In [1]:

```
from tensorflow.keras import layers
from tensorflow import keras
import tensorflow as tf

from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from ast import literal_eval
import pandas as pd
import numpy as np

import keras
from keras import optimizers
from keras import backend as K
from keras import regularizers
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, Flatten
from keras.layers import Embedding, Conv1D, MaxPooling1D, GlobalMaxPooling1D
#from keras.utils import plot_model
from keras.preprocessing import sequence
from keras.preprocessing.text import Tokenizer
from keras.callbacks import EarlyStopping
from tqdm import tqdm
from nltk.corpus import stopwords
from nltk.tokenize import RegexpTokenizer
import os, re, csv, math, codecs
import fasttext as ft
from nltk.tokenize import WhitespaceTokenizer
import pickle
from tensorflow.keras.utils import to_categorical
```

```
2022-10-28 18:53:40.696657: W tensorflow/stream_executor/platform/defa
ult/dso_loader.cc:64] Could not load dynamic library 'libcudart.so.11.
0'; dlerror: libcudart.so.11.0: cannot open shared object file: No suc
h file or directory
2022-10-28 18:53:40.696672: I tensorflow/stream_executor/cuda/cudart_s
tub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up
on your machine.
```

In [ ]:

```
```

In [2]:

```
data=pd.read_csv('data.csv',header=None)
```

In [3]:

```
data = data.sample(frac=1)
```

In [4]:

```
print(f"There are {len(data)} rows in the dataset.")
```

```
There are 82657 rows in the dataset.
```

In [5]:

```
data.shape
```

Out[5]:

```
(82657, 11)
```

In [6]:

```
data.head(1)
```

Out[6]:

	0	1	2	3	4	5	6
12478	239356	http://arxiv.org/abs/2207.01511v2	Uniting Machine Intelligence, Brain and Behavi...	I discuss here three important roles where mac...	NaN	2022-06-30 16:34:07+00:00	cs.CY

In [7]:

```
test_split = 0.2

# Initial train and test split.
train_df, test_df = train_test_split(data,test_size=test_split)

print(f"Number of rows in training set: {len(train_df)}")
print(f"Number of rows in test set: {len(test_df)}")
```

```
Number of rows in training set: 66125
Number of rows in test set: 16532
```

In [8]:

```
train_df.head(1)
```

Out[8]:

	0	1	2	3	4	5	
22631	41299	http://arxiv.org/abs/2209.07162v1	Brain Imaging Generation with Latent Diffusion...	Deep neural networks have brought remarkable b...	NaN	2022-09-15 09:16:21+00:00	eess.

In [9]:

```
#processing label of training/testing data
label_train=train_df[10].values
label_test=test_df[10].values
print(label_test)
catg=['physics', 'cs', 'q-bio', 'math', 'eess']
num_classes = len(catg)
mapping = {}
for x in range(len(catg)):
    mapping[catg[x]] = x

# integer representation
for x in range(len(label_train)):
    label_train[x] = mapping[label_train[x]]
#processing lebal of testing data

# integer representation
for x in range(len(label_test)):
    label_test[x] = mapping[label_test[x]]

#conveting to one-hot-encoding
y_train = to_categorical(label_train)
y_test=to_categorical(label_test)
```

```
['physics' 'physics' 'math' ... 'physics' 'q-bio' 'eess']
```

In [10]:

```
train_df[3].apply(lambda x: len(x.split(" "))).describe()
```

Out[10]:

```
count    66125.000000
mean       163.348673
std         59.216578
min          6.000000
25%        121.000000
50%        162.000000
75%        203.000000
max        530.000000
Name: 3, dtype: float64
```

In [ ]:

```
max_seqlen = 150
batch_size = 128
padding_token = "<pad>"
```

In [12]:

```
#loading embedding
ft_model = ft.load_model("cc.en.300.bin")
```

In [13]:

```
#pre_processing train/test data
MAX_NB_WORDS = 100000
# tokenizer = RegexpTokenizer(r'\w+')
tokenizer = WhitespaceTokenizer()

raw_docs_train = train_df[3].tolist()
raw_docs_test = test_df[3].tolist()
```

In [14]:

```
print("pre-processing train data...")
processed_docs_train = []
for line in tqdm(raw_docs_train):
    #line=text_cleaner(line)
    tokens = tokenizer.tokenize(line)
    #filtered = [word for word in tokens if word not in stop_words]
    processed_docs_train.append(" ".join(tokens))
#end for
```

```
pre-processing train data...

100%|██████████| 66125/66125 [00:02<00:00, 27753.12it/s]
```

In [15]:

```
print("pre-processing test data...")
processed_docs_test = []
for line in tqdm(raw_docs_test):
    #line=text_cleaner(line)
    tokens = tokenizer.tokenize(line)
    #filtered = [word for word in tokens if word not in stop_words]
    processed_docs_test.append(" ".join(tokens))
```

```
pre-processing test data...

100%|██████████| 16532/16532 [00:00<00:00, 27872.34it/s]
```

In [16]:

```
max_seq_len = 160
```

In [17]:

```
print("tokenizing input data...")
tokenizer = Tokenizer(num_words=MAX_NB_WORDS, lower=True, char_level=False)
tokenizer.fit_on_texts(processed_docs_train + processed_docs_test)  #leaky
word_seq_train = tokenizer.texts_to_sequences(processed_docs_train)
word_seq_test = tokenizer.texts_to_sequences(processed_docs_test)
word_index = tokenizer.word_index
print("dictionary size: ", len(word_index))
```

```
tokenizing input data...
dictionary size:  131631
```

In [19]:

```
from keras_preprocessing.sequence import pad_sequences
```

In [20]:

```
#pad sequences
word_seq_train = pad_sequences(word_seq_train, maxlen=max_seq_len)
word_seq_test = pad_sequences(word_seq_test, maxlen=max_seq_len)
```

In [21]:

```
#embedding matrix
embed_dim = 300
print('preparing embedding matrix...')
words_not_found = []
nb_words = min(MAX_NB_WORDS, len(word_index)+1)
embedding_matrix = np.zeros((nb_words, embed_dim))
for word, i in word_index.items():
    if i >= nb_words:
        continue
    embedding_vector = ft_model.get_word_vector(word)
    if (embedding_vector is not None) and len(embedding_vector) > 0:
        # words not found in embedding index will be all-zeros.
        embedding_matrix[i] = embedding_vector
    else:
        words_not_found.append(word)
print('number of null word embeddings: %d' % np.sum(np.sum(embedding_matrix, axis=1
```

```
preparing embedding matrix...
number of null word embeddings: 27
```

In [22]:

```
#model parameters
num_filters = 12
weight_decay = 1e-4
```

In [23]:

```
#CNN architecture
print("training CNN ...")
model = Sequential()
model.add(Embedding(nb_words, embed_dim,
          weights=[embedding_matrix], input_length=max_seq_len, trainable=False))
model.add(Conv1D(num_filters,  7, activation='relu', padding='same'))
model.add(MaxPooling1D(2))
model.add(Conv1D(num_filters, 7, activation='relu', padding='same'))
model.add(GlobalMaxPooling1D())
model.add(Dropout(0.5))
model.add(Dense(32, activation='relu', kernel_regularizer=regularizers.l2(weight_de
model.add(Dense(num_classes, activation='softmax'))  #multi-label (k-hot encoding)
adam = tf.optimizers.Adam()
model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'])
model.summary()
```

```
training CNN ...

2022-10-28 19:03:19.462837: W tensorflow/stream_executor/platform/defa
ult/dso_loader.cc:64] Could not load dynamic library 'libcuda.so.1'; d
lerror: libcuda.so.1: cannot open shared object file: No such file or
directory
2022-10-28 19:03:19.463569: W tensorflow/stream_executor/cuda/cuda_dri
ver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)
2022-10-28 19:03:19.479737: I tensorflow/stream_executor/cuda/cuda_dia
gnostics.cc:156] kernel driver does not appear to be running on this h
ost (saurabh-hp-prodesk): /proc/driver/nvidia/version does not exist
2022-10-28 19:03:19.590077: I tensorflow/core/platform/cpu_feature_gua
rd.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural
Network Library (oneDNN) to use the following CPU instructions in perf
ormance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the approp
riate compiler flags.
2022-10-28 19:03:20.163609: W tensorflow/core/framework/cpu_allocator_
impl.cc:82] Allocation of 120000000 exceeds 10% of free system memory.
2022-10-28 19:03:20.251171: W tensorflow/core/framework/cpu_allocator_
impl.cc:82] Allocation of 120000000 exceeds 10% of free system memory.
2022-10-28 19:03:21.712627: W tensorflow/core/framework/cpu_allocator_
impl.cc:82] Allocation of 120000000 exceeds 10% of free system memory.
2022-10-28 19:03:22.573761: W tensorflow/core/framework/cpu_allocator_
impl.cc:82] Allocation of 120000000 exceeds 10% of free system memory.

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 160, 300)          30000000

 conv1d (Conv1D)             (None, 160, 12)           25212

 max_pooling1d (MaxPooling1D  (None, 80, 12)           0
 )

 conv1d_1 (Conv1D)           (None, 80, 12)            1020

 global_max_pooling1d (Globa  (None, 12)               0
 lMaxPooling1D)

 dropout (Dropout)           (None, 12)                0
```

```
 dense (Dense)               (None, 32)                416

 dense_1 (Dense)             (None, 5)                 165

=================================================================
Total params: 30,026,813
Trainable params: 26,813
Non-trainable params: 30,000,000
_________________________________________________________________
```

In [24]:

```
#training params
batch_size = 256
num_epochs = 50
```

In [25]:

```
#model training
hist = model.fit(word_seq_train, y_train, batch_size=batch_size, epochs=num_epochs,
```

```
2022-10-28 19:03:29.596061: W tensorflow/core/framework/cpu_allocator_
impl.cc:82] Allocation of 38087680 exceeds 10% of free system memory.

Epoch 1/20
233/233 [==============================] - 21s 81ms/step - loss: 0.476
2 - accuracy: 0.3657 - val_loss: 0.3360 - val_accuracy: 0.6220
Epoch 2/20
233/233 [==============================] - 17s 71ms/step - loss: 0.345
1 - accuracy: 0.5845 - val_loss: 0.2784 - val_accuracy: 0.6713
Epoch 3/20
233/233 [==============================] - 17s 71ms/step - loss: 0.312
0 - accuracy: 0.6293 - val_loss: 0.2660 - val_accuracy: 0.6994
Epoch 4/20
233/233 [==============================] - 16s 70ms/step - loss: 0.300
1 - accuracy: 0.6488 - val_loss: 0.2606 - val_accuracy: 0.7085
Epoch 5/20
233/233 [==============================] - 16s 69ms/step - loss: 0.293
6 - accuracy: 0.6595 - val_loss: 0.2539 - val_accuracy: 0.7286
Epoch 6/20
233/233 [==============================] - 16s 69ms/step - loss: 0.285
4 - accuracy: 0.6713 - val_loss: 0.2515 - val_accuracy: 0.7340
Epoch 7/20
233/233 [==============================] - 16s 70ms/step - loss: 0.280
9 - accuracy: 0.6759 - val_loss: 0.2468 - val_accuracy: 0.7319
Epoch 8/20
233/233 [==============================] - 16s 68ms/step - loss: 0.275
7 - accuracy: 0.6830 - val_loss: 0.2429 - val_accuracy: 0.7372
Epoch 9/20
233/233 [==============================] - 16s 68ms/step - loss: 0.273
3 - accuracy: 0.6859 - val_loss: 0.2448 - val_accuracy: 0.7396
Epoch 10/20
233/233 [==============================] - 16s 69ms/step - loss: 0.269
8 - accuracy: 0.6940 - val_loss: 0.2436 - val_accuracy: 0.7505
Epoch 11/20
233/233 [==============================] - 16s 69ms/step - loss: 0.265
3 - accuracy: 0.6995 - val_loss: 0.2409 - val_accuracy: 0.7531
Epoch 12/20
233/233 [==============================] - 16s 68ms/step - loss: 0.262
5 - accuracy: 0.7044 - val_loss: 0.2466 - val_accuracy: 0.7540
Epoch 13/20
233/233 [==============================] - 16s 68ms/step - loss: 0.260
6 - accuracy: 0.7062 - val_loss: 0.2436 - val_accuracy: 0.7606
Epoch 14/20
233/233 [==============================] - 16s 70ms/step - loss: 0.257
7 - accuracy: 0.7086 - val_loss: 0.2443 - val_accuracy: 0.7641
Epoch 15/20
233/233 [==============================] - 16s 69ms/step - loss: 0.256
0 - accuracy: 0.7097 - val_loss: 0.2422 - val_accuracy: 0.7656
Epoch 16/20
233/233 [==============================] - 16s 69ms/step - loss: 0.253
5 - accuracy: 0.7134 - val_loss: 0.2363 - val_accuracy: 0.7673
Epoch 17/20
233/233 [==============================] - 16s 69ms/step - loss: 0.251
0 - accuracy: 0.7170 - val_loss: 0.2356 - val_accuracy: 0.7706
Epoch 18/20
```

```
233/233 [==============================] - 16s 68ms/step - loss: 0.251
1 - accuracy: 0.7170 - val_loss: 0.2382 - val_accuracy: 0.7635
Epoch 19/20
233/233 [==============================] - 16s 69ms/step - loss: 0.246
5 - accuracy: 0.7230 - val_loss: 0.2426 - val_accuracy: 0.7664
Epoch 20/20
233/233 [==============================] - 16s 69ms/step - loss: 0.245
8 - accuracy: 0.7256 - val_loss: 0.2385 - val_accuracy: 0.7680
```

In [26]:

```
y_test_result= model.predict(word_seq_test)
#print(y_test_result)
```

```
517/517 [==============================] - 1s 2ms/step
```

In [27]:

```
# evaluate the model
loss, accuracy = model.evaluate(word_seq_test, y_test, verbose=1)
print('Accuracy: %f' % (accuracy*100))
```

```
517/517 [==============================] - 1s 3ms/step - loss: 0.2423
- accuracy: 0.7650
Accuracy: 76.500124
```

In [28]:

```
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D, Bidirectional, D
```

In [29]:

```
#LSTM
lstm_out = 128
model_LSTM = keras.Sequential()
model_LSTM.add(Embedding(nb_words, embed_dim,
          weights=[embedding_matrix], input_length=max_seq_len, trainable=False))
model_LSTM.add(Bidirectional(LSTM(lstm_out, dropout=0.2)))
model_LSTM.add(Dense(128, activation = 'relu'))
model_LSTM.add(Dropout(0.5))
model_LSTM.add(Dense(64, activation = 'relu'))
model_LSTM.add(Dense(num_classes, activation='softmax'))  #multi-label (k-hot encod
adam = tf.optimizers.Adam()
model_LSTM.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy']
model_LSTM.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_1 (Embedding)      (None, 160, 300)          30000000

 bidirectional (Bidirectiona  (None, 256)              439296
 l)

 dense_2 (Dense)              (None, 128)               32896

 dropout_1 (Dropout)          (None, 128)               0

 dense_3 (Dense)              (None, 64)                8256

 dense_4 (Dense)              (None, 5)                 325

=================================================================
Total params: 30,480,773
Trainable params: 480,773
Non-trainable params: 30,000,000
_________________________________________________________________
```

In [31]:

```
#training params
batch_size_LSTM = 256
num_epochs_LSTM = 50
```

In [32]:

```
/_train, batch_size=batch_size_LSTM, epochs=num_epochs_LSTM,validation_split=0.1, sh
```

```
0.0559 - accuracy: 0.9453 - val_loss: 0.2697 - val_accuracy: 0.8191
Epoch 45/50
233/233 [==============================] - 88s 378ms/step - loss:
0.0522 - accuracy: 0.9495 - val_loss: 0.3089 - val_accuracy: 0.8281
Epoch 46/50
233/233 [==============================] - 88s 378ms/step - loss:
0.0520 - accuracy: 0.9495 - val_loss: 0.2757 - val_accuracy: 0.8083
Epoch 47/50
233/233 [==============================] - 88s 378ms/step - loss:
0.0496 - accuracy: 0.9522 - val_loss: 0.2906 - val_accuracy: 0.8207
Epoch 48/50
233/233 [==============================] - 88s 378ms/step - loss:
0.0470 - accuracy: 0.9539 - val_loss: 0.3015 - val_accuracy: 0.8172
Epoch 49/50
233/233 [==============================] - 88s 378ms/step - loss:
0.0457 - accuracy: 0.9561 - val_loss: 0.2898 - val_accuracy: 0.8175
Epoch 50/50
233/233 [==============================] - 88s 378ms/step - loss:
0.0436 - accuracy: 0.9581 - val_loss: 0.3167 - val_accuracy: 0.8098
```

In [ ]: