Imports

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

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

```
!python -m spacy download
# maybe try en_core_sci_lg that is biomedical ? https://allenai.github.io/scispacy/
```

```
Collecting en-core-web-lg==3.7.1
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_lg-3.7.1/en_core_web_lg-3.7.1-py3-
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 587.7/587.7 MB 2.6 MB/s eta 0:00:00
Requirement already satisfied: spacy<3.8.0,>=3.7.2 in /usr/local/lib/python3.10/dist-packages (from en-core-web-lg==3.7.
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2-
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.
Requirement already satisfied: thinc<8.3.0,>=8.2.2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2-
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2-
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.
Requirement already satisfied: weasel<0.4.0,>=0.1.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2
Requirement already satisfied: typer<0.10.0,>=0.3.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2-
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=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<3.8.0,>=3.7.2->en-core-web-
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-core-
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.8.0,>=3.7.2->en-co
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic!=1.8,!=1
Requirement already satisfied: pydantic-core==2.16.3 in /usr/local/lib/python3.10/dist-packages (from pydantic!=1.8,!=1.
Requirement already satisfied: typing-extensions>=4.6.1 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->sp
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<8.3.0,>=8.2.2->
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.10/dist-packages (from thinc<8.3.0,>=8
Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/python3.10/dist-packages (from typer<0.10.0,>=0.3.0
Requirement already satisfied: cloudpathlib<0.17.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from weasel<0.4.0
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->spacy<3.8.0,>=3.
Installing collected packages: en-core-web-lg
Successfully installed en-core-web-lg-3.7.1
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_lg')
⚠ Restart to reload dependencies
If you are in a Jupyter or Colab notebook, you may need to restart Python in
order to load all the package's dependencies. You can do this by selecting the
'Restart kernel' or 'Restart runtime' option.
```

```
import pandas as pd
import pickle

from torch.utils.data import DataLoader
import random
import numpy as np
import pandas as pd

from tqdm import tqdm

from spacy.training import Example
import random

import spacy
import os
import datetime
from google.colab import userdata, runtime
import re

import warnings

import nltk
from nltk.corpus import stopwords
import string

import spacy
from spacy.training import Example
import random
from tqdm import tqdm
import copy

nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

Load Dataset & Model

```
DATASET_DIR = '/content/drive/My Drive/_collaborations_/Synda_Health_Mile2/Dataset/' # data set, e.g. free text for deployme
GENERATIONS_DIR = '/content/drive/My Drive/_collaborations_/Synda_Health_Mile2/EVALUATION/Medical_Generations/'
SAVE_DIR = '/content/drive/My Drive/_collaborations_/Synda_Health_Mile2/EVALUATION/Downstream_Results/Medical/'

training_set = pd.read_csv(DATASET_DIR + 'train.csv')
# generations = pd.read_csv('path/to/generations')
validation_set = pd.read_csv(DATASET_DIR + 'valid.csv')
test_set = pd.read_csv(DATASET_DIR + 'test.csv')

MASKING_RATIO = 0.5
MODEL_NAME = 'roberta-large'

REAL_EVALUATE = True
GENERATIONS_EVALUATE = True

stop_words = set(stopwords.words('english'))
punctuation = set(string.punctuation)

RUN_NAME = ["roberta-large - Run - 02-11--14:40", "BiomedNLP-BiomedBERT-large-uncased-abstract - Run Medical - 03-02--18:56"


nlp_sci = spacy.load("/content/drive/My Drive/_collaborations_/Synda_Health_Mile2/NER_Models/en_core_sci_lg")
```

```
/usr/local/lib/python3.10/dist-packages/spacy/util.py:910: UserWarning: [W095] Model 'en_core_sci_lg' (0.5.3) was traine
  warnings.warn(warn_msg)
/usr/local/lib/python3.10/dist-packages/spacy/util.py:1740: UserWarning: [W111] Jupyter notebook detected: if using `pre
  warnings.warn(Warnings.W111)
```

```
data_train = training_set['Clinical Letters']
data_valid = validation_set['Clinical Letters']
data_test = test_set['Clinical Letters']
```

Get Entities & Train the Model

```
data_train
```

```
0      Please shower daily including washing incision...
1      * Increasing pain * Fever * Inability to eat o...
2      You were admitted to the hospital with a react...
```

```
3      Dear Mr. Known lastname , You were admitted to...
4      Please note: you have a mm nodule that was not...
                              ...
202    Please call the Hospital Clinic at Telephone/F...
203    Dr. Known lastname , it was a pleasure to part...
204    please call the Transplant Office Telephone/Fa...
205                                          To Hospital
206    # You were admitted to the hospital for shortn...
Name: Clinical Letters, Length: 207, dtype: object
```

```
def get_entities(data):
    all_entities = []

    for doc in nlp_sci.pipe(data):
        entities = [(ent.start_char, ent.end_char, ent.label_) for ent in doc.ents]
        all_entities.append((doc.text, {'entities': entities}))

    return all_entities


# if len(generations.keys()) < 10:
#     warnings.warn(f"generations keys length: {len(generations.keys())}")

entities = dict()

# entities['real'] = get_entities(data_train)
generations_dict = dict()

# with open(GENERATIONS_DIR + RUN_NAME + ".pkl", "rb") as f:
#     model_generations = pickle.load(f)

# for option in model_generations.keys():
#     if option == 'random':
#         for key in model_generations[option].keys():
#             generations = model_generations[option][key]
#             generations_dict[key] = generations
#     else:
#         generations = model_generations[option]
#         generations_dict[option] = generations

# for key in generations_dict.keys():
#     entities[key] = get_entities(list(generations_dict[key].values()))

entities['train'] = get_entities(data_train)
# entities['valid'] = get_entities(data_valid)
# entities['test'] = get_entities(data_test)


entities.keys()
```

```
dict_keys(['train'])
```

```
def evaluate_model(model, validation_data):
    losses = {}
    for text, annotations in validation_data:
        doc = model.make_doc(text)
        example = Example.from_dict(doc, annotations)
        model.update([example], drop=0.0, losses=losses, sgd=None)

    return losses['ner']

EPOCHS = 3

def train_model(training_data, validation_data):
    model = spacy.load("en_core_web_lg")

    if "ner" not in model.pipe_names:
        ner = model.create_pipe("ner")
        model.add_pipe(ner, last=True)
    else:
        ner = model.get_pipe("ner")

    ner.add_label('ENTITY')
    other_pipes = [pipe for pipe in model.pipe_names if pipe != "ner"]

    with model.disable_pipes(*other_pipes):
        optimizer = model.resume_training()
        best_loss = float('inf')
        best_model = None

        for epoch in range(EPOCHS): # setup epoch num
            random.shuffle(training_data)
            losses = {}

            for text, annotations in training_data:
                doc = model.make_doc(text) #model to be trained
                example = Example.from_dict(doc, annotations) # pack predication and expected/gold labels
                model.update([example], drop=0.5, losses=losses, sgd=optimizer) # the loss to be minimised

            val_loss = evaluate_model(model, validation_data) # validation - not really need in some situration

            if val_loss < best_loss:
                best_loss = val_loss
                best_model = copy.deepcopy(model)

    return best_model # return the best model that can be deployed.


models = dict()

for key in tqdm(list(entities.keys())):
    if key in ['valid', 'test']:
        continue

    models[key] = train_model(training_data=entities[key], validation_data=entities['train'])
```

```
  0%|          | 0/1 [00:00<?, ?it/s]/usr/local/lib/python3.10/dist-packages/spacy/util.py:1740: UserWarning: [W111] Jup
  warnings.warn(Warnings.W111)
100%|██████████| 1/1 [01:45<00:00, 105.02s/it]
```

Evaluate The Model

You were admitted with severe pancreatitis from gallstones. You were treated supportively during this and you improved. You required intubation twice to support your breathing. You were treated for pneumonia, acute kidney failure and aspiration. You are now improving and being sent to an acute rehabilitation facility. In the future it will be important for you to have a cholecystectomy in - months, once you have completely recovered from this hospitalization. You will also need to have follow up imaging of your lungs in months given the pulmonary nodules found during this admission. Some of your medications have changed: We have stopped actos and metformin. These were changed to insulin while you were sick. Once you go home you can restart these. We have stopped amlodipine. We have halved your lisinopril. We have started trazodone, simethicone, senna, colace, bisacodyl, insulin and lidocaine patch.

```
def evaluate_model(model):
    correct = 0
    predicted_total = 0
    actual_total = 0

    for text, original_annotation in zip(training_set['Clinical Letters'], entities['train']):
        doc = model(text) # in real setting, this returns the labels of input text by deploying saved best model.
        predicted_entities = set((ent.text, ent.label_) for ent in doc.ents) # extracting entities attached to text
        original_entities_set = set((text[start:end], label) for start, end, label in original_annotation[1]['entities']) #
        correct += len(predicted_entities.intersection(original_entities_set))
        predicted_total += len(predicted_entities)
        actual_total += len(original_entities_set)

    precision = correct / predicted_total if predicted_total > 0 else 0
    recall = correct / actual_total if actual_total > 0 else 0
    f1 = 2 * (precision * recall) / (precision + recall) if (precision + recall) > 0 else 0

    print(f"Precision: {precision}")
    print(f"Recall: {recall}")
    print(f"F1 Score: {f1}")

    return f1


results = dict()

for key in tqdm(models.keys()):
    results[key] = evaluate_model(models[key])

results
```

```
100%|██████████| 1/1 [00:02<00:00,  2.17s/it]Precision: 0.030669285390542377
Recall: 0.037894343849169 31
F1 Score: 0.033901135604542415

{'train': 0.033901135604542415}
```

```
doc = models['train']("You were admitted with severe pancreatitis from gallstones. You were treated supportively during this
```

```
list((ent.text, ent.label_) for ent in doc.ents)
```

```
[('admitted with', 'ENTITY'),
 ('severe', 'ENTITY'),
 ('pancreatitis', 'ENTITY'),
 ('gallstones', 'ENTITY'),
 ('treated supportively', 'ENTITY'),
 ('improved', 'ENTITY'),
 ('intubation', 'ENTITY'),
 ('breathing', 'ENTITY'),
 ('treated', 'ENTITY'),
 ('pneumonia', 'ENTITY'),
 ('acute kidney failure', 'ENTITY'),
 ('aspiration', 'ENTITY'),
 ('improving', 'ENTITY'),
 ('acute rehabilitation facility', 'ENTITY'),
 ('cholecystectomy', 'ENTITY'),
 ('months', 'ENTITY'),
 ('recovered', 'ENTITY'),
 ('hospitalization', 'ENTITY'),
 ('follow up', 'ENTITY'),
 ('imaging', 'ENTITY'),
 ('lungs', 'ENTITY'),
 ('months', 'ENTITY'),
 ('pulmonary nodules', 'ENTITY'),
 ('admission', 'ENTITY'),
 ('medications', 'ENTITY'),
 ('actos', 'ENTITY'),
 ('metformin', 'ENTITY'),
 ('insulin', 'ENTITY'),
 ('sick', 'ENTITY'),
 ('restart', 'ENTITY'),
 ('amlodipine', 'ENTITY'),
 ('halved', 'ENTITY'),
 ('lisinopril', 'ENTITY'),
 ('trazodone', 'ENTITY'),
 ('simethicone', 'ENTITY'),
 ('senna', 'ENTITY'),
 ('colace', 'ENTITY'),
 ('bisacodyl', 'ENTITY'),
 ('insulin', 'ENTITY'),
 ('lidocaine patch', 'ENTITY')]
```

```
list((ent.text) for ent in doc.ents)
```

```
['admitted with',
 'severe',
 'pancreatitis',
 'gallstones',
 'treated supportively',
 'improved',
```