```
import numpy as np
import pandas as pd
import nltk
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow import keras
from tensorflow.keras import layers
```

```
# Define the path to your zip file
zip_file_path = "/content/mal-api-2019.zip"

# Define the directory where you want to extract the contents
extracted_dir_path = "/content/mal-api"

# Unzip the file
!unzip "$zip_file_path" -d "$extracted_dir_path"
```

```
Archive:  /content/mal-api-2019.zip
  inflating: /content/mal-api/all_analysis_data.txt
```

```
df=pd.read_csv('/content/mal-api/all_analysis_data.txt')
```

```
# Rename the first column to 'api'
df = df.rename(columns={df.columns[0]: 'api'})

# Print the DataFrame to verify the changes
```

```
df.head()
```

	api
0	getsystemtimeasfiletime ntallocatevirtualmemor...
1	ldrgetdllhandle ldrgetprocedureaddress getsyst...
2	ldrloaddll ldrgetprocedureaddress ldrloaddll l...
3	ldrloaddll ldrgetprocedureaddress ldrgetproced...
4	ntprotectvirtualmemory ntprotectvirtualmemory ...

```
labels=pd.read_csv('/content/labels (1).csv')
```

```
labels.head()
```

	Trojan
0	Trojan
1	Backdoor
2	Backdoor
3	Trojan
4	Trojan

```
# Count unique words
from collections import Counter
def counter_word(text_col):
    count = Counter()
    for text in text_col.values:
        for word in text.split():
            count[word] += 1
    return count
```

```
counter = counter_word(df.api)
len(counter)
```

```
278
```

```
malware_mapping = {
    'Trojan': 1,
    'Backdoor': 2,
    'Downloader': 3,
    'Worms': 4,
    'Spyware': 5,
    'Adware': 6,
    'Dropper': 7,
    'Virus':8
}


labels['Trojan'] =labels['Trojan'].replace(malware_mapping)

# Print the DataFrame to verify the changes
print(labels)
```

```
      Trojan
0          1
1          2
2          2
3          1
4          1
...      ...
7101       8
7102       8
7103       8
7104       8
7105       8

[7106 rows x 1 columns]
```

```
Start coding or generate with AI.
```

```
X = df.api
y = labels.Trojan
# Split data into training and test sets
train_sentences, test_sentences, train_labels,test_labels = train_test_split(X, y, test_size=0.2, random_state=42)

# Further split training data into training and validation sets
train_sentences,val_sentences,train_labels,val_labels = train_test_split(train_sentences,train_labels, test_size=0.2, random

#train/val
train_sentences = train_sentences.to_numpy()
train_labels = train_labels.to_numpy()
val_sentences = val_sentences.to_numpy()
val_labels = val_labels.to_numpy()
test_sentences=test_sentences.to_numpy()
test_labels=test_labels.to_numpy()


train_sentences.shape, val_sentences.shape
```

```
((4547,), (1137,))
```

```
# Tokenize
# vectorize a text corpus by turning each text into a sequence of integers
#counter=278
num_unique_words=278 #len(counter)

tokenizer = Tokenizer(num_words=num_unique_words)
tokenizer.fit_on_texts(train_sentences) # fit only to training


api_index = tokenizer.word_index
api_index
```

```
{'getasynckeystate': 1,
 'ntdelayexecution': 2,
 'ntclose': 3,
 'ntreadfile': 4,
 'findfirstfileexw': 5,
 'process32nextw': 6,
 'ntallocatevirtualmemory': 7,
 'ntcreatefile': 8,
 'ntquerydirectoryfile': 9,
 'getsystemmetrics': 10,
 'copyfilea': 11,
 'exception': 12,
 'ldrgetprocedureaddress': 13,
 'ntwritefile': 14,
 'getkeystate': 15,
 'findwindowa': 16,
```

```
 'ldrgetdllhandle': 17,
 'deviceiocontrol': 18,
 'setfilepointer': 19,
 'ntfreevirtualmemory': 20,
 'getfileattributesw': 21,
 'deletefilew': 22,
 'readprocessmemory': 23,
 'gethostbyname': 24,
 'seterrormode': 25,
 'regclosekey': 26,
 'regopenkeyexw': 27,
 'findresourcea': 28,
 'ntopenfile': 29,
 'openscmanagera': 30,
 'getfilesize': 31,
 'getcursorpos': 32,
 'regqueryvalueexw': 33,
 'ntopenprocess': 34,
 'setfileattributesw': 35,
 'getforegroundwindow': 36,
 'ldrloaddll': 37,
 'socket': 38,
 'closesocket': 39,
 'shellexecuteexw': 40,
 'ntopenkeyex': 41,
 'regsetvalueexa': 42,
 'connect': 43,
 'ntquerykey': 44,
 'regenumkeyw': 45,
 'regqueryvalueexa': 46,
 'regopenkeyexa': 47,
 'ioctlsocket': 48,
 'loadstringa': 49,
 'regenumkeyexa': 50,
 'outputdebugstringa': 51,
 'select': 52,
 'cryptacquirecontexta': 53,
 'regenumvaluea': 54,
 'ntopenkey': 55,
 'ntprotectvirtualmemory': 56,
 'crypthashdata': 57,
 'ntqueryvaluekey': 58,
```

```
#apply on train, validation, and test sentences

train_sequences = tokenizer.texts_to_sequences(train_sentences)
val_sequences = tokenizer.texts_to_sequences(val_sentences)
test_sequences=tokenizer.texts_to_sequences(test_sentences)
```

```
train_sentences[0]
```

```
'ntallocatevirtualmemory seterrormode loadstringa oleinitialize ldrloaddll ld
rgetprocedureaddress ntallocatevirtualmemory ntallocatevirtualmemory ntalloca
tevirtualmemory getsysteminfo ntallocatevirtualmemory ntallocatevirtualmemory
ntallocatevirtualmemory ntallocatevirtualmemory ntallocatevirtualmemory ntall
ocatevirtualmemory ntallocatevirtualmemory ntprotectvirtualmemory ldrgetdllha
ndle ldrgetprocedureaddress ldrgetprocedureaddress ldrgetprocedureaddress ldr
getprocedureaddress ldrgetprocedureaddress ldrgetprocedureaddress ldrgetproce
dureaddress ldrgetprocedureaddress ldrgetprocedureaddress ldrgetprocedureaddr
```

```
train_sequences[0]
```

```
[7,
 26,
 48,
 172,
 62,
 23,
 7,
 7,
 7,
 121,
 7,
 7,
 7,
 7,
 7,
 7,
 7,
 82,
 12,
 23,
 23,
 23,
 23,
 23,
 23,
 23,
```

```
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
        23,
```

```
#Check
print(train_sentences[10:15])
```

```
['getcursorpos seterrormode seterrormode findresourcea findresourcea setwindowshookexa findresourcea findresourceexw loa
 'getcursorpos seterrormode seterrormode findresourcea findresourcea setwindowshookexa ntcreatemutant findresourcea find
 'ntallocatevirtualmemory ntallocatevirtualmemory ntallocatevirtualmemory ntfreevirtualmemory ntallocatevirtualmemory nt
 'ntallocatevirtualmemory ntfreevirtualmemory ldrloaddll ldrgetprocedureaddress ldrgetprocedureaddress ldrgetproceduread
 'ntallocatevirtualmemory ntfreevirtualmemory ntallocatevirtualmemory ntallocatevirtualmemory ntallocatevirtualmemory nt
```

```
# Pad the sequences to have the same length
#max_length = 1764422 #arbitrary number
max_length = int(50000)

train_padded = pad_sequences(train_sequences, maxlen=max_length, padding="post", truncating="post") #post-> 0
val_padded = pad_sequences(val_sequences, maxlen=max_length, padding="post", truncating="post")
test_padded = pad_sequences(test_sequences, maxlen=max_length, padding="post", truncating="post")


def data_generator(data, labels, batch_size):
    num_samples = len(data)
    while True:
        # Shuffle the data and labels
        indices = np.random.permutation(num_samples)
        shuffled_data = data[indices]
        shuffled_labels = labels[indices]

        # Generate batches
        for i in range(0, num_samples, batch_size):
            batch_data = shuffled_data[i:i+batch_size]
            batch_labels = shuffled_labels[i:i+batch_size]
            yield batch_data, batch_labels


batch_size = 32
train_generator = data_generator(train_padded, train_labels, batch_size)
val_generator = data_generator(val_padded, val_labels, batch_size)


#Check
train_padded.shape, val_padded.shape
train_padded[0]


train_sentences[10]


train_sequences[10]


train_padded[10]
```

```
train_padded[3]


# Embedding: Turns positive integers (indexes) into dense vectors of fixed size.

model = keras.models.Sequential()
model.add(layers.Embedding(num_unique_words, 100, input_length=max_length))

model.add(layers.LSTM(32, dropout=0.25))
model.add(layers.Dense(1, activation="sigmoid"))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 50000, 100)        27800

 lstm (LSTM)                 (None, 32)                17024

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 44857 (175.22 KB)
Trainable params: 44857 (175.22 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
loss = keras.losses.BinaryCrossentropy(from_logits=False)
optim = keras.optimizers.Adam(learning_rate=0.001)
metrics = ["accuracy"]

model.compile(loss=loss, optimizer=optim, metrics=metrics)


model.fit(train_generator, epochs=15, steps_per_epoch=len(train_padded)//batch_size,
          validation_data=val_generator, validation_steps=len(val_padded)//batch_size)
```

```
Epoch 1/15
142/142 [==============================] - 269s 2s/step - loss: -23.8107 - accuracy: 0.1334 - val_loss: -38.2680 - val_a
Epoch 2/15
142/142 [==============================] - 255s 2s/step - loss: -48.9179 - accuracy: 0.1338 - val_loss: -55.5830 - val_a
Epoch 3/15
142/142 [==============================] - 237s 2s/step - loss: -65.9098 - accuracy: 0.1333 - val_loss: -72.4774 - val_a
Epoch 4/15
142/142 [==============================] - 254s 2s/step - loss: -82.7718 - accuracy: 0.1329 - val_loss: -87.5461 - val_a
Epoch 5/15
142/142 [==============================] - 255s 2s/step - loss: -98.8244 - accuracy: 0.1340 - val_loss: -101.1984 - val_
Epoch 6/15
142/142 [==============================] - 253s 2s/step - loss: -114.4871 - accuracy: 0.1336 - val_loss: -120.1660 - val
Epoch 7/15
142/142 [==============================] - 256s 2s/step - loss: -131.2615 - accuracy: 0.1324 - val_loss: -132.8019 - val
Epoch 8/15
142/142 [==============================] - 256s 2s/step - loss: -145.9133 - accuracy: 0.1355 - val_loss: -150.3377 - val
Epoch 9/15
142/142 [==============================] - 254s 2s/step - loss: -162.4021 - accuracy: 0.1311 - val_loss: -167.6780 - val
Epoch 10/15
142/142 [==============================] - 234s 2s/step - loss: -179.3092 - accuracy: 0.1329 - val_loss: -178.0516 - val
Epoch 11/15
142/142 [==============================] - 237s 2s/step - loss: -193.2397 - accuracy: 0.1347 - val_loss: -193.3174 - val
Epoch 12/15
 29/142 [=====>........................] - ETA: 2:43 - loss: -204.3290 - accuracy: 0.1279
```

```
import tensorflow as tf

# Assume 'model' is your trained model
model.save('/content/best_model.h5')


model.fit(train_padded, train_labels, epochs=25, validation_data=(val_padded, val_labels), verbose=2)


import pickle

# Assume tokenizer is your tokenizer object
with open('tokenizer.pickle', 'wb') as handle:
    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)



prediction=model.predict(test_padded)
```