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
```

```
Requirement already satisfied: opendatasets in /usr/local/lib/python3.10/dis
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/c
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
WARNING: Running pip as the 'root' user can result in broken permissions anc
Requirement already satisfied: sentencepiece in /usr/local/lib/python3.10/di
WARNING: Running pip as the 'root' user can result in broken permissions anc
Requirement already satisfied: pip in /usr/local/lib/python3.10/dist-package
WARNING: Running pip as the 'root' user can result in broken permissions anc
Requirement already satisfied: seqeval in /usr/local/lib/python3.10/dist-pac
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
WARNING: Running pip as the 'root' user can result in broken permissions anc
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
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/
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
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.
WARNING: Running pip as the 'root' user can result in broken permissions and
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packag
WARNING: Running pip as the 'root' user can result in broken permissions and
Requirement already satisfied: datasets in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist
Requirement already satisfied: pyarrow>=12.0.0 in /usr/local/lib/python3.10/
Requirement already satisfied: pyarrow-hotfix in /usr/local/lib/python3.10/d
Requirement already satisfied: dill<0.3.9,>=0.3.0 in /usr/local/lib/python3.
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dis
Requirement already satisfied: xxhash in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: multiprocess in /usr/local/lib/python3.10/dis
```

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
Warning: Your Kaggle API key is readable by other users on this system! To fix
Warning: Looks like you're using an outdated API Version, please consider upda
ref                                                             title
--------------------------------------------------------------  ----------------
sudarshan24byte/online-food-dataset                             Online Food Data
nbroad/gemma-rewrite-nbroad                                     gemma-rewrite-nb
lovishbansal123/adult-census-income                             Adult Census Inc
sukhmandeepsinghbrar/most-subscribed-youtube-channel            Most Subscribed
sanyamgoyal401/customer-purchases-behaviour-dataset             Customer Purchas
startalks/pii-models                                            pii-models
fatemehmehrparvar/obesity-levels                                Obesity Levels
sahirmaharajj/employee-salaries-analysis                        Employee Salarie
bhavikjikadara/student-study-performance                        Student Study Pe
soumyajitjalua/crop-datasets-for-all-indian-states-2010-2017    Crop Datasets fo
sukhmandeepsinghbrar/housing-price-dataset                      Housing Price Da
divu2001/restaurant-order-data                                  Restaurant Order
willianoliveiragibin/worlds-wildlife                            world’s wildlife
sahirmaharajj/air-pollution-dataset                             Air Pollution Da
joshuanaude/effects-of-alcohol-on-student-performance           Effects of Alcoh
zubairmustafa/shopping-mall-customer-segmentation-data          Shopping Mall Cu
mohdshahnawazaadil/credit-card-dataset                          Credit Card Data
sahilnbajaj/loans-data                                          Loans Data
rushikeshdane20/global-trends-in-atmospheric-carbon-dioxide     Global Trends in
```

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
Downloading pii-detection-removal-from-educational-data.zip to ./pii-detectio
100%|██████████| 21.4M/21.4M [00:00<00:00, 192MB/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 [00:01<00:00, 4274.72it/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)
```

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

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=max_length,
                          truncation=True,
                          padding="max_length")

    # print(tokenized

    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 = "FacebookAI/roberta-base"
TRAINING_MAX_LENGTH = 511
OUTPUT_DIR = "output"

tokenizer = AutoTokenizer.from_pretrained(TRAINING_MODEL_PATH)

# tokenizer.model_max_length = model.config.max_position_embeddings

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/multiprocess/popen_fork.py:66: Runtime
  self.pid = os.fork()
```

Map (num_proc=3): 100% 6807/6807 [01:23<00:00, 46.69 examples/s]

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

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')
('ath', 'B-NAME_STUDENT')
('al', 'B-NAME_STUDENT')
('ie', 'B-NAME_STUDENT')
('ĠSy', 'I-NAME_STUDENT')
('lla', '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)
```

```
Some weights of RobertaForTokenClassification were not initialized from the mo
You should probably TRAIN this model on a down-stream task to be able to use i
```

Start coding or generate with AI.

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="wanb",
    evaluation_strategy="epoch",
    save_strategy="epoch",
    do_eval=True,
    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 09:54, Epoch 3/3]

Step	Training Loss
20	2.702300
40	2.366300
60	1.180400
80	0.048800
100	0.010100
120	0.009400
140	0.015600
160	0.008800
180	0.008100
200	0.007500
220	0.005000
240	0.007600
260	0.004100
280	0.008200
300	0.003000
320	0.002900
340	0.003400
360	0.002600
380	0.001500
400	0.003400
420	0.002200
440	0.006600
460	0.003600
480	0.001800
500	0.001000
520	0.002600

540	0.002400
560	0.002900
580	0.004100
600	0.002400
620	0.001200

login to hugginface to store the model in huggingface hub

700	0.002300

```
import huggingface_hub

hub_path = "model_robertabase_1024_token_classification"

huggingface_hub.login()

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

```
VBox(children=(HTML(value='<center>
<img\nsrc=https://huggingface.co/front/assets/huggingface_logo-noborder.sv…
```

model.safetensors: 100% 496M/496M [00:40<00:00, 27.6MB/s]

README.md: 100% 5.18k/5.18k [00:00<00:00, 358kB/s]

```
CommitInfo(commit_url='https://huggingface.co/adhi29/model_robertabase_1024_to
```

Inference

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

1020	0.001300
1040	0.000900
1060	0.002600
1080	0.003200

```
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 = 511
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,
    }
```

1540	0.002000

Load the model and data path

1600	0.000600
1620	0.001800

```
hub_path = "model_robertabase_1024_token_classification"
hub_path = "model_robertabase_1024_token_classification"

hub_model_full_path = "adhi29/" + hub_path

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(
```

tokenizer_config.json: 100% 1.22k/1.22k [00:00<00:00, 38.9kB/s]

vocab.json: 100% 798k/798k [00:00<00:00, 8.13MB/s]

merges.txt: 100% 456k/456k [00:00<00:00, 7.25MB/s]

tokenizer.json: 100% 2.11M/2.11M [00:00<00:00, 30.5MB/s]

special_tokens_map.json: 100% 280/280 [00:00<00:00, 12.3kB/s]

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

Map (num_proc=2): 100% 6807/6807 [00:57<00:00, 79.44 examples/s]

set the training pipline

2140	0.000200
2160	0.000300
2180	0.000600

```
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,
)
```

config.json: 100% 1.34k/1.34k [00:00<00:00, 61.9kB/s]

model.safetensors: 100% 496M/496M [00:05<00:00, 156MB/s]

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

2500	0.000200

get all the predictions from the model

```
CPU times: user 7min 3s, sys: 14.6 s, total: 7min 18s
```

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

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	10	0	B-NAME_STUDENT	Diego	2
3	10	1	I-NAME_STUDENT	Estrada	3
4	10	464	B-NAME_STUDENT	Diego	4
...	...	...	...	...	...
95	659	0	B-NAME_STUDENT	Madina	95
96	659	1	I-NAME_STUDENT	Eno	96
97	671	13	B-NAME_STUDENT	Amparo	97
98	714	3	B-NAME_STUDENT	Edgar	98
99	714	4	I-NAME_STUDENT	Lara	99

100 rows × 5 columns

Get scores

```
train_df = pd.read_json(open(data_path + "train.json"))
train_df["document"].max()

pred_df = pd.read_csv("sample_pred.csv")
pred_df["document"].max()
```

```
print("total len of pred_df : ", len(pred_df))

# create a copy
train_df_clean = train_df.copy()

def create_token_list(token_list):
  return list(range(len(token_list)))  # Create list based on current list length

train_df_clean["token"] = train_df_clean["tokens"].apply(create_token_list)

# explode the columns "tokens" and "labels"
train_df_clean = train_df_clean.explode(["tokens", "labels", "token"], ignore_ind

# drop the unnecessary columns
train_df_clean.drop(columns=["full_text", "trailing_whitespace"], inplace=True)

# rename the columns
train_df_clean.rename(columns={"tokens" : "token_str", "labels" : "label"}, inpla

# create a new column "token"
# train_df_clean["token"] = train_df_clean.index

# filter the rows based on label
train_df_clean = train_df_clean[train_df_clean["label"] != "O"]

# reset the index again
train_df_clean.reset_index(drop=True, inplace=True)

# create a column "row_id"
train_df_clean["row_id"] = train_df_clean.index

train_df_clean.head()

print("total len of train_df_cleaned : ", len(train_df_clean))
print("Total NA : ")
print(train_df_clean.isna().sum())
```

```
total len of pred_df :  2464
total len of train_df_cleaned :  2739
Total NA :
document     0
token_str    0
label        0
token        0
row_id       0
dtype: int64
<ipython-input-12-eb99efd7acef>:36: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
  train_df_clean["row_id"] = train_df_clean.index
```

Compare the values and get the score

```
pred_df_copy = pred_df.drop(columns=["row_id", "token_str"])
train_df_clean_copy = train_df_clean.drop(columns=["row_id", "token_str"])

comb_df = pd.merge(pred_df_copy, train_df_clean_copy, how="outer", suffixes=("","_
print("total len of comb_df : ", len(comb_df))
display(comb_df.isna().sum())

comb_df.fillna("O", inplace=True)

display(comb_df.isna().sum())
```

```
total len of comb_df :  3278
document        0
token           0
label         814
label_pred    523
dtype: int64
document      0
token         0
label         0
label_pred    0
dtype: int64
```

```
comb_df
```

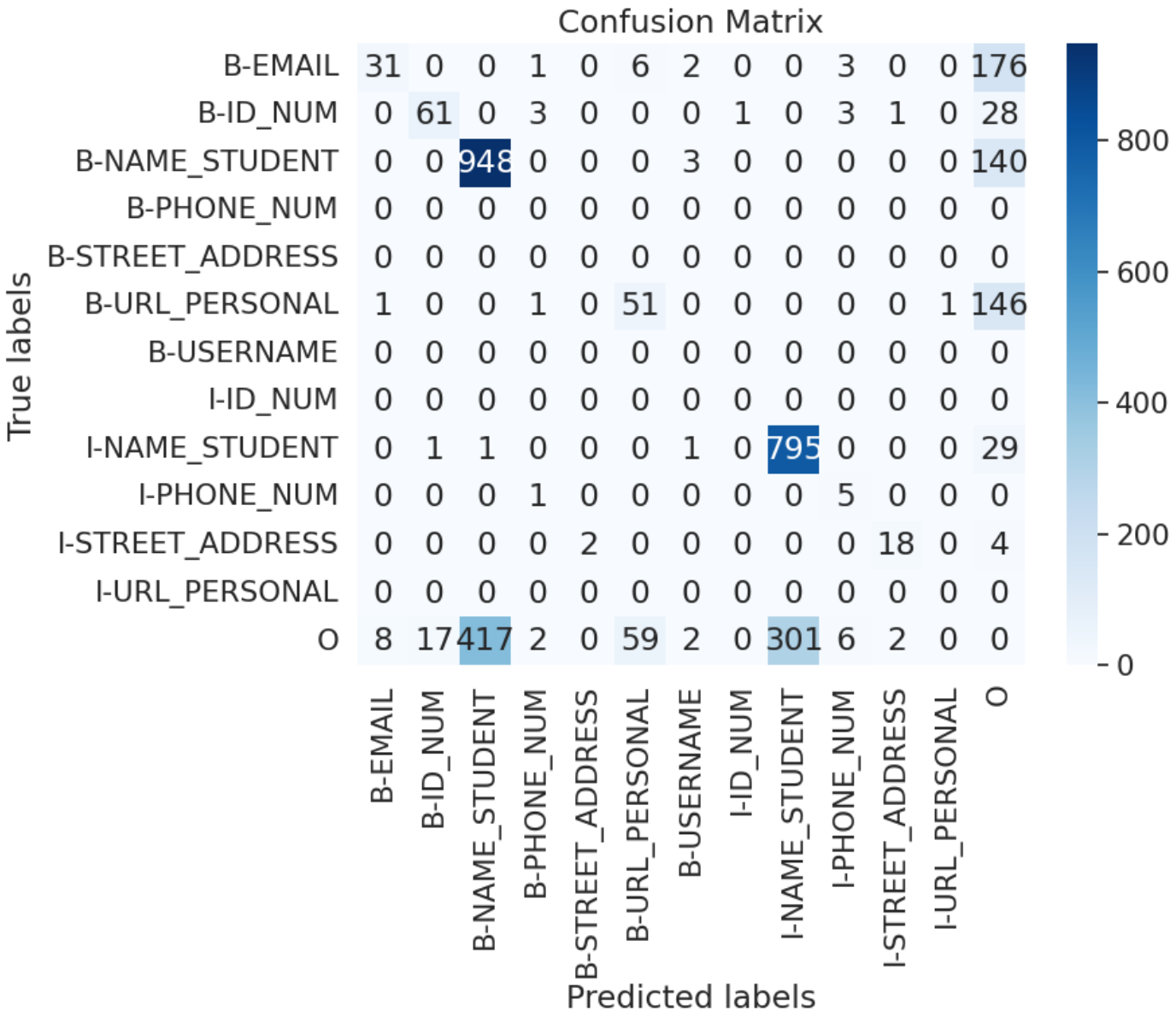
	document	token	label	label_pred
0	7	9	B-NAME_STUDENT	B-NAME_STUDENT
1	7	10	I-NAME_STUDENT	I-NAME_STUDENT
2	10	0	B-NAME_STUDENT	B-NAME_STUDENT
3	10	1	I-NAME_STUDENT	I-NAME_STUDENT
4	10	464	B-NAME_STUDENT	B-NAME_STUDENT
...	...	...	...	...
3273	13342	0	O	B-NAME_STUDENT
3274	13342	1	O	I-NAME_STUDENT
3275	13342	523	O	B-NAME_STUDENT
3276	13342	524	O	I-NAME_STUDENT
3277	15717	964	O	B-ID_NUM

3278 rows × 4 columns

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

y_true = comb_df["label"]
y_pred = comb_df["label_pred"]

labels = np.unique(np.concatenate((y_true, y_pred)))
print(labels)

# Compute confusion matrix
cm = confusion_matrix(y_true, y_pred, labels=labels)

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.set(font_scale=1.4)  # for label size
sns.heatmap(cm, annot=True, fmt='g', cmap='Blues',
            xticklabels=labels,
```

```
            yticklabels=labels)

plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
```

```
['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' 'O']
```



```
from sklearn.metrics import fbeta_score

score = fbeta_score(y_true, y_pred, beta=5, average="weighted")
print("F Beta Score : ", score)
```

```
F Beta Score :  0.5772161031609374
```

```
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score

f1_score = f1_score(y_true, y_pred, average="weighted")
print("F1 Score : ", score)

acc_score = accuracy_score(y_true, y_pred)
print("Accuracy Score : ", acc_score)

prec_score = precision_score(y_true, y_pred, average="weighted")
print("Precision Score : ", prec_score)

recall_score = recall_score(y_true, y_pred, average="weighted")
print("Recall Score : ", recall_score)
```

```
F1 Score :  0.5772161031609374
Accuracy Score :  0.5823672971323978
Precision Score :  0.5222448240397003
Recall Score :  0.5823672971323978
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:134
  _warn_prf(average, modifier, msg_start, len(result))
```

Start coding or generate with AI.