In [1]:
```
import os
import json
import torch
import numpy as np
import pandas as pd
import transformers
import seaborn as sns
import torch.nn as nn
from io import BytesIO
import plotly.express as px
import torch.optim as optim
from tqdm.notebook import tqdm
import torch.nn.functional as F
from more_itertools import take
import matplotlib.pyplot as plt
from urllib.request import urlopen
from sklearn.decomposition import PCA
from torch_models.mdl1 import Model1
from torch_models.tc_glove import Model2
from torch_utils.tc_utils import TC_UTILS
from torch_utils.tc_utils import lb_encoder
from torch_utils.tc_utils import ct_tokenizer
from torch.utils.data import Dataset, DataLoader
from torch_models.tc_baseline import tc_baseline
from torch_utils.tc_utils import CustomDataSetManger
from gensim.scripts.glove2word2vec import glove2word2vec
from transformers import BertForSequenceClassification, AdamW, get_linear_schedule_with_warmup
from transformers import DistilBertTokenizer, RobertaTokenizer, DistilBertTokenizerFast, DistilBertConfig, T
```

```
2022-11-13 14:00:40.619632: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is
optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in perfor
mance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2022-11-13 14:00:40.784299: E tensorflow/stream_executor/cuda/cuda_blas.cc:2981] Unable to register cuBLAS
factory: Attempting to register factory for plugin cuBLAS when one has already been registered
2022-11-13 14:00:41.361416: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load
dynamic library 'libnvinfer.so.7'; dlerror: libnvinfer.so.7: cannot open shared object file: No such file
or directory
2022-11-13 14:00:41.361491: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load
```

```
dynamic library 'libnvinfer_plugin.so.7'; dlerror: libnvinfer_plugin.so.7: cannot open shared object file:
No such file or directory
2022-11-13 14:00:41.361499: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Cannot
dlopen some TensorRT libraries. If you would like to use Nvidia GPU with TensorRT, please make sure the mi
ssing libraries mentioned above are installed properly.
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/markins/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [2]:
```
_plat = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print('Available system architecture for the models : {}'.format(_plat))
btch_sz, mxlen = 64, 128
```

```
Available system architecture for the models : cuda
```

In [3]:
```
# Loading up the dataset
files = ['../Dataset/pubmed-rct/PubMed_20k_RCT_numbers_replaced_with_at_sign/' + file for file in os.listdi
files
```

Out[3]:
```
['../Dataset/pubmed-rct/PubMed_20k_RCT_numbers_replaced_with_at_sign/dev.txt',
 '../Dataset/pubmed-rct/PubMed_20k_RCT_numbers_replaced_with_at_sign/test.txt',
 '../Dataset/pubmed-rct/PubMed_20k_RCT_numbers_replaced_with_at_sign/train.txt']
```

In [4]:
```
TC_UTILS_MANAGER_CLASS = TC_UTILS()
tr_lines = TC_UTILS_MANAGER_CLASS.render_lines(files[2])
tr_lines[:5]
print(f"The number of lines to be trained is {len(tr_lines)}")
```

```
The number of lines to be trained is 210040
```

In [5]:
```
train_smp = TC_UTILS_MANAGER_CLASS.pre_processor(files[2])
test_smp = TC_UTILS_MANAGER_CLASS.pre_processor(files[1])
val_smp = TC_UTILS_MANAGER_CLASS.pre_processor(files[0])

print("| Samples || Types      |",end="\n")
print(f"| {len(test_smp)}   || Testing    |",end="\n")
print(f"| {len(val_smp)}   || Validation |", end="\n")
print(f"| {len(train_smp)}  || Training   |",end="\n")
```

```
| Samples || Types      |
| 30135   || Testing    |
| 30212   || Validation |
| 180040  || Training   |
```

In [6]:
```
tr_df,ts_df,vl_df = pd.DataFrame(train_smp), pd.DataFrame(test_smp), pd.DataFrame(val_smp)
tr_df.head(),ts_df.head(),vl_df.head()
```

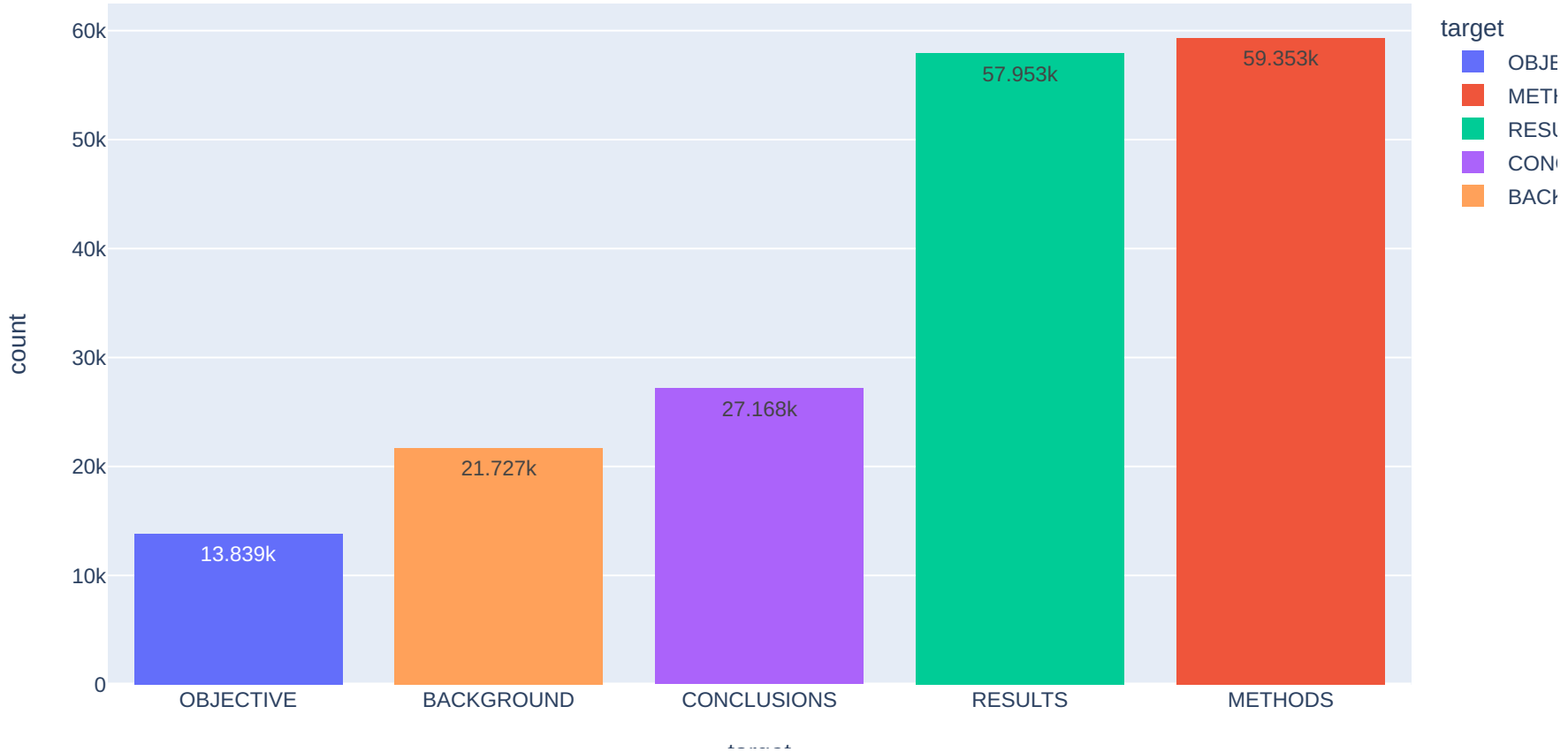
Out[6]:
```
(      target                                               text  line_number  \
 0  OBJECTIVE  to investigate the efficacy of @ weeks of dail...            0
 1    METHODS  a total of @ patients with primary knee oa wer...            1
 2    METHODS  outcome measures included pain reduction and i...            2
 3    METHODS  pain was assessed using the visual analog pain...            3
 4    METHODS  secondary outcome measures included the wester...            4

    total_lines
 0           11
 1           11
 2           11
 3           11
 4           11  ,
        target                                               text  line_number  \
 0  BACKGROUND  this study analyzed liver function abnormaliti...            0
 1     RESULTS  a post hoc analysis was conducted with the use...            1
 2     RESULTS  liver function tests ( lfts ) were measured at...            2
 3     RESULTS  survival analyses were used to assess the asso...            3
 4     RESULTS  the percentage of patients with abnormal lfts ...            4

    total_lines
 0            8
 1            8
 2            8
 3            8
 4            8  ,
        target                                               text  line_number  \
 0  BACKGROUND  ige sensitization to aspergillus fumigatus and...            0
 1  BACKGROUND  it is not clear whether these patients would b...            1
 2   OBJECTIVE  we sought to determine whether a @-month cours...            2
 3     METHODS  asthmatic patients who were ige sensitized to ...            3
 4     METHODS  primary outcomes were improvement in quality o...            4

    total_lines
 0            9
 1            9
 2            9
```

```
3            9
4            9  )
```

In [7]:
```
# fetching the value counts for each columns
tr_df['target'].value_counts(), ts_df['target'].value_counts(), vl_df['target'].value_counts()
```

Out[7]:
```
(METHODS        59353
 RESULTS        57953
 CONCLUSIONS    27168
 BACKGROUND     21727
 OBJECTIVE      13839
 Name: target, dtype: int64,
 METHODS        9897
 RESULTS        9713
 CONCLUSIONS    4571
 BACKGROUND     3621
 OBJECTIVE      2333
 Name: target, dtype: int64,
 METHODS        9964
 RESULTS        9841
 CONCLUSIONS    4582
 BACKGROUND     3449
 OBJECTIVE      2376
 Name: target, dtype: int64)
```

In [8]:
```
px.histogram(tr_df, x='target', color="target", text_auto=True,barmode='stack').update_xaxes(categoryorder=
```



In [9]:
```
prep_df = tr_df.copy()
prep_df.text = prep_df.text.apply(TC_UTILS_MANAGER_CLASS.nltk_preprocessor)
print(f"{tr_df.text.values[0]}\n\n{prep_df.text.values[0]}")
print("The number of sentences for training are : {} \nThe number of sentences for vaildation are : {}\n The
```

```
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
['i', 'me', 'my', 'myself', 'we']
```

In [10]:
```
train_sentences = tr_df['text'].tolist()
val_sentences = vl_df['text'].tolist()
test_sentences = ts_df['text'].tolist()

len(train_sentences), len(val_sentences), len(test_sentences)

X = prep_df["text"].values
y = prep_df["target"].values
```

In [11]:
```
train_sz,val_sz,test_sz = 0.7,0.2,0.1
X_train, X_val, X_test, y_train, y_val, y_test = TC_UTILS_MANAGER_CLASS.data_splitter(X=X, y=y, train_size=
print('Trained Data shape ----> X_train : {} , Y_train : {} \nValidation Data Shape -----> X_val : {} , Y_va
print (f"Sample point: {X_train[0]} → {y_train[0]}")
```

```
Trained Data shape ----> X_train : (126027,) , Y_train : (126027,)
Validation Data Shape -----> X_val : (27006,) , Y_val : (27006,)
Testing Data Shape -----> X_test : (27007,) , Y_test : (27007,)
Sample point: group allocation actively disclosed participants outcome assessors masked group allocation →
METHODS
```

In [12]:
```
label_enc = lb_encoder()
label_enc.lb_fit(y_train)
n_classes = len(label_enc)
label_enc.target_classes
```

Out[12]:
```
{'BACKGROUND': 0, 'CONCLUSIONS': 1, 'METHODS': 2, 'OBJECTIVE': 3, 'RESULTS': 4}
```

In [13]:
```
cl_nm =label_enc.target_classes.keys()
cl_nm
```

Out[13]:
```
dict_keys(['BACKGROUND', 'CONCLUSIONS', 'METHODS', 'OBJECTIVE', 'RESULTS'])
```

In [14]:
```
y_train = label_enc.lb_encode(y_train)
y_val = label_enc.lb_encode(y_val)
y_test = label_enc.lb_encode(y_test)
```

In [15]:
```
cnts = np.bincount(y_train)
clw_wts = {i: 1.0/count for i, count in enumerate(cnts)}
print (f"counts: {cnts}\nweights: {clw_wts}")
```

```
counts: [15209 19017 41547  9687 40567]
weights: {0: 6.575054244197515e-05, 1: 5.258452963138245e-05, 2: 2.4069126531398175e-05, 3: 0.000103231134
51016826, 4: 2.4650578056055415e-05}
```

In [16]:
```
tokenizer = ct_tokenizer(ch_lvl=False, nos_tkns=60000)
tokenizer.fitter(texts=X_train)
VOCAB_SIZE = len(tokenizer)
print(VOCAB_SIZE)
```

```
38868
```

In [17]:
```
print (take(5, tokenizer.tkn_to_idx.items()))
print (f"least freq token's freq: {tokenizer.min_token_freq}")
```

```
[('', 873), ('patients', 1), ('group', 2), ('treatment', 3), ('study', 4)]
least freq token's freq: 1
```

In [18]:
```
X_train = tokenizer.txt_seq(X_train)
X_val = tokenizer.txt_seq(X_val)
X_test = tokenizer.txt_seq(X_test)
```

In [19]:
```
train_dataset = CustomDataSetManger(X=X_train, y=y_train)
val_dataset = CustomDataSetManger(X=X_val, y=y_val)
test_dataset = CustomDataSetManger(X=X_test, y=y_test)
```

In [20]:
```
train_dataset[12]

train_dl = train_dataset.create_datald(batch_size=btch_sz)
val_dl = val_dataset.create_datald(batch_size=btch_sz)
test_dl = test_dataset.create_datald(batch_size=btch_sz)

len(train_dl), len(val_dl), len(test_dl)

batch_X, batch_seq_lens, batch_y = next(iter(train_dl))
batch_X[0]
```

```
/home/markins/Sequential-text-classification-using-deep-sequence-modelling/Torch_release_experiments/torch
_utils/tc_utils.py:448: VisibleDeprecationWarning:

Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays
with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object'
when creating the ndarray.
```

Out[20]:
```
tensor([   2, 1031, 6318, 9864,   21,   49, 4191, 1007,    2, 1031,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0])
```

Model 1 ---> LSTM + RNN

In [21]:
```
hd_dm, emd_dm = 128,128

model = Model1(
    embed_dm=emd_dm,
    hd_dim= hd_dm,
    voc_size=VOCAB_SIZE,
    num_lyrs=2,
    ln_output=64,
    num_classes=n_classes,
)

model = model.to(_plat)
model
```

Out[21]:
```
Model1(
  (embed): Embedding(38868, 128)
  (lstm): LSTM(128, 128, num_layers=2, batch_first=True, bidirectional=True)
  (fcd1): Linear(in_features=256, out_features=64, bias=True)
  (fcd2): Linear(in_features=64, out_features=5, bias=True)
  (drop_fn): Dropout(p=0.3, inplace=False)
)
```

In [22]:
```
lr_rate, patience, epochs  = 1e-4, 5, 25
cl_wts_tensor = torch.Tensor(list(clw_wts.values())).to(_plat)
loss_func = nn.CrossEntropyLoss(weight=cl_wts_tensor)
optimizer = optim.Adam(model.parameters(), lr=lr_rate)
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', factor = 0.1, patience=3)
trainer_arch = tc_baseline(
    model=model,
    device=_plat,
    loss_func=loss_func,
    optimizer=optimizer,
    scheduler=scheduler,
    dump_path='model_dumps/model1.pt'
)
```

In [23]:
```
appt_model = trainer_arch.training_engine(
    epochs=epochs,
    patience=patience,
    train_dl=train_dl,
    val_dl=val_dl
)
```

```
<----------------------Training in EPOCH: 1 ------------------------>
```

100% 1970/1970 [00:57<00:00, 35.91it/s]

100% 422/422 [00:03<00:00, 126.23it/s]

```
/home/markins/Sequential-text-classification-using-deep-sequence-modelling/Torch_release_experiments/torch
_models/tc_baseline.py:148: UserWarning:

Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argumen
t.


Dumping the model into the system in model_dumps/model1.pt
Please wait the model is being saved for epoch 0
Training loss : 1.200,  Training Accuracy : 0.538,      Validation loss :  0.965,       Validation Accurac
y : 0.653,      Learning-rate : 1.000E-04,      Patience : 5

<----------------------Training in EPOCH: 2 ------------------------>
```

In [24]:
```
# grab the predictions
test_ls, accu, y_true, y_prob = trainer_arch.evaluation_architecture(dloader=test_dl)
y_pred = np.argmax(y_prob, axis=1)

# performance
perf = TC_UTILS_MANAGER_CLASS.metrics_evaluater(y_true, y_pred, classes=label_enc.total_classes)
print(json.dumps(perf["overall"], indent=2))
```

```
100%                                                422/422 [00:03<00:00, 124.84it/s]

{
  "precision": 0.7466396349464208,
  "recall": 0.7414003776798608,
  "f1": 0.74321056391827,
  "num_samples": 27007.0
}
```

In [25]:
```
# !wget http://nlp.stanford.edu/data/glove.6B.zip
TC_UTILS_MANAGER_CLASS.unzipper('glove.6B.zip')
```

In [27]:
```
for ky in cl_nm:
    print(f"\nPrediction for {ky} : {perf['class'][ky]}")
```

```
Prediction for BACKGROUND : {'precision': 0.5656081485919713, 'recall': 0.5793188094507518, 'f1': 0.572381
3854782478, 'num_samples': 3259.0}

Prediction for CONCLUSIONS : {'precision': 0.5987601539119282, 'recall': 0.6871933267909716, 'f1': 0.63993
60292437742, 'num_samples': 4076.0}

Prediction for METHODS : {'precision': 0.8442669067176775, 'recall': 0.8427496349545097, 'f1': 0.843507588
5328837, 'num_samples': 8903.0}

Prediction for OBJECTIVE : {'precision': 0.5707898658718331, 'recall': 0.5534682080924855, 'f1': 0.5619955
979457081, 'num_samples': 2076.0}

Prediction for RESULTS : {'precision': 0.8258558892596712, 'recall': 0.7686644426550098, 'f1': 0.796234509
056244, 'num_samples': 8693.0}
```

Model 2 ---> Glove Embedded RNN+LSTM

In [28]:
```
hd_dm, emd_dm = 128,300
file = "glove.6B.{0}d.txt".format(emd_dm)
glove_emb = TC_UTILS_MANAGER_CLASS.load_embeddings_glove(filename=file)
emb_matrix = TC_UTILS_MANAGER_CLASS.embedding_mtrx_architecture(embeddings=glove_emb, wrd_idx=tokenizer.tkn_

print(f"\n The shape of the embedding matrix :: {emb_matrix.shape[0]}, and its dimensions are :: {emb_matri
```

```
 The shape of the embedding matrix :: 38868, and its dimensions are :: 300
```

In [30]:
```
model2 = Model2(
    embed_dm=emd_dm,
    hd_dim= hd_dm,
    voc_size=VOCAB_SIZE,
    num_lyrs=2,
    ln_output=128,
    num_classes=n_classes,
    pre_embed=emb_matrix
)

model2 = model2.to(_plat)
model2
```

Out[30]:
```
Model2(
  (embeddings): Embedding(38868, 300, padding_idx=0)
  (lstm): LSTM(300, 128, num_layers=2, batch_first=True, bidirectional=True)
  (fcd1): Linear(in_features=256, out_features=128, bias=True)
  (fcd2): Linear(in_features=128, out_features=5, bias=True)
  (drop_fn): Dropout(p=0.3, inplace=False)
)
```

In [31]:
```
lr_rate, patience, epochs  = 1e-4, 5, 25
cl_wts_tensor = torch.Tensor(list(clw_wts.values())).to(_plat)
loss_func = nn.CrossEntropyLoss(weight=cl_wts_tensor)
optimizer = optim.Adam(model.parameters(), lr=lr_rate)
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', factor = 0.1, patience=3)
trainer_arch = tc_baseline(
    model=model,
    device=_plat,
    loss_func=loss_func,
    optimizer=optimizer,
    scheduler=scheduler,
    dump_path='model_dumps/model2.pt'
)
```

In [32]:
```
appt_model = trainer_arch.training_engine(
    epochs=epochs,
    patience=patience,
    train_dl=train_dl,
    val_dl=val_dl
)
```

```
<----------------------Training in EPOCH: 1 ------------------------>

100%                                                1970/1970 [00:58<00:00, 34.64it/s]

/home/markins/Sequential-text-classification-using-deep-sequence-modelling/Torch_release_experiments/torch
_utils/tc_utils.py:448: VisibleDeprecationWarning:

Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays
with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object'
when creating the ndarray.


100%                                                422/422 [00:03<00:00, 129.57it/s]

/home/markins/Sequential-text-classification-using-deep-sequence-modelling/Torch_release_experiments/torch
_models/tc_baseline.py:148: UserWarning:

Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argumen
t.


Dumping the model into the system in model_dumps/model2.pt
Please wait the model is being saved for epoch 0
Training loss : 0.512,  Training Accuracy : 0.832,      Validation loss :  0.934,       Validation Accurac
y : 0.728,      Learning-rate : 1.000E-04,      Patience : 5

<----------------------Training in EPOCH: 2 ------------------------>

100%                                                1970/1970 [00:58<00:00, 35.26it/s]

100%                                                422/422 [00:03<00:00, 123.31it/s]

Training loss : 0.480,  Training Accuracy : 0.843,      Validation loss :  0.881,       Validation Accura
```

Training loss : 0.400, Training Accuracy : 0.846, Validation loss : 0.981, Validation Accurac
y : 0.733, Learning-rate : 1.000E-04, Patience : 5

<----------------------Training in EPOCH: 3 ------------------------>

100% 1970/1970 [00:59<00:00, 34.24it/s]

100% 422/422 [00:03<00:00, 118.59it/s]

Training loss : 0.449, Training Accuracy : 0.853, Validation loss : 1.025, Validation Accurac
y : 0.727, Learning-rate : 1.000E-04, Patience : 5

<----------------------Training in EPOCH: 4 ------------------------>

100% 1970/1970 [01:00<00:00, 35.07it/s]

100% 422/422 [00:03<00:00, 121.85it/s]

Training loss : 0.420, Training Accuracy : 0.863, Validation loss : 1.086, Validation Accurac
y : 0.729, Learning-rate : 1.000E-04, Patience : 5

<----------------------Training in EPOCH: 5 ------------------------>

100% 1970/1970 [00:59<00:00, 34.55it/s]

100% 422/422 [00:03<00:00, 123.29it/s]

Training loss : 0.393, Training Accuracy : 0.871, Validation loss : 1.128, Validation Accurac
y : 0.724, Learning-rate : 1.000E-05, Patience : 5

<----------------------Training in EPOCH: 6 ------------------------>

100% 1970/1970 [00:59<00:00, 34.97it/s]

100% 422/422 [00:03<00:00, 120.29it/s]

Patience state is zero !!!!!

Stopping early

In [34]:
```
# grab the predictions
test_ls, acc, y_true, y_prob = trainer_arch.evaluation_architecture(dloader=test_dl)
y_pred = np.argmax(y_prob, axis=1)

# performance
perf = TC_UTILS_MANAGER_CLASS.metrics_evaluater(y_true, y_pred, classes=label_enc.total_classes)
print(json.dumps(perf["overall"], indent=2))
```

```
100%                                                422/422 [00:03<00:00, 128.84it/s]

{
  "precision": 0.741369766530698,
  "recall": 0.7378087162587478,
  "f1": 0.7390789364251559,
  "num_samples": 27007.0
}
```

In [36]:
```
for ky in cl_nm:
    print(f"\n\nPrediction for {ky} : {perf['class'][ky]}")
```

```
Prediction for BACKGROUND : {'precision': 0.5539130434782609, 'recall': 0.5863761890150353, 'f1': 0.569682
5160232524, 'num_samples': 3259.0}


Prediction for CONCLUSIONS : {'precision': 0.6030431863951667, 'recall': 0.6611874386653582, 'f1': 0.63077
82328847279, 'num_samples': 4076.0}


Prediction for METHODS : {'precision': 0.8391357886488302, 'recall': 0.8419633831292823, 'f1': 0.840547207
8941467, 'num_samples': 8903.0}


Prediction for OBJECTIVE : {'precision': 0.5735829433177327, 'recall': 0.5313102119460501, 'f1': 0.5516379
094773693, 'num_samples': 2076.0}


Prediction for RESULTS : {'precision': 0.8164480077745384, 'recall': 0.7731508109973542, 'f1': 0.794209748
8921714, 'num_samples': 8693.0}
```

In [ ]: