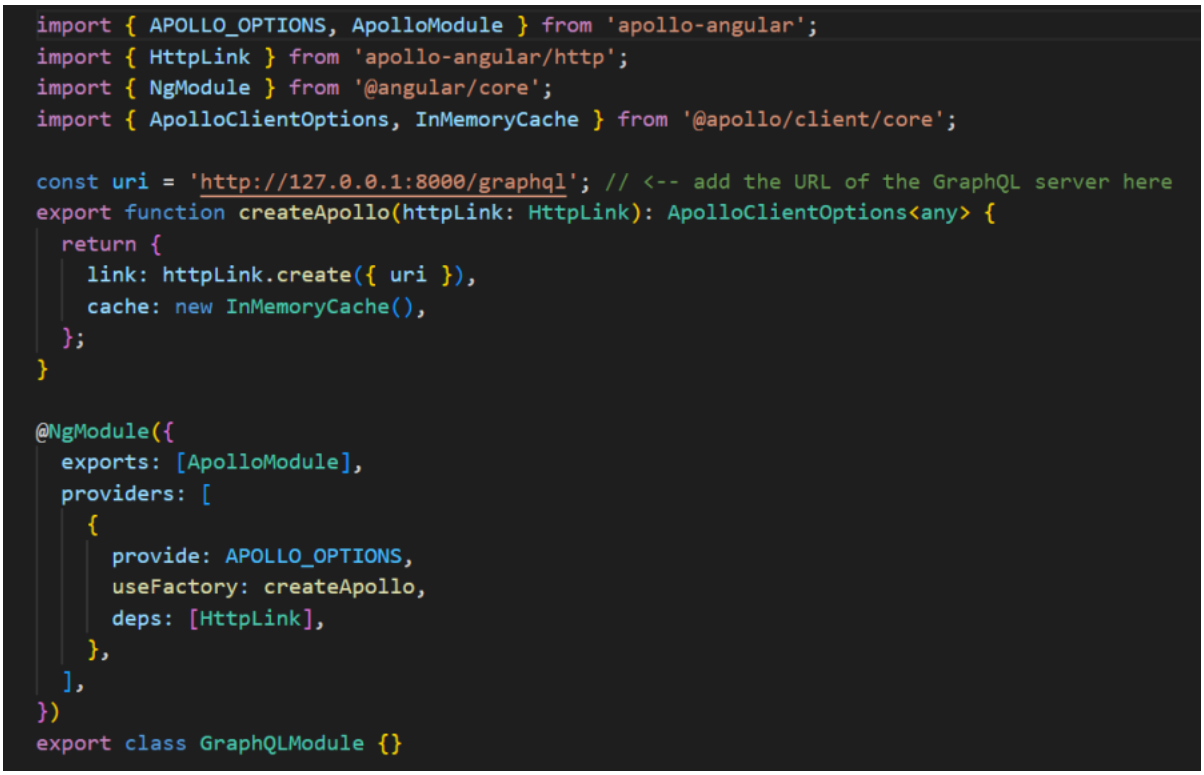
```python
@strawberry.type
class Question:
    id: str
    answer: str
    score: int

questions: List[Question] = [ ]

@strawberry.type
class Query:
    @strawberry.field
    def questions(self) -> List[Question]:
        return questions

@strawberry.type
class Mutation:
    @strawberry.mutation
    def create_question(self, id: str, answer: str) -> List[Question]:
        if(id=="1"):
            questions.clear()
        score=ready_model.predict(answer, id)
        questions.append(Question(id=id, answer=answer, score=score))
        return questions
```

```python
schema = strawberry.Schema(query=Query, mutation=Mutation)

graphql_app = GraphQL(schema)
app.add_route("/graphql", graphql_app)
app.add_websocket_route("/graphql", graphql_app)
```

Angular integration (frontend) :

graphql.module:

Apollo facilitates GraphQL query execution, cache management, subscribe to real-time updates and much more. It also provides tools and components to facilitate integrate Apollo into your Angular application, enabling you to quicklydevelop features based on GraphQL-based functionality.



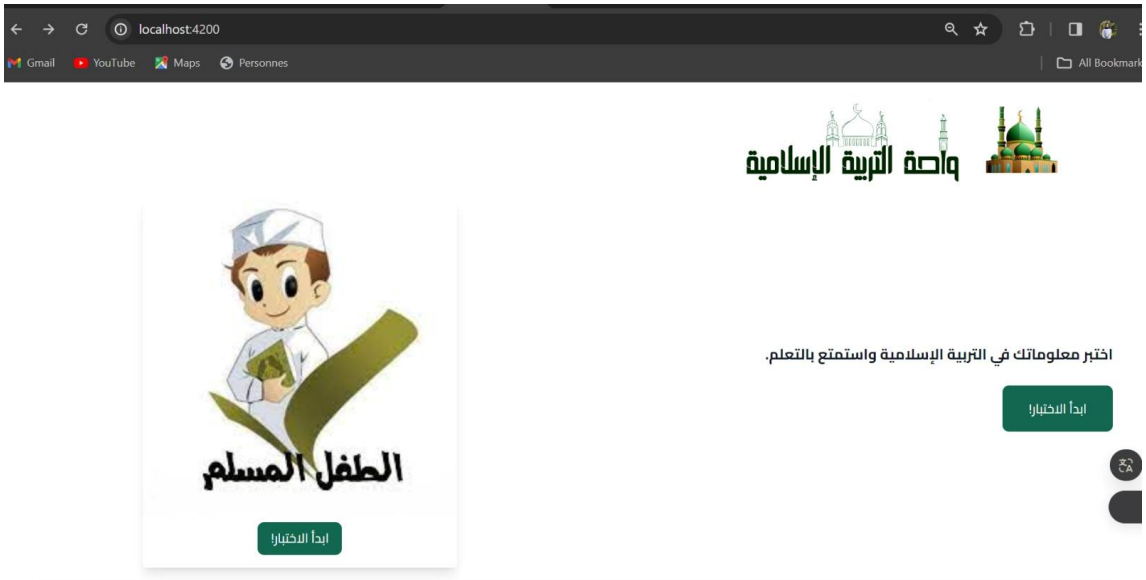
```
import { APOLLO_OPTIONS, ApolloModule } from 'apollo-angular';
import { HttpLink } from 'apollo-angular/http';
import { NgModule } from '@angular/core';
import { ApolloClientOptions, InMemoryCache } from '@apollo/client/core';

const uri = 'http://127.0.0.1:8000/graphql'; // <-- add the URL of the GraphQL server here
export function createApollo(httpLink: HttpLink): ApolloClientOptions<any> {
  return {
    link: httpLink.create({ uri }),
    cache: new InMemoryCache(),
  };
}

@NgModule({
  exports: [ApolloModule],
  providers: [
    {
      provide: APOLLO_OPTIONS,
      useFactory: createApollo,
      deps: [HttpLink],
    },
  ],
})
export class GraphQLModule {}
```

Graphql.operations :

The GET_ANSWERS and CREATE_QUESTION constants contain the to retrieve answers from existing questions and to create a new create a new question, respectively.

```
import { gql } from 'apollo-angular'

const GET_ANSWERS=gql`

query {
  questions {
    id
    answer
    score
  }
}
    `

const CREATE_QUESTION = gql`
mutation($id: String!, $answer: String!) {
  createQuestion(id: $id, answer: $answer) {
    id
    answer
    score
  }
}
`
export { GET_ANSWERS, CREATE_QUESTION}
```

Application:



واحة التربية الإسلامية

اختبر معلوماتك في التربية الإسلامية واستمتع بالتعلم.

السؤال 1 : من هو خاتم الأنبياء والمرسلين؟

السؤال 2 : كم استمرت الدعوة السرية؟

السؤال 3 : ما هي السور التي بدأت بالحمد ؟

السؤال 4 : ما هو اسم والدة نبي الله عيسى عليه السلام؟

السؤال 4 : ما هو اسم والدة نبي الله عيسى عليه السلام؟

السؤال 5 : من هي أول من دخل في الإسلام من النساء؟

السؤال 6 : ما هي السورة التي لا تبدأ بالبسملة ؟

السؤال 7 : ما هو اسم مرضعة الرسول صلى الله عليه وسلم؟

السؤال 8 : من هو النبي الذي التقمه الحوت؟

السؤال 9 : ما هـ... السورة التي تحدثت عن تقسيم المواريث؟

اختبر معلوماتك في التربية الإسلامية واستمتع بالتعلم.

السؤال 1 : من هو خاتم الأنبياء والمرسلين؟

محمد صلى الله عليه وسلم

السؤال 2 : كم استمرت الدعوة السرية؟

للدعوة السرية

السؤال 3 : ما هي السور التي بدأت بالحمد ؟

السور التي تفتتح بالحمد هي: الفاتحة، الأنعام، الكهف، سبأ، وفاطر.

السؤال 4 : ما هو اسم والدة نبي الله عيسى عليه السلام؟

اسم والدة نبي الله عيسى هو ياسمين، وكانت ملكة في ذلك الوقت

السؤال 5 : من هي أول من دخل في الإسلام من النساء؟

خديجة بنت خويلد

السؤال 6 : ما هي السورة التي لا تبتدأ بالبسملة ؟

البسملة غير موجودة في بداية سورة التوبة

السؤال 7 : ما هو اسم مرضعة الرسول صلى الله عليه وسلم؟

حليمة السعدية رضي الله عنها توفيت في عهد عمر بن الخطاب رضي الله عنه

السؤال 8 : من هو النبي الذي التقمه الحوت؟

لا أعرف أي نبي ابتلعه الحوت

السؤال 9 : ما هي السورة التي تحدثت عن تقسيم الغنائم؟

عندما انتصر المسلمون في غزوة بدر، واجهتهم مهمة أخرى: كيف يوزعون الغنائم التي حصلوا عليها؟ لم يتركهم الإسلام في حيرة، فقد نزلت سورة الأنفال تحمل تشريعًا دقيقًا لتقسيم الفيء بأمانة وعدل.

السؤال 10 : ما هي أقصر سورة في القرآن الكريم ؟

سورة الكوثر هي أقصر سورة في القرآن الكريم

عرض النتيجة