Pip install necessary libraries

```
!pip install opendatasets
!pip install sentencepiece
!pip install --upgrade pip
!pip3.10 install seqeval
!pip install transformers
!pip install tqdm
!pip install datasets
!pip install transformers[torch]
!pip install accelerate -U
!pip install Kaggle
```

```
Collecting opendatasets
  Downloading opendatasets-0.1.22-py3-none-any.whl (15 kB)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/d
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dis
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/py
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dis
Installing collected packages: opendatasets
Successfully installed opendatasets-0.1.22
Requirement already satisfied: sentencepiece in /usr/local/lib/python3.10/di
Requirement already satisfied: pip in /usr/local/lib/python3.10/dist-package
Collecting pip
  Downloading pip-24.0-py3-none-any.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 12.0 MB/s eta 0:00:
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 23.1.2
    Uninstalling pip-23.1.2:
      Successfully uninstalled pip-23.1.2
Successfully installed pip-24.0
```

```
Collecting seqeval
  Downloading seqeval-1.2.2.tar.gz (43 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 43.6/43.6 kB 1.8 MB/s eta 0:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.10/di
Requirement already satisfied: scikit-learn>=0.21.3 in /usr/local/lib/pythor
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dis
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/pythor
Building wheels for collected packages: seqeval
  Building wheel for seqeval (setup.py) ... done
  Created wheel for seqeval: filename=seqeval-1.2.2-py3-none-any.whl size=16
  Stored in directory: /root/.cache/pip/wheels/1a/67/4a/ad4082dd7dfc30f2abfe
Successfully built seqeval
Installing collected packages: seqeval
Successfully installed seqeval-1.2.2
WARNING: Running pip as the 'root' user can result in broken permissions and
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dis
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: huggingface-hub<1.0,>=0.19.3 in /usr/local/li
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.1
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: tokenizers<0.19,>=0.14 in /usr/local/lib/pyth
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10
```

Upload your kaggle auth json

- This step is done to download the data directly using kaggle api

```
from google.colab import files
files.upload()
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ./kaggle.json
! kaggle datasets list
```

Choose Files no files selected Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle.json
mkdir: cannot create directory ‘/root/.kaggle’: File exists
401 - Unauthorized - Unauthenticated
```

Downlaod the data and load it to the disk

```
# download the datasets

import os
import opendatasets as od
import pandas as pd
import json

data_path = "./pii-detection-removal-from-educational-data/"

# download the data from kaggle
if not os.path.exists(data_path):
    print("Dataset not found, downloading from Kaggle")
    dataset = "https://www.kaggle.com/competitions/pii-detection-removal-from-edu
    od.download(dataset)
else:
    print("Dataset found in disk")

# check for the files present there
assert os.path.exists(data_path + "train.json"), "train.json file missing"
assert os.path.exists(data_path + "test.json"), "test.json file missing"


train_df = pd.read_json(open(data_path + "train.json"))
print("train_df loaded")

test_df = pd.read_json(open(data_path + "test.json"))
print("test_df loaded")
```

```
Dataset not found, downloading from Kaggle
Please provide your Kaggle credentials to download this dataset. Learn more: h
Your Kaggle username: aditya2901
Your Kaggle Key: ··········
Downloading pii-detection-removal-from-educational-data.zip to ./pii-detectio
100%|██████████| 21.4M/21.4M [00:01<00:00, 19.9MB/s]

Extracting archive ./pii-detection-removal-from-educational-data/pii-detectio
train_df loaded
test_df loaded
```

Check the splitup of labels in the data

```
from tqdm import tqdm
import numpy as np

data = json.load(open(data_path + "train.json"))
pos = []
neg = []

for d in tqdm(data):
    if any(np.array(d["labels"]) != "O"):
        pos.append(d)
    else:
        neg.append(d)

print("total datapoints : ", len(data))
print("positive examples : ", len(pos))
print("negative examples : ", len(neg))
```

```
100%|██████████| 6807/6807 [00:00<00:00, 7337.35it/s]total datapoints :  6807
positive examples :  945
negative examples :  5862
```

Get the unique labels and create a map

- there are totally 14 unique labels
- EMAIL, ID_NUM, NAME_STUDENT, PHONE_NUM, STREET_ADDRESS, URL_PERSONAL, USERNAME
- These labels represent what class the tokens belongs to
- Every label are subdivided into 2 parts -> B and I. There are represented in the prefix eg: B-EMAIL, I-EMAIL, I-URL_PERSONAL, B-USERNAME
- B represents begining of the class, I represent Intermediate of the class.
- A set of tokens can be represented by Begining or Intermediate. Eg: "Nathalie Sylvia" -> "B-USERNAME I-USERNAME"
- Other eg: "My Name is Aditya" -> "O O O B-NAME"
- tokens/words not belonging to the above mentioned class are represented by "O" -> object

```
from itertools import chain

# get the unique labels from the data
labels = sorted(list(set(chain(*[x["labels"] for x in data]))))

# create a map label to unique numbers
label_to_id = {l: i for i,l in enumerate(labels)}

# create a reverse map : unique numbers to label
id_to_label = {v:k for k,v in label_to_id.items()}

target = [
    'B-EMAIL', 'B-ID_NUM', 'B-NAME_STUDENT', 'B-PHONE_NUM',
    'B-STREET_ADDRESS', 'B-URL_PERSONAL', 'B-USERNAME', 'I-ID_NUM',
    'I-NAME_STUDENT', 'I-PHONE_NUM', 'I-STREET_ADDRESS', 'I-URL_PERSONAL'
]

print(id_to_label)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-c0c714e68540> in <cell line: 4>()
      2
      3
----> 4 labels = sorted(list(set(chain(*[x["labels"] for x in data]))))
      5 label_to_id = {l: i for i,l in enumerate(labels)}
      6 id_to_label = {v:k for k,v in label_to_id.items()}

NameError: name 'data' is not defined
```

Create a custom tokenizers

- The input data has been tokenized in a different way but we need to tokenize the data as per the model requirements
- Every model has their own unique tokenizers. For DeBERTa, we use DeBERTa tokenizer.
- In order to re-tokenize the data, we need to initially combine all the tokens (de-tokenize) and then use DeBERTa tokenizer to re-tokenize it

```
def tokenize(example, tokenizer, label2id, max_length):

    # rebuild text from tokens
    text = []
    labels = []

    for t, l, ws in zip(example["tokens"], example["provided_labels"], example["t
        text.append(t)
        labels.extend([l] * len(t))

        if ws:
            text.append(" ")
            labels.append("O")

    # actual tokenization
    tokenized = tokenizer("".join(text), return_offsets_mapping=True, max_length=

    labels = np.array(labels)

    text = "".join(text)
    token_labels = []

    for start_idx, end_idx in tokenized.offset_mapping:
        # CLS token
        if start_idx == 0 and end_idx == 0:
            token_labels.append(label2id["O"])
            continue

        # case when token starts with whitespace
        if text[start_idx].isspace():
            start_idx += 1

        token_labels.append(label2id[labels[start_idx]])

    length = len(tokenized.input_ids)

    return {**tokenized, "labels": token_labels, "length": length}
```

Init the model config

- Convert the input data into Dataset class as expected by the model

```
from transformers import AutoTokenizer
from datasets import Dataset, features

TRAINING_MODEL_PATH = "microsoft/deberta-v3-base"
TRAINING_MAX_LENGTH = 1024
OUTPUT_DIR = "output"

tokenizer = AutoTokenizer.from_pretrained(TRAINING_MODEL_PATH)

ds = Dataset.from_dict({
    "full_text": [x["full_text"] for x in data],
    "document": [str(x["document"]) for x in data],
    "tokens": [x["tokens"] for x in data],
    "trailing_whitespace": [x["trailing_whitespace"] for x in data],
    "provided_labels": [x["labels"] for x in data],
})
ds = ds.map(tokenize, fn_kwargs={"tokenizer": tokenizer, "label2id": label_to_id,
# ds = ds.class_encode_column("group")
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: Us
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access pu
  warnings.warn(
```

tokenizer_config.json: 100% 52.0/52.0 [00:00<00:00, 3.05kB/s]

config.json: 100% 579/579 [00:00<00:00, 34.9kB/s]

spm.model: 100% 2.46M/2.46M [00:00<00:00, 50.2MB/s]

```
/usr/local/lib/python3.10/dist-packages/transformers/convert_slow_tokenizer.py
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/multiprocess/popen_fork.py:66: Runtime
  self.pid = os.fork()
```

Map (num_proc=3): 100% 6807/6807 [01:47<00:00, 9.78 examples/s]

```
Truncation was not explicitly activated but `max_length` is provided a specifi
Truncation was not explicitly activated but `max_length` is provided a specifi
Truncation was not explicitly activated but `max_length` is provided a specifi
```

Get some sample (token, class) pair

```
x = ds[0]

for t,l in zip(x["tokens"], x["provided_labels"]):
    if l != "O":
        print((t,l))

print("*"*100)

for t, l in zip(tokenizer.convert_ids_to_tokens(x["input_ids"]), x["labels"]):
    if id_to_label[l] != "O":
        print((t,id_to_label[l]))
```

```
('Nathalie', 'B-NAME_STUDENT')
('Sylla', 'I-NAME_STUDENT')
('Nathalie', 'B-NAME_STUDENT')
('Sylla', 'I-NAME_STUDENT')
('Nathalie', 'B-NAME_STUDENT')
('Sylla', 'I-NAME_STUDENT')
****************************************************************************************************
('N', 'B-NAME_STUDENT')
('atha', 'B-NAME_STUDENT')
('lie', 'B-NAME_STUDENT')
('_S', 'I-NAME_STUDENT')
('ylla', 'I-NAME_STUDENT')
('N', 'B-NAME_STUDENT')
('atha', 'B-NAME_STUDENT')
('lie', 'B-NAME_STUDENT')
('_S', 'I-NAME_STUDENT')
('ylla', 'I-NAME_STUDENT')
('N', 'B-NAME_STUDENT')
('atha', 'B-NAME_STUDENT')
('lie', 'B-NAME_STUDENT')
('_S', 'I-NAME_STUDENT')
('ylla', 'I-NAME_STUDENT')
```

Create your custom evaluation:

- By default, the model pipeline has categorical cross entropy. But we need to override it to custom eval metrics (recall, precision and F1)

```
from seqeval.metrics import recall_score, precision_score
from seqeval.metrics import classification_report
from seqeval.metrics import f1_score

def compute_metrics(p, all_labels):
    predictions, labels = p
    predictions = np.argmax(predictions, axis=2)

    # Remove ignored index (special tokens)
    true_predictions = [
        [all_labels[p] for (p, l) in zip(prediction, label) if l != -100]
        for prediction, label in zip(predictions, labels)
    ]
    true_labels = [
        [all_labels[l] for (p, l) in zip(prediction, label) if l != -100]
        for prediction, label in zip(predictions, labels)
    ]

    recall = recall_score(true_labels, true_predictions)
    precision = precision_score(true_labels, true_predictions)
    f1_score = (1 + 5*5) * recall * precision / (5*5*precision + recall)

    results = {
        'recall': recall,
        'precision': precision,
        'f1': f1_score
    }
    return results
```

Init the model

```
from transformers import AutoModelForTokenClassification, DataCollatorForTokenCla
from functools import partial

model = AutoModelForTokenClassification.from_pretrained(
    TRAINING_MODEL_PATH,
    num_labels=len(labels),
    id2label=id_to_label,
    label2id=label_to_id,
    ignore_mismatched_sizes=True
)
collator = DataCollatorForTokenClassification(tokenizer, pad_to_multiple_of=16)
```

```
pytorch_model.bin: 100% [progress bar] 371M/371M [00:02<00:00, 106MB/s]
Some weights of DebertaV2ForTokenClassification were not initialized from the
You should probably TRAIN this model on a down-stream task to be able to use i
```

create the training pipeline

```
# I actually chose to not use any validation set. This is only for the model I us
args = TrainingArguments(
    output_dir=OUTPUT_DIR,
    fp16=True,
    learning_rate=2e-5,
    num_train_epochs=3,
    per_device_train_batch_size=4,
    gradient_accumulation_steps=2,
    report_to="none",
    evaluation_strategy="no",
    do_eval=False,
    save_total_limit=1,
    logging_steps=20,
    lr_scheduler_type='cosine',
    metric_for_best_model="f1",
    greater_is_better=True,
    warmup_ratio=0.1,
    weight_decay=0.01
)

trainer = Trainer(
    model=model,
    args=args,
    train_dataset=ds,
    data_collator=collator,
    tokenizer=tokenizer,
    compute_metrics=partial(compute_metrics, all_labels=labels),
)
```

```
/usr/local/lib/python3.10/dist-packages/accelerate/accelerator.py:436: FutureW
dataloader_config = DataLoaderConfiguration(dispatch_batches=None, split_batch
  warnings.warn(
```

Train the model

```
%%time
trainer.train()
```

[2553/2553 23:58, Epoch 3/3]

Step	Training Loss

20	1.868700
40	1.488000
60	0.565200
80	0.026000
100	0.007600
120	0.007500
140	0.014500
160	0.011400
180	0.013200
200	0.011600
220	0.008300
240	0.010800
260	0.006000
280	0.009900
300	0.002800
320	0.003000
340	0.008900
360	0.003100
380	0.001700
400	0.005600
420	0.005800
440	0.005000
460	0.005500
480	0.003200
500	0.001400
520	0.002600
540	0.001300
560	0.003100

580	0.008600
600	0.002900
620	0.001400

login to hugginface to store the model in huggingface hub

700	0.002100

```
import huggingface_hub

hub_path = "model_deberta3base_1024_token_classification"

huggingface_hub.login()

trainer.model.push_to_hub(hub_path)
tokenizer.push_to_hub(hub_path)
```

```
Token is valid (permission: write).
n has been saved in your configured git credential helper
ur token has been saved to /root/.cache/huggingface/tok
Login successful
```

model.safetensors: 100% 735M/735M [00:22<00:00, 50.6MB/s]

README.md: 100% 5.18k/5.18k [00:00<00:00, 305kB/s]

spm.model: 100% 2.46M/2.46M [00:00<00:00, 8.11MB/s]

```
CommitInfo(commit_url='https://huggingface.co/adhi29/model_deberta3base_1024_t
```

1020	0.002100

Inference

- create a tokenizer for inference, which does the same thing as the tokenizer in training but here we dont want to include labels

1120	0.001600

```
import json
import argparse
from itertools import chain
import pandas as pd
from pathlib import Path
from transformers import AutoTokenizer, AutoModelForTokenClassification, Trainer,
from datasets import Dataset
import numpy as np

INFERENCE_MAX_LENGTH = 2048
MODEL_PATH = ""

def tokenize_inference(example, tokenizer):
    text = []
    token_map = []

    idx = 0

    for t, ws in zip(example["tokens"], example["trailing_whitespace"]):

        text.append(t)
        token_map.extend([idx]*len(t))
        if ws:
            text.append(" ")
            token_map.append(-1)

        idx += 1


    tokenized = tokenizer("".join(text), return_offsets_mapping=True, truncation=

    return {
        **tokenized,
        "token_map": token_map,
    }
```

1580	0.000700

Load the model and data path

1640	0.000400
1660	0.002000

```
hub_model_full_path = "adhi29/model_deberta3base_1024_token_classification"

data = json.load(open("./pii-detection-removal-from-educational-data/train.json")

ds = Dataset.from_dict({
    "full_text": [x["full_text"] for x in data],
    "document": [x["document"] for x in data],
    "tokens": [x["tokens"] for x in data],
    "trailing_whitespace": [x["trailing_whitespace"] for x in data],
})

tokenizer = AutoTokenizer.from_pretrained(hub_model_full_path)
ds = ds.map(tokenize_inference, fn_kwargs={"tokenizer": tokenizer}, num_proc=2)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: Us
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access pu
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/multiprocess/popen_fork.py:66: Runtime
  self.pid = os.fork()
Map (num_proc=2): 100% 6807/6807 [00:56<00:00, 12.31 examples/s]
/usr/local/lib/python3.10/dist-packages/multiprocess/popen_fork.py:66: Runtime
  self.pid = os.fork()
```

set the training pipline

2060	0.000300
2080	0.000300
2100	0.000300
2120	0.000600
2140	0.000200
2160	0.000400
2180	0.000600
2200	0.001200
2220	0.000700

```
model = AutoModelForTokenClassification.from_pretrained(hub_model_full_path)
# model = trainer.model
collator = DataCollatorForTokenClassification(tokenizer, pad_to_multiple_of=16)

args = TrainingArguments(
    ".",
    per_device_eval_batch_size=1,
    report_to="none",
)

trainer = Trainer(
    model=model,
    args=args,
    data_collator=collator,
    tokenizer=tokenizer,
)
```

```
/usr/local/lib/python3.10/dist-packages/accelerate/accelerator.py:436: FutureW
dataloader_config = DataLoaderConfiguration(dispatch_batches=None, split_batch
  warnings.warn(
```

get all the predictions from the model

```
predictions = trainer.predict(ds).predictions
pred_softmax = np.exp(predictions) / np.sum(np.exp(predictions), axis = 2).reshap
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-5-a33eb6775289> in <cell line: 5>()
      3
      4 # config = json.load(open(Path(model_path) / "config.json"))
----> 5 id2label = config["id2label"]
      6 # id2label = dict(map(lambda x: (str(x[0]), x[1]),
id_to_label.items()))
      7 preds = predictions.argmax(-1)

NameError: name 'config' is not defined
```

load the label-id map from the config files of the model

```
# config = json.load(open(Path(hub_model_full_path) / "config.json"))

config = model.config.to_dict()
id2label = config["id2label"]

# id_to_label = config["id_to_label"]
# id2label = dict(map(lambda x: (str(x[0]), x[1]), id_to_label.items()))

preds = predictions.argmax(-1)
preds_without_O = pred_softmax[:,:,:12].argmax(-1)
O_preds = pred_softmax[:,:,12]

threshold = 0.9
preds_final = np.where(O_preds < threshold, preds_without_O , preds)

print(id2label)
```

```
{0: 'B-EMAIL', 1: 'B-ID_NUM', 2: 'B-NAME_STUDENT', 3: 'B-PHONE_NUM', 4: 'B-STR
```

modify the output from the model such that it can be evaluated later

```
triplets = []
document, token, label, token_str = [], [], [], []

for i, V in enumerate(zip(preds_final, ds["token_map"], ds["offset_mapping"], ds['
    p, token_map, offsets, tokens, doc = V

    for token_pred, (start_idx, end_idx) in zip(p, offsets):
        label_pred = id2label[(token_pred)]

        if start_idx + end_idx == 0: continue

        if token_map[start_idx] == -1:
            start_idx += 1

        # ignore "\n\n"
        while start_idx < len(token_map) and tokens[token_map[start_idx]].isspace
            start_idx += 1

        if start_idx >= len(token_map): break

        token_id = token_map[start_idx]

        # ignore "O" predictions and whitespace preds
        if label_pred != "O" and token_id != -1:
            triplet = (label_pred, token_id, tokens[token_id])

            if triplet not in triplets:
                document.append(doc)
                token.append(token_id)
                label.append(label_pred)
                token_str.append(tokens[token_id])
                triplets.append(triplet)
```

store the pred output into a csv

```
df = pd.DataFrame({
    "document": document,
    "token": token,
    "label": label,
    "token_str": token_str
})
df["row_id"] = list(range(len(df)))

df.to_csv("sample_pred.csv", sep = ",", index=False, encoding="utf-8")

display(df.head(100))
```

	document	token	label	token_str	row_id
0	7	9	B-NAME_STUDENT	Nathalie	0
1	7	10	I-NAME_STUDENT	Sylla	1
2	7	482	B-NAME_STUDENT	Nathalie	2
3	7	483	I-NAME_STUDENT	Sylla	3
4	7	741	B-NAME_STUDENT	Nathalie	4
...	...	...	...	...	...
95	609	71	B-NAME_STUDENT	Swetha	95
96	609	72	I-NAME_STUDENT	Swetha	96
97	609	74	B-ID_NUM	784372734211	97
98	609	77	B-NAME_STUDENT	Alex	98
99	609	78	I-NAME_STUDENT	Swetha	99

100 rows × 5 columns