```
# for references
#https://spacy.io/usage/processing-pipelines#sourced-components
#https://spacy.io/api/doc
#https://spacy.io/usage/training
#https://spacy.io/usage/saving-loading
# https://stackoverflow.com/questions/69181078/spacy-how-do-you-add-custom-ner-labels-to-a-pre-trained-model
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# installing Med7 (GLOVE and roberta embeddings) and it's related libraries
!python -m pip install jedi
!python -m pip install -U wheel pip setuptools pip install spacy==3.4.4 pip install spacy-transformers==1.1.9
!python -m pip install https://huggingface.co/kormilitzin/en_core_med7_lg/resolve/main/en_core_med7_lg-any-py3-none-any.whl
!python -m pip install https://huggingface.co/kormilitzin/en_core_med7_trf/resolve/main/en_core_med7_trf-any-py3-none-any.whl
```

```
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.10/dist-packages (from thinc<8.2.0,>=8.
Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/python3.10/dist-packages (from typer<0.8.0,>=0.3.0->
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->spacy<3.5.0,>=3.4
Installing collected packages: en-core-med7-lg
Successfully installed en-core-med7-lg-3.4.2.1
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system packag
Collecting en-core-med7-trf==any
  Downloading https://huggingface.co/kormilitzin/en_core_med7_trf/resolve/main/en_core_med7_trf-any-py3-none-any.whl (101
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 GB 1.9 MB/s eta 0:00:00
Requirement already satisfied: spacy<3.5.0,>=3.4.2 in /usr/local/lib/python3.10/dist-packages (from en-core-med7-trf==any
Requirement already satisfied: spacy-transformers<1.2.0,>=1.1.6 in /usr/local/lib/python3.10/dist-packages (from en-core-
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.10 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>=3
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>=3.4.2->
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>=3.4.2
Requirement already satisfied: thinc<8.2.0,>=8.1.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>=3.4.2->
Requirement already satisfied: wasabi<1.1.0,>=0.9.1 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>=3.4.2-
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>=3.4.2->
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>=3.4
Requirement already satisfied: typer<0.8.0,>=0.3.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>=3.4.2->
Requirement already satisfied: pathy>=0.3.5 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>=3.4.2->en-core
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>=3.
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>=3.4.2->
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>=3.4.2->en-cor
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>=3.4
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<1.11.0,>=1.7.4 in /usr/local/lib/python3.10/dist-packages (from spa
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>=3.4.2->en-core-med7-
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>=3.4.2->en-core-m
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>=3.4.2->en-c
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.5.0,>=3.4
Requirement already satisfied: transformers<4.26.0,>=3.4.0 in /usr/local/lib/python3.10/dist-packages (from spacy-transfo
Requirement already satisfied: torch>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from spacy-transformers<1.2.0,>=1
Requirement already satisfied: spacy-alignments<1.0.0,>=0.7.2 in /usr/local/lib/python3.10/dist-packages (from spacy-trar
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from pydantic!=1.8,!=
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spa
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /usr/local/lib/python3.10/dist-packages (from thinc<8.2.0,>=8.1.0->s
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.10/dist-packages (from thinc<8.2.0,>=8.
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch>=1.6.0->spacy-transformers
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch>=1.6.0->spacy-transformers<1.
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.6.0->spacy-transformers
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages (from torch>=1.6.0->spacy-transfo
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch>=1.6.0->spacy-
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch>=1.6.0->spacy-tr
Requirement already satisfied: huggingface-hub<1.0,>=0.10.0 in /usr/local/lib/python3.10/dist-packages (from transformers
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers<4.26.0,>=3.4.0->
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers<4.26.0,>=3
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in /usr/local/lib/python3.10/dist-packages (from transfo
Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/python3.10/dist-packages (from typer<0.8.0,>=0.3.0->
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->spacy<3.5.0,>=3.4
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.10.0->tran
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.6.0->spacy-t
Installing collected packages: en-core-med7-trf
Successfully installed en-core-med7-trf-3.4.2.1
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system packag
```

```
# DO NOT FORGET TO CHANGE THE PATHS AND THE FILES NAMES

# this code is to read bert data format (tokens, ner_tags, input_ids, attention_mask, labels)
# and change the data format to character offset for spacy ner model
# it generate  an excel file with all the tokens in the discharge summary that are labelled as entities only.
```

```
# you can edit the code to generate an excel file with all the tokens in the discharge summary, either an entity or not
# by uncommenting line No 34 "gold_data.append(gold_data_dic)"

import pandas as pd
def preprocess(df, name, path):
  gold_data_entities = []
  gold_data_tokens = []
  for i, tags in enumerate(df.ner_tags.to_list()):
    gold = []
    ch_start = 0
    for l, label in enumerate(tags):
      if label == "O":
        token = df._get_value(i, 'tokens')
        tmp_token = token[l]
        tmp_label = label
        tmp_start = l
        tmp_end = l+1

        gold_data_dic = {}
        ch_end = ch_start + len(tmp_token)
        gold_data_dic["file_id"] = i
        gold_data_dic['gold_label'] = tmp_label
        gold_data_dic['token_start'] = tmp_start
        gold_data_dic['token_end'] = tmp_end
        gold_data_dic['entity_text'] = tmp_token
        gold_data_dic['ch_start'] = ch_start
        gold_data_dic['ch_end'] = ch_end
        ch_start = ch_end+1
        #print(tmp_start, tmp_end, tmp_label, tmp_token)
        # uncomment the line below if you want to generate a file with all the tokens (entity or not)
        #gold_data_entities.append(gold_data_dic)
        gold_data_tokens.append(gold_data_dic)

      if label != "O":
        if "B-" in label:
          token = df._get_value(i, 'tokens')
          tmp_token = token[l]
          tmp_label = label[2:]
          tmp_start = l

          out = 0
          if l+1 < len(tags):
            tmp_end = l+1
            #print(l, l+1, tmp_label, tmp_token)
            for nl in range(l+1,len(tags)):
              #print(nl, token[nl])
              if "B-" in tags[nl]:
                out = 1
                break
              elif "O" == tags[nl]:
                out = 1
                break
              elif "I-" in tags[nl]:
                token = df._get_value(i, 'tokens')
                tmp_token += " " + token[nl]
                tmp_start = l
                tmp_end = nl+1
                #print(tmp_start, tmp_end, tmp_label, tmp_token)
            if out == 1:
              gold_data_dic = {}

              ch_end = ch_start + len(tmp_token)
              gold_data_dic["file_id"] = i
              gold_data_dic['gold_label'] = tmp_label
              gold_data_dic['token_start'] = tmp_start
              gold_data_dic['token_end'] = tmp_end
              gold_data_dic['entity_text'] = tmp_token
              gold_data_dic['ch_start'] = ch_start
              gold_data_dic['ch_end'] = ch_end
              ch_start = ch_end+1
              #print(tmp_start, tmp_end, tmp_label, tmp_token)
              gold_data_entities.append(gold_data_dic)
              gold_data_tokens.append(gold_data_dic)


  gold_data_entity = pd.DataFrame.from_records(gold_data_entities)
  gold_data_token = pd.DataFrame.from_records(gold_data_tokens)
  print(len(gold_data_entity), len(gold_data_token))
  print(gold_data_entity)
  print(gold_data_token)

  # this line to save the excel file of entities only
```

```
    gold_data_entity.to_excel(path+'gold_data_entity_CHoffsetEntitiesOnly_'+name+'.xlsx', index=False)

    # this line to save the excel file with all tokens
    gold_data_token.to_excel(path+'gold_data_entity_CHoffset_all_token_'+name+'.xlsx', index=False)
```

```
path = "/content/drive/MyDrive/collabrations_/HumanLoopH/Med7/data/"

train_validation = pd.read_json(path+"train_validation_429.json", lines=True)
print("pre-processing train_validation set")
preprocess(train_validation, 'train_validation_429', path)

train = pd.read_json(path+"train_353.json", lines=True)
print("pre-processing training set")
preprocess(train, 'train_353', path)

validation = pd.read_json(path+"validation_76.json", lines=True)
print("pre-processing validation set")
preprocess(validation, 'validation_76', path)

test = pd.read_json(path+"test_76.json", lines=True)
print("pre-processing testing set")
preprocess(test, 'test_76', path)
```

```
pre-processing train_validation set
70809 850795
       file_id gold_label  token_start  token_end         entity_text  \
0            0       Drug           21         22              Keflex
1            0       Drug           22         23             Orencia
2            0       Drug           23         24            Remicade
3            0       Drug          113        114                vanc
4            0       Drug          115        116               cipro
...        ...        ...          ...        ...                 ...
70804      428       Drug         1731       1734  narcotic pain meds
70805      428        ADE         1736       1737        constipating
70806      428     Reason         1764       1765         constipated
70807      428       Drug         1777       1778           Metamucil
70808      428       Drug         1779       1782    Milk of Magnesia

       ch_start  ch_end
0           105     111
1           112     119
2           120     128
3           633     637
4           642     647
...         ...     ...
70804      9474    9492
70805      9500    9512
70806      9657    9668
70807      9726    9735
70808      9739    9755

[70809 rows x 7 columns]
        file_id gold_label  token_start  token_end entity_text  ch_start  \
0             0          O            0          1   Admission         0
1             0          O            1          2        Date        10
2             0          O            2          3        2202        15
3             0          O            3          4           1        20
4             0          O            4          5           8        22
...         ...        ...          ...        ...         ...       ...
850790      428          O         2225       2226   Completed     12350
850791      428          O         2226       2227          by     12360
850792      428          O         2227       2228        2142     12363
850793      428          O         2228       2229           8     12368
850794      428          O         2229       2230          28     12370

        ch_end
0            9
1           14
2           19
3           21
4           23
...        ...
850790   12359
850791   12362
850792   12367
850793   12369
850794   12372

[850795 rows x 7 columns]
pre-processing training set
59212 700032
```

```
# test the results of en_core_med7_trf and en_core_med7_lg on testing set WITHOUT fine-tuining

# 1- create one discharge summary with its NER labels
# 2- send the summary to Med7 for prediction NER labels
# 3- evaluate the results with Gold standard

import pandas as pd
import spacy


def testing(df, name, path):

  med7 = spacy.load(name)

  str_dic_lg = []
  predict_labels_lg_dic = []
  predict_labels_lg = []
  for i, token in enumerate(df.tokens.to_list()):
    token_str_lg = ' '.join(str(t) for t in token)
    labels_lg = df._get_value(i, 'ner_tags')
    str_dic_lg.append([token_str_lg, labels_lg])
    predicts_lg = []
    entities = med7(token_str_lg)
    for e in entities.ents:
      predict_dic = {}
      predict_dic["predict_file_id"] = i
      predict_dic['predict_label'] = e.label_
      predict_dic['predict_start'] = e.start
      predict_dic['predict_end'] = e.end
      predict_dic['predict_text'] = e.text
      predict_dic['start_char'] = e.start_char
      predict_dic['end_char'] = e.end_char

      #print("e.text", e.text)
      #print("e.label_", e.label_)
      #print('start', e.start)
      #print('end', e.end)
      #print('char_span', e.char_span(e.start_char, e.end_char))
      #print('start_char', e.start_char)
      #print('end_char', e.end_char)
      #print('ent_id', e.ent_id)
      #print('ent_id_', e.ent_id_)
      #print('ents', e.ents[0])
      #print('label', e.label)
      #print('id', e.id)
      #print('id_', e.id_)

      predict_labels_lg_dic.append(predict_dic)
      predicts_lg.append([e.start, e.end, e.text, e.label_])
    print(predict_dic)
    predict_labels_lg.append(predicts_lg)

  print(len(predict_labels_lg))
  print(len(predict_labels_lg_dic))
  predict_label_entity = pd.DataFrame.from_records(predict_labels_lg_dic)
  print(predict_label_entity)

  # uncomment the line below to save the output of MED7 prediction
  predict_label_entity.to_excel(path + 'predict_label_entity_76testDataset_'+name+'.xlsx', index=False)


path = "/content/drive/MyDrive/collabrations_/HumanLoopH/Med7/data/"
df = pd.read_json(path+"test_76.json", lines=True)
model = ["en_core_med7_lg", "en_core_med7_trf"]
for name in model:
  print("testing the performance of "+name+" over the testing set")
  testing(df, name, path)
```

```
testing the performance of en_core_med7_lg over the testing set
{'predict_file_id': 0, 'predict_label': 'DRUG', 'predict_start': 2594, 'predict_end': 2595, 'predict_text': 'lamictal', '
{'predict_file_id': 1, 'predict_label': 'FREQUENCY', 'predict_start': 3244, 'predict_end': 3245, 'predict_text': 'PRN', '
{'predict_file_id': 2, 'predict_label': 'DRUG', 'predict_start': 3204, 'predict_end': 3205, 'predict_text': 'lovenox', 's
{'predict_file_id': 3, 'predict_label': 'FREQUENCY', 'predict_start': 808, 'predict_end': 810, 'predict_text': 'q MWF', '
{'predict_file_id': 4, 'predict_label': 'DURATION', 'predict_start': 1577, 'predict_end': 1581, 'predict_text': 'for 5 mo
{'predict_file_id': 5, 'predict_label': 'DRUG', 'predict_start': 2481, 'predict_end': 2482, 'predict_text': 'Ciprofloxaci
{'predict_file_id': 6, 'predict_label': 'FREQUENCY', 'predict_start': 1141, 'predict_end': 1143, 'predict_text': 'per day
{'predict_file_id': 7, 'predict_label': 'FREQUENCY', 'predict_start': 111, 'predict_end': 113, 'predict_text': 'q day', '
{'predict_file_id': 8, 'predict_label': 'DOSAGE', 'predict_start': 1570, 'predict_end': 1571, 'predict_text': '4', 'start
{'predict_file_id': 9, 'predict_label': 'FREQUENCY', 'predict_start': 587, 'predict_end': 588, 'predict_text': 'ASDIR', '
{'predict_file_id': 10, 'predict_label': 'DRUG', 'predict_start': 1628, 'predict_end': 1629, 'predict_text': 'THIAZIDE',
{'predict_file_id': 11, 'predict_label': 'FREQUENCY', 'predict_start': 1743, 'predict_end': 1745, 'predict_text': 'every
{'predict_file_id': 12, 'predict_label': 'FREQUENCY', 'predict_start': 1715, 'predict_end': 1719, 'predict_text': 'four t
{'predict_file_id': 13, 'predict_label': 'DOSAGE', 'predict_start': 1954, 'predict_end': 1955, 'predict_text': '1', 'star
```

```
{'predict_file_id': 14, 'predict_label': 'ROUTE', 'predict_start': 2912, 'predict_end': 2914, 'predict_text': 'nasal cann
{'predict_file_id': 15, 'predict_label': 'DOSAGE', 'predict_start': 686, 'predict_end': 687, 'predict_text': '60', 'start
{'predict_file_id': 16, 'predict_label': 'DRUG', 'predict_start': 76, 'predict_end': 77, 'predict_text': 'Megace', 'start
{'predict_file_id': 17, 'predict_label': 'FREQUENCY', 'predict_start': 418, 'predict_end': 421, 'predict_text': 'q eight
{'predict_file_id': 18, 'predict_label': 'DRUG', 'predict_start': 2018, 'predict_end': 2021, 'predict_text': 'narcotic pa
{'predict_file_id': 19, 'predict_label': 'FORM', 'predict_start': 2479, 'predict_end': 2480, 'predict_text': 'gum', 'star
{'predict_file_id': 20, 'predict_label': 'FORM', 'predict_start': 1716, 'predict_end': 1717, 'predict_text': 'patch', 'st
{'predict_file_id': 21, 'predict_label': 'FREQUENCY', 'predict_start': 2153, 'predict_end': 2155, 'predict_text': 'as nee
{'predict_file_id': 22, 'predict_label': 'DRUG', 'predict_start': 602, 'predict_end': 603, 'predict_text': 'vasopressors'
{'predict_file_id': 23, 'predict_label': 'DOSAGE', 'predict_start': 2151, 'predict_end': 2152, 'predict_text': 'taper', '
{'predict_file_id': 24, 'predict_label': 'FREQUENCY', 'predict_start': 1474, 'predict_end': 1476, 'predict_text': 'with m
{'predict_file_id': 25, 'predict_label': 'DOSAGE', 'predict_start': 2987, 'predict_end': 2988, 'predict_text': '3', 'star
{'predict_file_id': 26, 'predict_label': 'FREQUENCY', 'predict_start': 2195, 'predict_end': 2197, 'predict_text': 'as nee
{'predict_file_id': 27, 'predict_label': 'DRUG', 'predict_start': 1232, 'predict_end': 1233, 'predict_text': 'ciprofloxac
{'predict_file_id': 28, 'predict_label': 'FREQUENCY', 'predict_start': 759, 'predict_end': 762, 'predict_text': 'once a d
{'predict_file_id': 29, 'predict_label': 'DRUG', 'predict_start': 1890, 'predict_end': 1891, 'predict_text': 'steroid', '
{'predict_file_id': 30, 'predict_label': 'DRUG', 'predict_start': 1672, 'predict_end': 1674, 'predict_text': 'stool softe
{'predict_file_id': 31, 'predict_label': 'FREQUENCY', 'predict_start': 1987, 'predict_end': 1990, 'predict_text': 'once p
{'predict_file_id': 32, 'predict_label': 'FREQUENCY', 'predict_start': 1642, 'predict_end': 1643, 'predict_text': 'daily'
{'predict_file_id': 33, 'predict_label': 'DRUG', 'predict_start': 3608, 'predict_end': 3609, 'predict_text': 'antibiotics
{'predict_file_id': 34, 'predict_label': 'DRUG', 'predict_start': 1339, 'predict_end': 1340, 'predict_text': 'Keppra', 's
{'predict_file_id': 35, 'predict_label': 'DOSAGE', 'predict_start': 1153, 'predict_end': 1154, 'predict_text': '1', 'star
{'predict_file_id': 36, 'predict_label': 'FREQUENCY', 'predict_start': 1160, 'predict_end': 1163, 'predict_text': 'q 8 h'
{'predict_file_id': 37, 'predict_label': 'DURATION', 'predict_start': 2120, 'predict_end': 2122, 'predict_text': '2 weeks
{'predict_file_id': 38, 'predict_label': 'DOSAGE', 'predict_start': 1648, 'predict_end': 1649, 'predict_text': '28', 'sta
{'predict_file_id': 39, 'predict_label': 'DRUG', 'predict_start': 2059, 'predict_end': 2060, 'predict_text': 'Methylpheni
{'predict_file_id': 40, 'predict_label': 'FREQUENCY', 'predict_start': 2747, 'predict_end': 2749, 'predict_text': 'every
{'predict_file_id': 41, 'predict_label': 'DRUG', 'predict_start': 919, 'predict_end': 920, 'predict_text': 'bicarbonate',
{'predict_file_id': 42, 'predict_label': 'FREQUENCY', 'predict_start': 1350, 'predict_end': 1355, 'predict_text': 'once a
{'predict_file_id': 43, 'predict_label': 'FREQUENCY', 'predict_start': 1458, 'predict_end': 1464, 'predict_text': 'every
{'predict_file_id': 44, 'predict_label': 'FREQUENCY', 'predict_start': 2541, 'predict_end': 2542, 'predict_text': 'daily'
{'predict_file_id': 45, 'predict_label': 'DRUG', 'predict_start': 675, 'predict_end': 676, 'predict_text': 'lipitor', 'st
{'predict_file_id': 46, 'predict_label': 'DRUG', 'predict_start': 1933, 'predict_end': 1934, 'predict_text': 'steroids',
{'predict_file_id': 47, 'predict_label': 'FREQUENCY', 'predict_start': 3234, 'predict_end': 3235, 'predict_text': 'daily'
{'predict_file_id': 48, 'predict_label': 'DRUG', 'predict_start': 2436, 'predict_end': 2440, 'predict_text': 'hypoglycemi
{'predict_file_id': 49, 'predict_label': 'DURATION', 'predict_start': 2084, 'predict_end': 2087, 'predict_text': 'for 2 d
{'predict_file_id': 50, 'predict_label': 'DURATION', 'predict_start': 115, 'predict_end': 117, 'predict_text': '14 days',
{'predict_file_id': 51, 'predict_label': 'DRUG', 'predict_start': 2380, 'predict_end': 2381, 'predict_text': 'LISINOPRIL'
{'predict_file_id': 52, 'predict_label': 'FREQUENCY', 'predict_start': 1640, 'predict_end': 1643, 'predict_text': 'every
{'predict_file_id': 53, 'predict_label': 'FREQUENCY', 'predict_start': 1208, 'predict_end': 1212, 'predict_text': 'bid ti
{'predict_file_id': 54, 'predict_label': 'DURATION', 'predict_start': 1041, 'predict_end': 1044, 'predict_text': 'x 4 day
{'predict_file_id': 55, 'predict_label': 'FREQUENCY', 'predict_start': 2376, 'predict_end': 2379, 'predict_text': 'once a
{'predict_file_id': 56, 'predict_label': 'FREQUENCY', 'predict_start': 1511, 'predict_end': 1512, 'predict_text': 'qPM',
```

```
# this code is to save the results of prediction in suitable format to calculate the confusion matrix of TP, FN, FP, TN
# using type match (at least part of the token text is annotated with the correct entity type)
# and using strict match (the token text and the entity type has to be matched the gold data)
# COR: correct annotation of type
# INC: incorrect annotation of type
# MIS: missing annotation by Med7
# SPU: Spurius is for a token predicted by Med7 with an entity label but it's not in the gold data

# see this website explains NER evaluation:
# https://www.davidsbatista.net/blog/2018/05/09/Named_Entity_Evaluation/


import pandas as pd

def processing(true_label_entity, predict_label_entity, name, path):

  Eval = []

  predict_label_entity_len = len(predict_label_entity)
  type_list = []
  strict_list = []
  for i, row in true_label_entity.iterrows():
    if i%500 == 0:

      print("batch: ", i)
    tmp = 0
    for c, srow in predict_label_entity.iterrows():
      #print(c)
      if srow[0] > row[0]:
        #print('LARGER', row[0], srow[0])
        break
      #if c%5000 == 0:
        #print("batch c", c)
      Eval_dic = {}
      if row[0] == srow[0]:
        #print('EQUALS', row[0], srow[0])
        if row[5] == srow[5] and row[6] == srow[6] and str(row[4]) == str(srow[4]):
        #if str(row[4]).lower() == str(srow[4]).lower():
          if str(row[1]).lower() == str(srow[1]).lower():
            Eval_dic['file_id'] = str(row[0])
            Eval_dic['true_label'] = str(row[1]).lower()
            Eval_dic['true_start'] = row[5]
```

```
            Eval_dic['true_end'] = row[6]
            Eval_dic['true_text'] = str(row[4])
            Eval_dic['predict_file_id'] = str(srow[0])
            Eval_dic['predict_label'] = str(srow[1]).lower()
            Eval_dic['predict_start'] = srow[5]
            Eval_dic['predict_end'] = srow[6]
            Eval_dic['predict_text'] = str(srow[4])
            Eval_dic['strict_label'] = 'COR'
            Eval_dic['type_label'] = 'COR'
            #print('str(row[4]).lower() == str(srow[4]).lower()', Eval_dic)
            Eval.append(Eval_dic)

            strict_list.append('COR')
            type_list.append('COR')
            #print("strict_list.append(COR), type_list.append(COR)")
            #true_label_entity = true_label_entity.drop([i])
            #predict_label_entity = predict_label_entity.drop([c])
            #predict_label_entity_len -= 1
            #tmp = 1
            #break
          elif str(row[1]).lower() != str(srow[1]).lower():
            Eval_dic['file_id'] = str(row[0])
            Eval_dic['true_label'] = str(row[1]).lower()
            Eval_dic['true_start'] = row[5]
            Eval_dic['true_end'] = row[6]
            Eval_dic['true_text'] = str(row[4])
            Eval_dic['predict_file_id'] = str(srow[0])
            Eval_dic['predict_label'] = str(srow[1]).lower()
            Eval_dic['predict_start'] = srow[5]
            Eval_dic['predict_end'] = srow[6]
            Eval_dic['predict_text'] = str(srow[4])
            Eval_dic['strict_label'] = 'INC'
            Eval_dic['type_label'] = 'INC'
            #print('str(row[4]).lower() == str(srow[4]).lower()', Eval_dic)
            Eval.append(Eval_dic)

            strict_list.append('INC')
            type_list.append('INC')
            #print("strict_list.append(INC), type_list.append(INC)")
          true_label_entity = true_label_entity.drop([i])
          predict_label_entity = predict_label_entity.drop([c])
          predict_label_entity_len -= 1
          tmp = 1
          break

        elif row[5] == srow[5] and str(row[4]) in str(srow[4]):
          if str(row[1]).lower() == str(srow[1]).lower():
            Eval_dic['file_id'] = str(row[0])
            Eval_dic['true_label'] = str(row[1]).lower()
            Eval_dic['true_start'] = row[5]
            Eval_dic['true_end'] = row[6]
            Eval_dic['true_text'] = str(row[4])
            Eval_dic['predict_file_id'] = str(srow[0])
            Eval_dic['predict_label'] = str(srow[1]).lower()
            Eval_dic['predict_start'] = srow[5]
            Eval_dic['predict_end'] = srow[6]
            Eval_dic['predict_text'] = str(srow[4])
            Eval_dic['strict_label'] = 'INC'
            Eval_dic['type_label'] = 'COR'
            #print('row[2] <= srow[2]', Eval_dic)
            Eval.append(Eval_dic)

            type_list.append('COR')
            strict_list.append('INC')
            #print("strict_list.append(INC), type_list.append(COR)")

          elif str(row[1]).lower() != str(srow[1]).lower():
            Eval_dic['file_id'] = str(row[0])
            Eval_dic['true_label'] = str(row[1]).lower()
            Eval_dic['true_start'] = row[5]
            Eval_dic['true_end'] = row[6]
            Eval_dic['true_text'] = str(row[4])
            Eval_dic['predict_file_id'] = str(srow[0])
            Eval_dic['predict_label'] = str(srow[1]).lower()
            Eval_dic['predict_start'] = srow[5]
            Eval_dic['predict_end'] = srow[6]
            Eval_dic['predict_text'] = str(srow[4])
            Eval_dic['strict_label'] = 'INC'
            Eval_dic['type_label'] = 'INC'
            #print('row[2] <= srow[2]', Eval_dic)
            Eval.append(Eval_dic)
```

```
            type_list.append('INC')
            strict_list.append('INC')
            #print("strict_list.append(INC), type_list.append(INC)")
          #predict_label_entity = predict_label_entity.drop([c])
          #predict_label_entity_len -= 1
          #if row[3] < srow[3]:
          true_label_entity = true_label_entity.drop([i])
          tmp = 1


        elif row[6] == srow[6] and str(row[4]) in str(srow[4]):
          if str(row[1]).lower() == str(srow[1]).lower():
            Eval_dic['file_id'] = str(row[0])
            Eval_dic['true_label'] = str(row[1]).lower()
            Eval_dic['true_start'] = row[5]
            Eval_dic['true_end'] = row[6]
            Eval_dic['true_text'] = str(row[4])
            Eval_dic['predict_file_id'] = str(srow[0])
            Eval_dic['predict_label'] = str(srow[1]).lower()
            Eval_dic['predict_start'] = srow[5]
            Eval_dic['predict_end'] = srow[6]
            Eval_dic['predict_text'] = str(srow[4])
            Eval_dic['strict_label'] = 'INC'
            Eval_dic['type_label'] = 'COR'
            #print('row[3] <= srow[3]', Eval_dic)
            Eval.append(Eval_dic)

            type_list.append('COR')
            strict_list.append('INC')
            #print("strict_list.append(INC), type_list.append(COR)")
              #true_label_entity = true_label_entity.drop([i])

              #break
          elif str(row[1]).lower() != str(srow[1]).lower():
              #print("equals", i, c)
            Eval_dic['file_id'] = str(row[0])
            Eval_dic['true_label'] = str(row[1]).lower()
            Eval_dic['true_start'] = row[5]
            Eval_dic['true_end'] = row[6]
            Eval_dic['true_text'] = str(row[4])
            Eval_dic['predict_file_id'] = str(srow[0])
            Eval_dic['predict_label'] = str(srow[1]).lower()
            Eval_dic['predict_start'] = srow[5]
            Eval_dic['predict_end'] = srow[6]
            Eval_dic['predict_text'] = str(srow[4])
            Eval_dic['strict_label'] = 'INC'
            Eval_dic['type_label'] = 'INC'
            #print('row[3] <= srow[3]', Eval_dic)
            Eval.append(Eval_dic)

            type_list.append('INC')
            strict_list.append('INC')
            #print("strict_list.append(INC), type_list.append(INC)")
              #true_label_entity = true_label_entity.drop([i])
          predict_label_entity = predict_label_entity.drop([c])
          predict_label_entity_len -= 1
          true_label_entity = true_label_entity.drop([i])
              #break
          tmp = 1



      if tmp == 0:
        if i in true_label_entity.index:
          Eval_dic = {}

          Eval_dic['file_id'] = str(row[0])
          Eval_dic['true_label'] = str(row[1]).lower()
          Eval_dic['true_start'] = row[5]
          Eval_dic['true_end'] = row[6]
          Eval_dic['true_text'] = str(row[4])
          Eval_dic['predict_file_id'] = str(srow[0])
          Eval_dic['predict_label'] = "O"
          Eval_dic['predict_start'] = row[5]
          Eval_dic['predict_end'] = row[6]
          Eval_dic['predict_text'] = str(row[4])
          Eval_dic['strict_label'] = 'MIS'
          Eval_dic['type_label'] = 'MIS'
          Eval.append(Eval_dic)

          #print(row)
          strict_list.append('MIS')
```

```
          type_list.append('MIS')
          #print("strict_list.append(MIS), type_list.append(MIS)")
          true_label_entity = true_label_entity.drop([i])

      #print(len(predict_label_entity), predict_label_entity_len)
    for c, srow in predict_label_entity.iterrows():
      #if len(predict_label_entity) > predict_label_entity_len:
      Eval_dic = {}
      Eval_dic['file_id'] = str(srow[0])
      Eval_dic['true_label'] = 'O'
      Eval_dic['true_start'] = srow[5]
      Eval_dic['true_end'] = srow[6]
      Eval_dic['true_text'] = str(srow[4])
      Eval_dic['predict_file_id'] = str(srow[0])
      Eval_dic['predict_label'] = str(srow[1]).lower()
      Eval_dic['predict_start'] = srow[5]
      Eval_dic['predict_end'] = srow[6]
      Eval_dic['predict_text'] = str(srow[4])
      Eval_dic['strict_label'] = 'SPU'
      Eval_dic['type_label'] = 'SPU'
      Eval.append(Eval_dic)
      #print(srow)
      strict_list.append('SPU')
      type_list.append('SPU')
      #print("strict_list.append(SPU), type_list.append(SPU)")
      predict_label_entity = predict_label_entity.drop([c])
      predict_label_entity_len -= 1

    true_predict_eval = pd.DataFrame.from_records(Eval)
    print(len(true_predict_eval))
    # uncomment this line to save the file, DO NOT FORGET TO CHANGE PATH AND FILE NAME
    true_predict_eval.to_excel(path+'true_predict_label_entity_76testDataset_'+name+'.xlsx', index=False)  #true_predict_label_e


path = "/content/drive/MyDrive/collabrations_/HumanLoopH/Med7/data/"

true_label_entity = pd.read_excel(path+'gold_data_entity_CHoffsetEntitiesOnly_test_76.xlsx')
print(true_label_entity)

model = ["en_core_med7_lg", "en_core_med7_trf"]
for name in model:
  print("processing the output of "+name+" predictions over the testing set")
  predict_label_entity = pd.read_excel(path + 'predict_label_entity_76testDataset_'+name+'.xlsx') #predict_label_entity_76test
  print(predict_label_entity)
  processing(true_label_entity, predict_label_entity, name, path)
```

```
       file_id gold_label  token_start  token_end  \
0            0       Drug           21         22
1            0       Drug           22         25
2            0       Drug           25         26
3            0       Drug          115        116
4            0       Drug          124        125
...        ...        ...          ...        ...
12536       75  Frequency          909        911
12537       75       Drug          986        987
12538       75       Drug          988        989
12539       75       Drug          990        991
12540       75       Drug          992        993

                        entity_text  ch_start  ch_end
0                       Penicillins       108     119
1      Sulfa Sulfonamide Antibiotics       120     149
2                          Lamictal       150     158
3                       Clindamycin       734     745
4                          Levaquin       778     786
...                             ...       ...     ...
12536                       q day      5535    5540
12537                      Percocet      6038    6046
12538                        Motrin      6049    6055
12539                        Celexa      6058    6064
12540                         Lasix      6067    6072

[12541 rows x 7 columns]
processing the output of en_core_med7_lg predictions over the testing set
       predict_file_id predict_label  predict_start  predict_end predict_text  \
0                    0          DRUG             21           22  Penicillins
1                    0          DRUG            115          116  Clindamycin
2                    0        DOSAGE            117          118            1
3                    0          DRUG            124          125     Levaquin
4                    0          DRUG            126          127   Vancomycin
...                ...           ...            ...          ...          ...
12050               75          DRUG            986          987     Percocet
12051               75      STRENGTH            987          989     2 Motrin
```

```
12052              75        DOSAGE            989          990            3
12053              75          DRUG            990          991       Celexa
12054              75      STRENGTH            991          993      4 Lasix

       start_char  end_char
0             108       119
1             734       745
2             748       749
3             778       786
4             791       801
...           ...       ...
12050        6038      6046
12051        6047      6055
12052        6056      6057
12053        6058      6064
12054        6065      6072

[12055 rows x 7 columns]
batch:  0
batch:  500
batch:  1000
```

```
# This code is to calculate P, R, F1 scores based on matching the entity type between reference and candidate.
# support in the output is the number of reference per entity type.

import pandas as pd
from sklearn.metrics import classification_report
def class_report(y_true, y_pred, path, name):

  file = open(path+"prediction_results_without_fine-tuning_"+name+".txt", 'w')
  labels = ['reason', 'ade', 'form', 'strength', 'dosage', 'drug', 'route', 'frequency', 'duration']
  print(type(labels), labels)
  report = classification_report(y_true, y_pred, labels = labels)
  print(report)
  file.writelines(report)

  labels.remove('reason') # remove 'reason' label from evaluation
  labels.remove('ade') # remove 'reason' label from evaluation
  print(type(labels), labels)
  report = classification_report(y_true, y_pred, labels = labels)
  print(report)
  file.writelines(report)
  file.close()


  #return report


path = "/content/drive/MyDrive/collabrations_/HumanLoopH/Med7/data/"
model = ["en_core_med7_lg", "en_core_med7_trf"]
for name in model:
  print("caluculating P, R, F1 scores of "+name+" predictions over the testing set")
  eval_report_all = pd.read_excel(path+'true_predict_label_entity_76testDataset_'+name+'.xlsx') #predict_label_entity_76testDa
  print("76 Test dataset "+name)
  eval_report_copy_all = eval_report_all.copy()
  y_true = eval_report_copy_all['true_label'].tolist()
  y_pred = eval_report_copy_all['predict_label'].tolist()
  print('type matching')
  #all = class_report(y_true, y_pred)
  class_report(y_true, y_pred, path, name)
```

```
caluculating P, R, F1 scores of en_core_med7_lg predictions over the testing set
76 Test dataset en_core_med7_lg
type matching
<class 'list'> ['reason', 'ade', 'form', 'strength', 'dosage', 'drug', 'route', 'frequency', 'duration']
              precision    recall  f1-score   support

      reason       0.00      0.00      0.00       927
         ade       0.00      0.00      0.00       242
        form       0.90      0.90      0.90      1696
    strength       0.70      0.80      0.75      1639
      dosage       0.11      0.24      0.15      1039
        drug       0.90      0.77      0.83      3954
       route       0.96      0.94      0.94      1341
   frequency       0.74      0.79      0.76      1564
    duration       0.73      0.75      0.74       139

   micro avg       0.71      0.69      0.70     12541
   macro avg       0.56      0.58      0.56     12541
weighted avg       0.71      0.69      0.70     12541

<class 'list'> ['form', 'strength', 'dosage', 'drug', 'route', 'frequency', 'duration']
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
```

```
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
  _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

        form       0.90      0.90      0.90      1696
    strength       0.70      0.80      0.75      1639
      dosage       0.11      0.24      0.15      1039
        drug       0.90      0.77      0.83      3954
       route       0.96      0.94      0.94      1341
   frequency       0.74      0.79      0.76      1564
    duration       0.73      0.75      0.74       139

   micro avg       0.71      0.77      0.74     11372
   macro avg       0.72      0.74      0.72     11372
weighted avg       0.78      0.77      0.77     11372

caluculating P, R, F1 scores of en_core_med7_trf predictions over the testing set
76 Test dataset en_core_med7_trf
type matching
<class 'list'> ['reason', 'ade', 'form', 'strength', 'dosage', 'drug', 'route', 'frequency', 'duration']
              precision    recall  f1-score   support

      reason       0.00      0.00      0.00       927
         ade       0.00      0.00      0.00       242
        form       0.84      0.93      0.88      1696
    strength       0.70      0.79      0.74      1639
      dosage       0.28      0.42      0.34      1039
        drug       0.95      0.90      0.92      3954
       route       0.95      0.96      0.96      1341
   frequency       0.80      0.80      0.80      1564
    duration       0.79      0.86      0.83       139

   micro avg       0.79      0.76      0.77     12541
```

```
# this code is to fine-tune Med7 with two versions
#en_core_med7_trf en_core_med7_lg

import random
import pandas as pd
import spacy
from spacy import util
from spacy.tokens import Doc
from spacy.training import Example
from spacy.tokens import DocBin
from spacy.language import Language


def customizing_pipeline_component(path, nlp: Language, offset, name):
    file = open(path+"loss_log.txt", "w")

    #optimizer = nlp.create_optimizer()
    optimizer = nlp.resume_training()
    print(type(offset), len(offset))
    print("   Training ...")
    # setup the number of iterations here
    iter = 2
    file.writelines("post-training " + name + "for " + str(iter) + "iterations\n")
    for _ in range(iter):
        print("iteration: " + str(_))
        random.shuffle(offset)
        losses = {}
        for raw_text, entity_offsets in offset: # add character indexes
            entities = spacy.training.offsets_to_biluo_tags(nlp.make_doc(raw_text), entity_offsets)
            #print(entities)
            doc = nlp.make_doc(raw_text)
            example = Example.from_dict(doc, {"entities": entity_offsets})
            #print('[example]', len([example]))
            nlp.update([example], sgd=optimizer, losses=losses)
        print(_, losses)
        file.writelines("iteration: "+ str(_) + str(losses)+"\n")
    file.close()
    # save the post-trained model
    nlp.to_disk(path + name +"_plus")

    # Result after training
    print(f"Result AFTER training:")
    df = pd.read_json(path + "test_76.json", lines=True)

    predict_labels_lg_dic = []

    for i, token in enumerate(df.tokens.to_list()):
      token_str_lg = ' '.join(str(t) for t in token)
```

```
        entities = nlp(token_str_lg)
        for e in entities.ents:
          predict_dic = {}
          predict_dic["predict_file_id"] = i
          predict_dic['predict_label'] = e.label_
          predict_dic['predict_start'] = e.start
          predict_dic['predict_end'] = e.end
          predict_dic['predict_text'] = e.text
          predict_dic['start_char'] = e.start_char
          predict_dic['end_char'] = e.end_char

          predict_labels_lg_dic.append(predict_dic)
        print('file_id:', i)

    print(len(predict_labels_lg_dic))
    predict_label_entity = pd.DataFrame.from_records(predict_labels_lg_dic)
    print(predict_label_entity)

    # uncomment to save the output of the prediction after fine-tuning MED7
    predict_label_entity.to_excel(path + 'test_76_'+name+'_fine_tuned_'+str(iter)+'iterations.xlsx', index=False)

def main():
    input_dir = "/content/drive/MyDrive/collabrations_/HumanLoopH/Med7/data/"
    print("read data")
    json = pd.read_json(input_dir + "train_validation_429.json", lines=True)
    excel = pd.read_excel(input_dir + 'gold_data_entity_CHoffsetEntitiesOnly_train_validation_429.xlsx')
    input_annotations_all = []
    print("processing data")
    for i, token in enumerate(json.tokens.to_list()):
      input_annotations = []
      # create one discharge summary with its NER labels
      token_str_lg = ' '.join(str(t) for t in token)
      file_id =  excel[excel['file_id'] == i]
      for i, row in file_id.iterrows():
        input_annotations.append((int(row[5]), int(row[6]), row[1])) #(start, end, label)
      input_annotations_all.append([token_str_lg, input_annotations])
    print("post-training en_core_med7_lg")
    med7 = spacy.load("en_core_med7_lg")
    customizing_pipeline_component(input_dir, med7, input_annotations_all, "en_core_med7_lg")

    # uncomment th elines below to fine-tune "en_core_med7_trf"
    print("post-training en_core_med7_trf")
    med7 = spacy.load("en_core_med7_trf")
    customizing_pipeline_component(input_dir, med7, input_annotations_all, "en_core_med7_trf")

if __name__ == '__main__':
    main()
```

```
11721              75          Drug            992          993        Lasix

       start_char  end_char
0             108       119
1             120       125
2             150       158
3             734       745
4             748       749
...           ...       ...
11717        6049      6055
11718        6056      6057
11719        6058      6064
11720        6065      6066
11721        6067      6072

[11722 rows x 7 columns]
post-training en_core_med7_trf
<class 'list'> 429
   Training ...
iteration: 0
```

```
# same code as above
# this code is to save the results of prediction in suitable format to calculate the confusion matrix of TP, FN, FP, TN
# using type match (at least part of the token text is annotated with the correct entity type)
# and using strict match (the token text and the entity type has to be matched the gold data)


import pandas as pd

def processing(true_label_entity, predict_label_entity, name, path):

  Eval = []

  predict_label_entity_len = len(predict_label_entity)
  type_list = []
  strict_list = []
  for i, row in true_label_entity.iterrows():
    if i%500 == 0:

      print("batch: ", i)
    tmp = 0
    for c, srow in predict_label_entity.iterrows():
      #print(c)
      if srow[0] > row[0]:
        #print('LARGER', row[0], srow[0])
        break
      #if c%5000 == 0:
        #print("batch c", c)
      Eval_dic = {}
      if row[0] == srow[0]:
        #print('EQUALS', row[0], srow[0])
        if row[5] == srow[5] and row[6] == srow[6] and str(row[4]) == str(srow[4]):
        #if str(row[4]).lower() == str(srow[4]).lower():
          if str(row[1]).lower() == str(srow[1]).lower():
            Eval_dic['file_id'] = str(row[0])
            Eval_dic['true_label'] = str(row[1]).lower()
            Eval_dic['true_start'] = row[5]
            Eval_dic['true_end'] = row[6]
            Eval_dic['true_text'] = str(row[4])
            Eval_dic['predict_file_id'] = str(srow[0])
            Eval_dic['predict_label'] = str(srow[1]).lower()
            Eval_dic['predict_start'] = srow[5]
            Eval_dic['predict_end'] = srow[6]
            Eval_dic['predict_text'] = str(srow[4])
            Eval_dic['strict_label'] = 'COR'
            Eval_dic['type_label'] = 'COR'
            #print('str(row[4]).lower() == str(srow[4]).lower()', Eval_dic)
            Eval.append(Eval_dic)

            strict_list.append('COR')
            type_list.append('COR')
            #print("strict_list.append(COR), type_list.append(COR)")
            #true_label_entity = true_label_entity.drop([i])
            #predict_label_entity = predict_label_entity.drop([c])
            #predict_label_entity_len -= 1
            #tmp = 1
            #break
          elif str(row[1]).lower() != str(srow[1]).lower():
            Eval_dic['file_id'] = str(row[0])
            Eval_dic['true_label'] = str(row[1]).lower()
            Eval_dic['true_start'] = row[5]
            Eval_dic['true_end'] = row[6]
            Eval_dic['true_text'] = str(row[4])
            Eval_dic['predict_file_id'] = str(srow[0])
```

```
            Eval_dic['predict_label'] = str(srow[1]).lower()
            Eval_dic['predict_start'] = srow[5]
            Eval_dic['predict_end'] = srow[6]
            Eval_dic['predict_text'] = str(srow[4])
            Eval_dic['strict_label'] = 'INC'
            Eval_dic['type_label'] = 'INC'
            #print('str(row[4]).lower() == str(srow[4]).lower()', Eval_dic)
            Eval.append(Eval_dic)

            strict_list.append('INC')
            type_list.append('INC')
            #print("strict_list.append(INC), type_list.append(INC)")
          true_label_entity = true_label_entity.drop([i])
          predict_label_entity = predict_label_entity.drop([c])
          predict_label_entity_len -= 1
          tmp = 1
          break

        elif row[5] == srow[5] and str(row[4]) in str(srow[4]):
          if str(row[1]).lower() == str(srow[1]).lower():
            Eval_dic['file_id'] = str(row[0])
            Eval_dic['true_label'] = str(row[1]).lower()
            Eval_dic['true_start'] = row[5]
            Eval_dic['true_end'] = row[6]
            Eval_dic['true_text'] = str(row[4])
            Eval_dic['predict_file_id'] = str(srow[0])
            Eval_dic['predict_label'] = str(srow[1]).lower()
            Eval_dic['predict_start'] = srow[5]
            Eval_dic['predict_end'] = srow[6]
            Eval_dic['predict_text'] = str(srow[4])
            Eval_dic['strict_label'] = 'INC'
            Eval_dic['type_label'] = 'COR'
            #print('row[2] <= srow[2]', Eval_dic)
            Eval.append(Eval_dic)

            type_list.append('COR')
            strict_list.append('INC')
            #print("strict_list.append(INC), type_list.append(COR)")

          elif str(row[1]).lower() != str(srow[1]).lower():
            Eval_dic['file_id'] = str(row[0])
            Eval_dic['true_label'] = str(row[1]).lower()
            Eval_dic['true_start'] = row[5]
            Eval_dic['true_end'] = row[6]
            Eval_dic['true_text'] = str(row[4])
            Eval_dic['predict_file_id'] = str(srow[0])
            Eval_dic['predict_label'] = str(srow[1]).lower()
            Eval_dic['predict_start'] = srow[5]
            Eval_dic['predict_end'] = srow[6]
            Eval_dic['predict_text'] = str(srow[4])
            Eval_dic['strict_label'] = 'INC'
            Eval_dic['type_label'] = 'INC'
            #print('row[2] <= srow[2]', Eval_dic)
            Eval.append(Eval_dic)

            type_list.append('INC')
            strict_list.append('INC')
            #print("strict_list.append(INC), type_list.append(INC)")
          #predict_label_entity = predict_label_entity.drop([c])
          #predict_label_entity_len -= 1
          #if row[3] < srow[3]:
          true_label_entity = true_label_entity.drop([i])
          tmp = 1


        elif row[6] == srow[6] and str(row[4]) in str(srow[4]):
          if str(row[1]).lower() == str(srow[1]).lower():
            Eval_dic['file_id'] = str(row[0])
            Eval_dic['true_label'] = str(row[1]).lower()
            Eval_dic['true_start'] = row[5]
            Eval_dic['true_end'] = row[6]
            Eval_dic['true_text'] = str(row[4])
            Eval_dic['predict_file_id'] = str(srow[0])
            Eval_dic['predict_label'] = str(srow[1]).lower()
            Eval_dic['predict_start'] = srow[5]
            Eval_dic['predict_end'] = srow[6]
            Eval_dic['predict_text'] = str(srow[4])
            Eval_dic['strict_label'] = 'INC'
            Eval_dic['type_label'] = 'COR'
            #print('row[3] <= srow[3]', Eval_dic)
            Eval.append(Eval_dic)
```

```
            type_list.append('COR')
            strict_list.append('INC')
            #print("strict_list.append(INC), type_list.append(COR)")
              #true_label_entity = true_label_entity.drop([i])

              #break
          elif str(row[1]).lower() != str(srow[1]).lower():
              #print("equals", i, c)
            Eval_dic['file_id'] = str(row[0])
            Eval_dic['true_label'] = str(row[1]).lower()
            Eval_dic['true_start'] = row[5]
            Eval_dic['true_end'] = row[6]
            Eval_dic['true_text'] = str(row[4])
            Eval_dic['predict_file_id'] = str(srow[0])
            Eval_dic['predict_label'] = str(srow[1]).lower()
            Eval_dic['predict_start'] = srow[5]
            Eval_dic['predict_end'] = srow[6]
            Eval_dic['predict_text'] = str(srow[4])
            Eval_dic['strict_label'] = 'INC'
            Eval_dic['type_label'] = 'INC'
            #print('row[3] <= srow[3]', Eval_dic)
            Eval.append(Eval_dic)

            type_list.append('INC')
            strict_list.append('INC')
            #print("strict_list.append(INC), type_list.append(INC)")
              #true_label_entity = true_label_entity.drop([i])
          predict_label_entity = predict_label_entity.drop([c])
          predict_label_entity_len -= 1
          true_label_entity = true_label_entity.drop([i])
              #break
          tmp = 1



    if tmp == 0:
      if i in true_label_entity.index:
        Eval_dic = {}

        Eval_dic['file_id'] = str(row[0])
        Eval_dic['true_label'] = str(row[1]).lower()
        Eval_dic['true_start'] = row[5]
        Eval_dic['true_end'] = row[6]
        Eval_dic['true_text'] = str(row[4])
        Eval_dic['predict_file_id'] = str(srow[0])
        Eval_dic['predict_label'] = "O"
        Eval_dic['predict_start'] = row[5]
        Eval_dic['predict_end'] = row[6]
        Eval_dic['predict_text'] = str(row[4])
        Eval_dic['strict_label'] = 'MIS'
        Eval_dic['type_label'] = 'MIS'
        Eval.append(Eval_dic)

        #print(row)
        strict_list.append('MIS')
        type_list.append('MIS')
        #print("strict_list.append(MIS), type_list.append(MIS)")
        true_label_entity = true_label_entity.drop([i])

    #print(len(predict_label_entity), predict_label_entity_len)
  for c, srow in predict_label_entity.iterrows():
    #if len(predict_label_entity) > predict_label_entity_len:
    Eval_dic = {}
    Eval_dic['file_id'] = str(srow[0])
    Eval_dic['true_label'] = 'O'
    Eval_dic['true_start'] = srow[5]
    Eval_dic['true_end'] = srow[6]
    Eval_dic['true_text'] = str(srow[4])
    Eval_dic['predict_file_id'] = str(srow[0])
    Eval_dic['predict_label'] = str(srow[1]).lower()
    Eval_dic['predict_start'] = srow[5]
    Eval_dic['predict_end'] = srow[6]
    Eval_dic['predict_text'] = str(srow[4])
    Eval_dic['strict_label'] = 'SPU'
    Eval_dic['type_label'] = 'SPU'
    Eval.append(Eval_dic)
    #print(srow)
    strict_list.append('SPU')
    type_list.append('SPU')
    #print("strict_list.append(SPU), type_list.append(SPU)")
    predict_label_entity = predict_label_entity.drop([c])
    predict_label_entity_len -= 1
```

```
  true_predict_eval = pd.DataFrame.from_records(Eval)
  print(len(true_predict_eval))
  # if you changed the number of iterations to other than 30 change it in the line below
#  true_predict_eval.to_excel(path+'true_predict_label_entity_76testDataset_'+name+'_fine_tuned_30iterations.xlsx', index=Fals
  true_predict_eval.to_excel(path+'true_predict_label_entity_76testDataset_'+name+'_fine_tuned_2iterations.xlsx', index=False)


path = "/content/drive/MyDrive/collabrations_/HumanLoopH/Med7/data/"

true_label_entity = pd.read_excel(path+'gold_data_entity_CHoffsetEntitiesOnly_test_76.xlsx')
print(true_label_entity)

#model = ["en_core_med7_lg", "en_core_med7_trf"]
model = ["en_core_med7_lg"]
for name in model:
  print("processing the output of "+name+" predictions over the testing set")

  # make sure the number of iterations is correct in the file name
  # if you changed the number of iterations to other than 30 change it in the line below
#  predict_label_entity = pd.read_excel(path + 'test_76_'+name+'_fine_tuned_30iterations.xlsx') #predict_label_entity_76testDa
  predict_label_entity = pd.read_excel(path + 'test_76_'+name+'_fine_tuned_2iterations.xlsx') #predict_label_entity_76testData
  print(predict_label_entity)
  processing(true_label_entity, predict_label_entity, name, path)
```

```
12540                         Lasix      6067    6072

[12541 rows x 7 columns]
processing the output of en_core_med7_lg predictions over the testing set
       predict_file_id predict_label  predict_start  predict_end predict_text  \
0                    0          Drug             21           22  Penicillins
1                    0          Drug             22           23        Sulfa
2                    0          Drug             25           26     Lamictal
3                    0          Drug            115          116  Clindamycin
4                    0        Dosage            117          118            1
...                ...           ...            ...          ...          ...
11717               75          Drug            988          989       Motrin
11718               75      Strength            989          990            3
11719               75          Drug            990          991       Celexa
11720               75      Strength            991          992            4
11721               75          Drug            992          993        Lasix

       start_char  end_char
0             108       119
1             120       125
2             150       158
3             734       745
4             748       749
...           ...       ...
11717        6049      6055
11718        6056      6057
11719        6058      6064
11720        6065      6066
11721        6067      6072

[11722 rows x 7 columns]
batch:  0
batch:  500
batch:  1000
batch:  1500
batch:  2000
batch:  2500
batch:  3000
batch:  3500
batch:  4000
batch:  4500
batch:  5000
batch:  5500
batch:  6000
batch:  6500
batch:  7000
batch:  7500
batch:  8000
batch:  8500
batch:  9000
batch:  9500
batch:  10000
batch:  10500
batch:  11000
batch:  11500
batch:  12000
batch:  12500
13386
```

```
# same code as above
# This code is to calculate P, R, F1 scores based on matching the entity type between reference and candidate.
# support in the output is the number of reference per entity type.

# DONOT FORGET TO CHANGE PATHS AND FILES NAMES

import pandas as pd
from sklearn.metrics import classification_report


def class_report(y_true, y_pred, path, name):

  file = open(path+"prediction_results_after_fine-tuning_"+name+".txt", 'w')
  labels = ['reason', 'ade', 'form', 'strength', 'dosage', 'drug', 'route', 'frequency', 'duration']
  print(type(labels), labels)
  report = classification_report(y_true, y_pred, labels = labels)
  print(report)
  file.writelines(report)

  labels.remove('reason') # remove 'reason' label from evaluation
  labels.remove('ade') # remove 'reason' label from evaluation
  print(type(labels), labels)
  report = classification_report(y_true, y_pred, labels = labels)
  print(report)
  file.writelines(report)
  file.close()


  #return report


path = "/content/drive/MyDrive/collabrations /HumanLoopH/Med7/data/"
#model = ["en_core_med7_lg", "en_core_med7_trf"]
model = ["en_core_med7_lg"]
for name in model:
#  print("caluculating P, R, F1 scores of "+name+" predictions over the testing set AFTER fine-tuning on 429 n2c2-2018 trainin
#  eval_report_all = pd.read_excel(path+'true_predict_label_entity_76testDataset_'+name+'_fine_tuned_30iterations.xlsx') #pred
  print("caluculating P, R, F1 scores of "+name+" predictions over the testing set AFTER fine-tuning on 429 n2c2-2018 training
  eval_report_all = pd.read_excel(path+'true_predict_label_entity_76testDataset_'+name+'_fine_tuned_2iterations.xlsx') #predic
  print("76 Test dataset "+name)
  eval_report_copy_all = eval_report_all.copy()
  y_true = eval_report_copy_all['true_label'].tolist()
  y_pred = eval_report_copy_all['predict_label'].tolist()
  print('type matching')
  #all = class_report(y_true, y_pred)
  class_report(y_true, y_pred, path, name)
```

```
caluculating P, R, F1 scores of en_core_med7_lg predictions over the testing set AFTER fine-tuning on 429 n2c2-2018 trair
76 Test dataset en_core_med7_lg
type matching
<class 'list'> ['reason', 'ade', 'form', 'strength', 'dosage', 'drug', 'route', 'frequency', 'duration']
              precision    recall  f1-score   support

      reason       0.78      0.37      0.50       927
         ade       0.86      0.05      0.09       242
        form       0.94      0.95      0.95      1696
    strength       0.95      0.97      0.96      1639
      dosage       0.90      0.93      0.91      1039
        drug       0.93      0.93      0.93      3954
       route       0.96      0.95      0.95      1341
   frequency       0.85      0.95      0.90      1564
    duration       0.88      0.58      0.70       139

   micro avg       0.92      0.88      0.90     12541
   macro avg       0.89      0.74      0.77     12541
weighted avg       0.91      0.88      0.88     12541

<class 'list'> ['form', 'strength', 'dosage', 'drug', 'route', 'frequency', 'duration']
              precision    recall  f1-score   support

        form       0.94      0.95      0.95      1696
    strength       0.95      0.97      0.96      1639
      dosage       0.90      0.93      0.91      1039
        drug       0.93      0.93      0.93      3954
       route       0.96      0.95      0.95      1341
   frequency       0.85      0.95      0.90      1564
    duration       0.88      0.58      0.70       139

   micro avg       0.92      0.94      0.93     11372
   macro avg       0.92      0.90      0.90     11372
weighted avg       0.92      0.94      0.93     11372
```

```
# All code below is for pre-process Brat annotaion (NER)
# skip if you don't need


!python /content/drive/MyDrive/Reasearch_Assistantship/HILO_project/brat2CoNLL-main/brat2CoNLL/format_convertor2.py --input_di
!python /content/drive/MyDrive/Reasearch_Assistantship/HILO_project/brat2CoNLL-main/brat2CoNLL/format_convertor2.py --input_di
!python /content/drive/MyDrive/Reasearch_Assistantship/HILO_project/brat2CoNLL-main/brat2CoNLL/format_convertor2.py --input_di


# Edit pre-process
import pandas as pd

df = pd.read_excel('/content/drive/MyDrive/Reasearch_Assistantship/HILO_project/n2c2_2018/train/output/output.xlsx')
print(df.columns)

df_copy = df.copy()
line = df_copy.index[df_copy['Unnamed: 2'] == "O"].tolist()
print(len(df_copy))
print(len(line))
df2 = pd.DataFrame(columns = ['Token', 'Label', 'Unnamed: 2'])
for id in line:
  tmp = df_copy.iloc[id]
  #print(tmp[2])

  test = pd.DataFrame({'Token' : tmp[0], 'Label' : tmp[2], 'Unnamed: 2' : pd.NA}, index=[id])
  df2 = pd.concat([df2, test], axis=0, ignore_index=False)
  #df2 = df2.append(test, ignore_index=False)
  test = pd.DataFrame({'Token' : tmp[1], 'Label' : tmp[2], 'Unnamed: 2' : pd.NA}, index=[id+1])
  df2 = pd.concat([df2, test], axis=0, ignore_index=False)
  #df2 = df2.append(test, ignore_index=False)

  #print(tmp)
print(len(df2))
print(df_copy.loc[[29490]])
df_copy = df_copy.drop(line)
print(len(df_copy))
line = df2.iloc[0]
print(line)
line = df2.iloc[1]
print(line)
# https://huggingface.co/course/chapter7/2
df3 = pd.concat([df2, df_copy], axis=0)
print(df3)
df3.to_excel('/content/drive/MyDrive/Reasearch_Assistantship/HILO_project/n2c2_2018/train/output/output2.xlsx', index=False)



# split all discharge sammaries into seperate files, a file for a discharge summary

df = pd.read_excel('/content/drive/MyDrive/Reasearch_Assistantship/HILO_project/n2c2_2018/valid/output/output.xlsx', index=Fal
print(len(df))
print(df.columns)

df_copy = df.copy()
line = []
for i, row in df_copy.iterrows():
  #print(i, row[0], row[1])
  if ".txt" in row[1]:
    print(i, row[0], row[1])
    line.append(i)
print(len(line))
print(line)
i = 0
while i+1 <= len(line):
  if i+1 == len(line):
    df_slice = df_copy.iloc[line[i]:len(df_copy), :]
    txt_ind = df_slice["Label"].tolist()
    print(txt_ind[0])
    df_slice.to_excel('/content/drive/MyDrive/Reasearch_Assistantship/HILO_project/n2c2_2018/valid/output/'+txt_ind[0]+'.xlsx'
  else:
    df_slice = df_copy.iloc[line[i]:line[i+1], :]
    txt_ind = df_slice["Label"].tolist()
    print(txt_ind[0])
    df_slice.to_excel('/content/drive/MyDrive/Reasearch_Assistantship/HILO_project/n2c2_2018/valid/output/'+txt_ind[0]+'.xlsx'
  i +=1
```

✓ 5 秒 完成时间：16:30