CSE4022 – Natural Language Processing
Final Report

Biomedical Text Mining (Acronym and Text Extraction) Doctor Helper

Parth Maheshwari – 19BCT0221
Pramit Gupta – 19BCT0204
Rohit Ravichandran – 19BCT0014

Abstract

Doctor-helper is a two way information extraction/biomedical text-mining project. It uses Python as a base and many NLP related libraries such as spaCy for extracting information. It uses the standard 6-step process to process raw unstructured data into formattable structured data. First it takes input from the patient (user1-patient), processes and converts it into a tabular format with some mandatory fields such as patient name, age and contact number, and other fields like address, symptoms, body-temprature, etc which is achieved by a modified parts of speech tagging. It will also contain a field which displays the possible ailments which the patient could be suffering from, based on probability calculated using symptoms observed, weather of the address, time of year, last known cases/outbreaks. The doctor can use this information and it will be stored in a patient-database. In the second stage, the doctor will write a report which will be taken as input (user2-doctor), processed and converted it into a tabular format which is easier to read and be understood by the patient. It will contain fields like ailments(confirmed and suspected), tests to be taken(blood test, etc.), drugs prescribed, dosage, side effects, alternative drugs, etc. It will also give the patient an option to add a reminder for each dosage using the Google Calender API.

Introduction

In these trying times, the importance of doctors worldwide has risen and so have their responsibilites. COVID-19, being highly contagious, has significantly brought down the accessibility of medical help to patients suffering from ailments other than COVID-19. Whether it be due to medical professionals being engaged by COVID-19 patients or due to the person being afraid to seek help from a clinic/hospital from fear of contracting the infamous corona virus, this has directly affected the patients who have contracted other ailments.

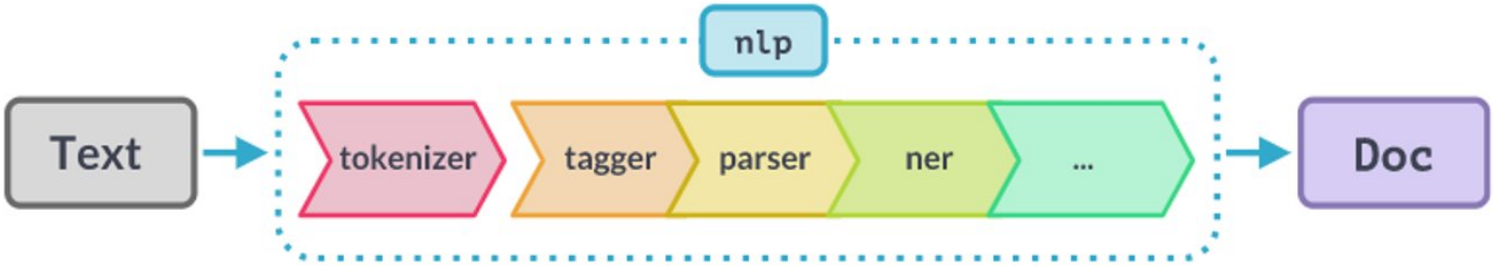
Doctor-helper aims to ease the load on medical professionals by making their task much easier. It takes input from the patient and extracts the patient details, various symptoms present, etc. and detects the possible ailments that the patient may be suffering from(which is then confirmed by the doctor). Then it takes the doctors report as input and displays the output in a tabular format, containing the drugs prescribed, dosage, precautions, side-effects, alternative drugs, etc. It will also give an option to set a reminder for each dose using the Google Calender.

This application will help streamline the process of consulting the doctor and save their time on unnecessary paperwork by automating most of it. It will aid the doctors judgement and also help with online consultation from medical professionals without the patient risking contact with individuals affected from COVID-19 and following lockdown mandates.

Problem Statement

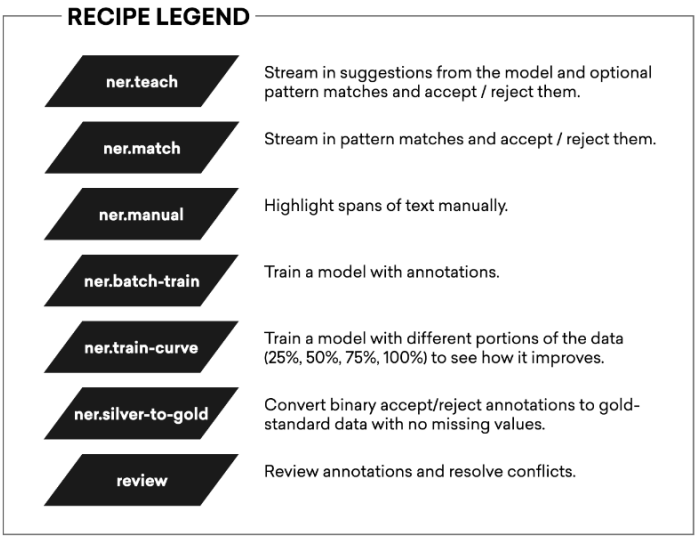
- We are expecting to implement basic concepts of Text Mining and NLP in this review.
- We are going to play with the small CORPUS i.e. data-set we have right now.
- We have made sure that the data set is complicated enough, i.e. we have taken random data from different journals.
- In this review we are going to label the DRUG and ILLNESS NAMES, that are present in the data-set.
- We are going to use MACHINE LEARNING, NLP and TEXT MINING to achieve so.

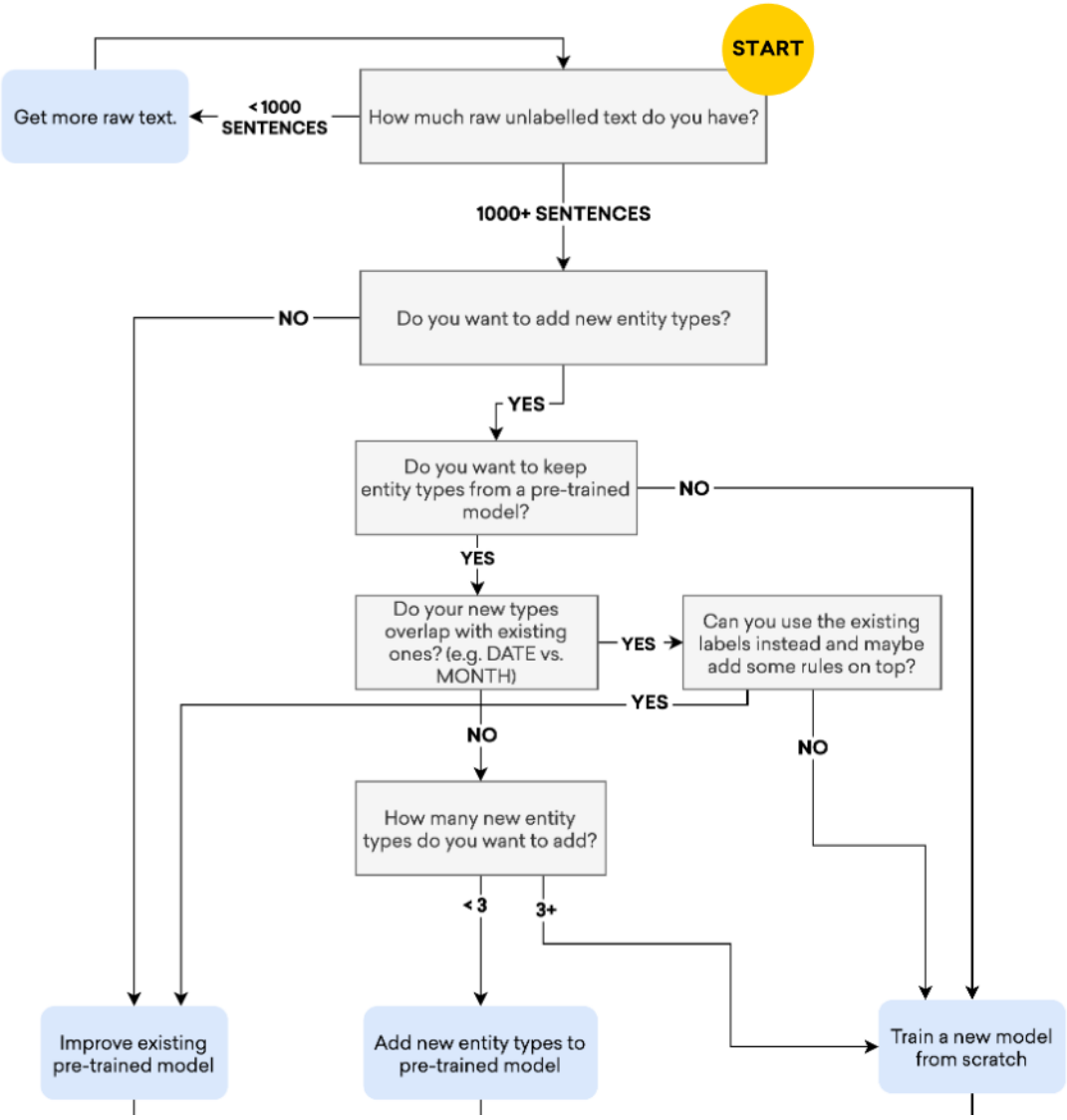
Architecture Diagram



NAME	COMPONENT	CREATES	DESCRIPTION
tokenizer	`Tokenizer`	`Doc`	Segment text into tokens.
PROCESSING PIPELINE			
tagger	`Tagger`	`Token.tag`	Assign part-of-speech tags.
parser	`DependencyParser`	`Token.head`, `Token.dep`, `Doc.sents`, `Doc.noun_chunks`	Assign dependency labels.
ner	`EntityRecognizer`	`Doc.ents`, `Token.ent_iob`, `Token.ent_type`	Detect and label named entities.
lemmatizer	`Lemmatizer`	`Token.lemma`	Assign base forms.
textcat	`TextCategorizer`	`Doc.cats`	Assign document labels.
custom	custom components	`Doc._.xxx`, `Token._.xxx`, `Span._.xxx`	Assign custom attributes, methods or properties.

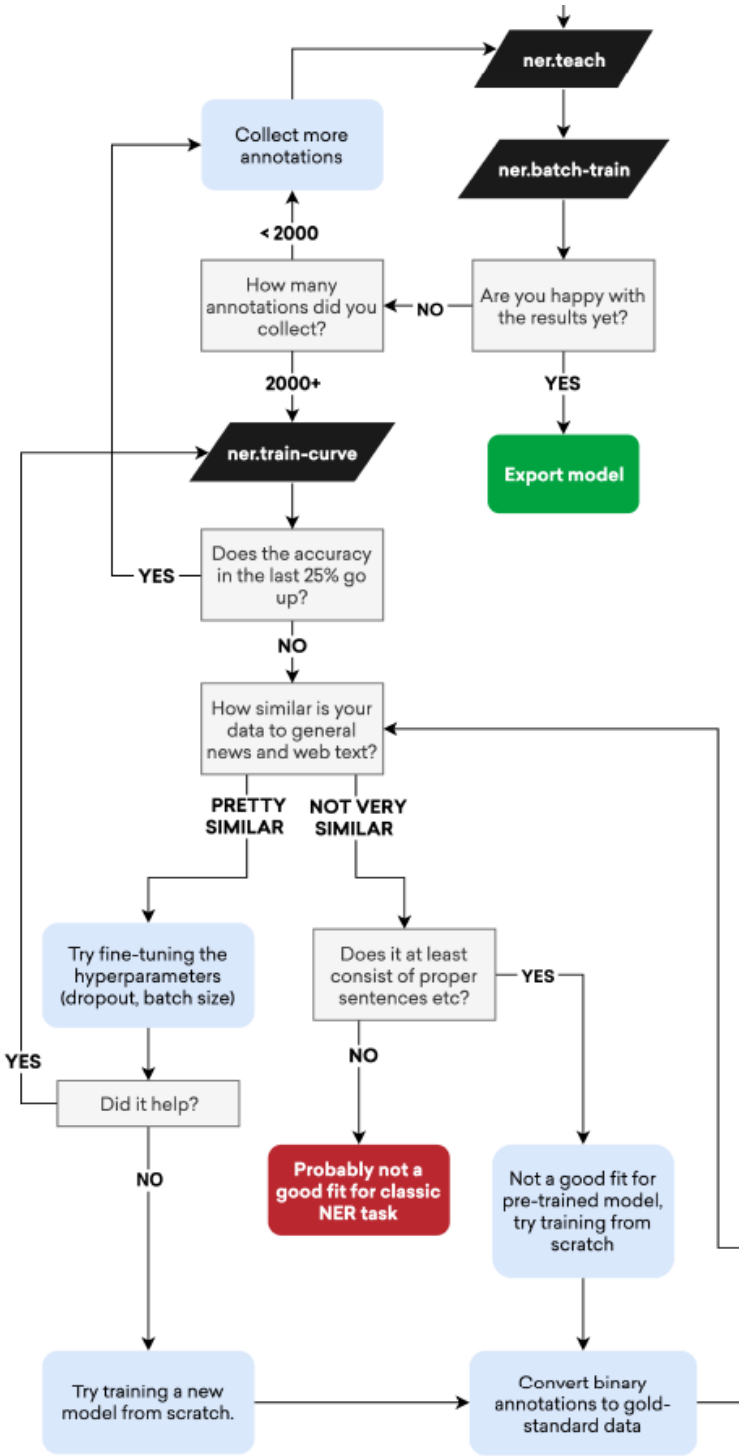
Flow Diagram



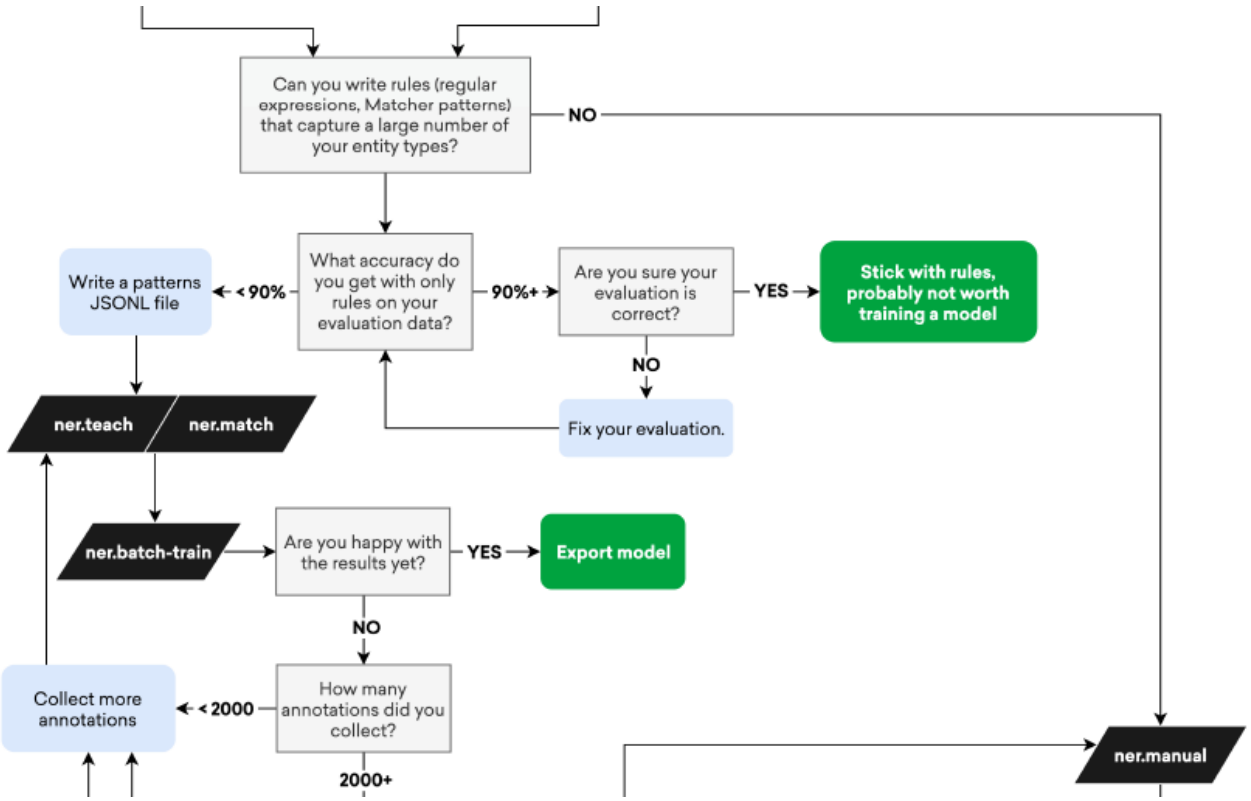


p1 p2

♦P1

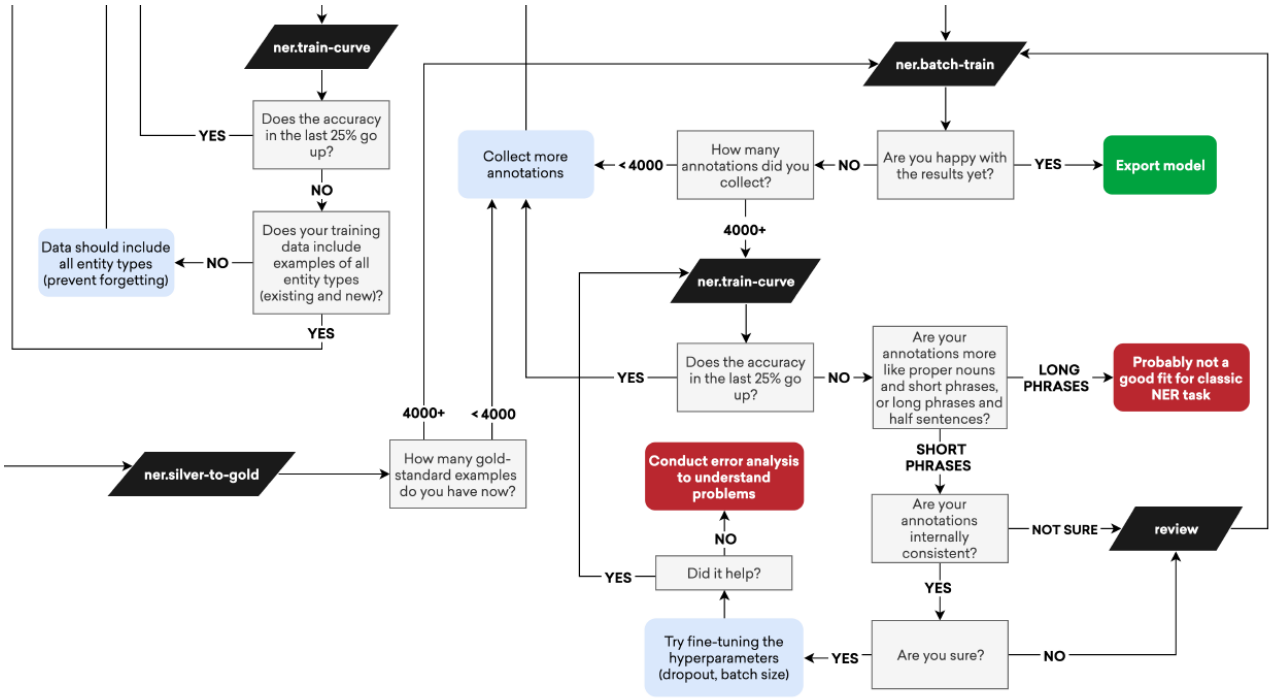

ner.teach
ner.batch-train
Are you happy with the results yet?
NO
How many annotations did you collect?
< 2000
Collect more annotations
2000+
YES
Export model
ner.train-curve
Does the accuracy in the last 25% go up?
YES
NO
How similar is your data to general news and web text?
PRETTY SIMILAR
NOT VERY SIMILAR
Try fine-tuning the hyperparameters (dropout, batch size)
Does it at least consist of proper sentences etc?
YES
NO
YES
Did it help?
NO
Probably not a good fit for classic NER task
Not a good fit for pre-trained model, try training from scratch
Try training a new model from scratch.
Convert binary annotations to gold-standard data

♦P2


Can you write rules (regular expressions, Matcher patterns) that capture a large number of your entity types?
NO
What accuracy do you get with only rules on your evaluation data?
< 90%
Write a patterns JSONL file
90%+
Are you sure your evaluation is correct?
YES
Stick with rules, probably not worth training a model
NO
Fix your evaluation.
ner.teach
ner.match
ner.batch-train
Are you happy with the results yet?
YES
Export model
NO
How many annotations did you collect?
< 2000
Collect more annotations
2000+
ner.manual

p3

♦P3


ner.train-curve
Does the accuracy in the last 25% go up?
YES
NO
Does your training data include examples of all entity types (existing and new)?
NO
Data should include all entity types (prevent forgetting)
YES
ner.silver-to-gold
How many gold-standard examples do you have now?
4000+
< 4000
Collect more annotations
ner.batch-train
Are you happy with the results yet?
YES
Export model
NO
How many annotations did you collect?
< 4000
4000+
ner.train-curve
Does the accuracy in the last 25% go up?
YES
NO
Are your annotations more like proper nouns and short phrases, or long phrases and half sentences?
LONG PHRASES
Probably not a good fit for classic NER task
SHORT PHRASES
Are your annotations internally consistent?
NOT SURE
review
YES
Are you sure?
NO
YES
Try fine-tuning the hyperparameters (dropout, batch size)
Did it help?
NO
Conduct error analysis to understand problems
YES

Pseudocode

```
>>
import required packages
import data set and convert to data frame
create object for spaCy using default english corpus
pass each sentence in data frame through object for processing
using nlp(tokenizer→tagger→parser→ner→postprocessing)
import sample drug corpus and convert to data frame
apply spaCy ner pipeline with new corpus to processed data
using loop, identify index position of each drug mentioned and tag
train data using loop and use displaCy to display result
do the same for illnesses using illness.txt corpus
```

Experiment

- Data set
  - taken from various medical journals and medical reports
  - contains patient descriptions, medical prescriptions, blogs about medical health, etc.

dataset.txt - Notepad

File Edit Format View Help

Acne is a skin condition caused by dead skin cells sticking together and clogging up pores. Bacteria can play a role, too. A big trigger for the onset of acne is puberty.
Doxycycline is a tetracycline antibiotic that fights bacteria in the body. Doxycycline is used to treat many different bacterial infections, such as acne, urinary tract infections, intestinal infections, respiratory infections, eye infections, gonorrhea, chlamydia, syphilis, periodontitis (gum disease), and others.
Spironolactone, sold under the brand name Aldactone among others, is a medication that is primarily used to treat fluid build-up due to heart failure, liver scarring, or kidney disease.
Minocycline sold under the brand name Minocin among others, is a tetracycline antibiotic used to treat a number of bacterial infections such as pneumonia.
Aldactone is a prescription medicine used to treat the symptoms of high blood pressure (hypertension), low potassium levels (hypokalemia) and heart failure. Aldactone may be used alone or with other medications.
Clindamycin is an antibiotic used for the treatment of a number of bacterial infections, including bone or joint infections, pelvic inflammatory disease, strep throat, pneumonia, middle ear infections, and endocarditis.
Tazarotene sold under the brand name Tazorac, among others, is a third-generation prescription topical retinoid.
Tretinoin is a medication used to treat acne and sun-damaged skin. It can't erase deep wrinkles, but it can help improve the appearance of surface wrinkles, fine lines, and darks spots.
Isotretinoin also known as 13-cis-retinoic acid and sold under the brand name Accutane among others, is a medication primarily used to treat severe acne.
Aczone Gel is a prescription medicine used to treat the symptoms of Acne Vulgaris. Aczone Gel may be used alone or with other medications.
Adapalene is a third-generation topical retinoid primarily used in the treatment of mild-moderate acne, and is also used off-label to treat keratosis pilaris as well as other skin conditions.
Doryx is a tetracycline antibiotic that fights bacteria in the body. Doryx is used to treat many different bacterial infections, such as acne, urinary tract infections, intestinal infections, eye infections, gonorrhea, chlamydia, periodontitis (gum disease), and others.
Cefalexin also spelled cephalexin, is an antibiotic that can treat a number of bacterial infections. It kills gram-positive and some gram-negative bacteria by disrupting the growth of the bacterial cell wall.
Tetracycline sold under the brand name Sumycin among others, is an oral antibiotic in the tetracyclines family of medications, used to treat a number of infections, including acne, cholera, brucellosis, plague, malaria, and syphilis.
Vibramycin is a tetracycline antibiotic that fights bacteria in the body. Vibramycin is used to treat many different bacterial infections, such as acne, urinary tract infections, intestinal infections, eye infections, gonorrhea, chlamydia, periodontitis (gum disease), and others.
ACTICLATE (doxycycline hyclate) Tablets and ACTICLATE CAP (doxycycline hyclate) Capsules contain doxycycline hyclate, a tetracycline class drug synthetically derived from oxytetracycline, in an immediate release formulation for oral administration.
Attention deficit hyperactivity disorder (ADHD) is a common behavioral disorder diagnosed in roughly 10 percent of

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

- Corpus
  - corpus for ner tagging
  - contains list of 20000+ pharmaceutical drugs

drugs.txt - Notepad

File Edit Format View Help

```
doxycycline
spironolactone
minocycline
Aldactone
clindamycin
Tazorac
tazarotene
tretinoin
isotretinoin
Bactrim
Epiduo
Aczone
Differin
benzoyl peroxide
adapalene
Doryx
Septra
Solodyn
cephalexin
tetracycline
Vibramycin
Acticlate
Vyvanse
Adderall
Concerta
amantadine
methylphenidate
Strattera
Dexedrine
Ritalin
clonidine
Intuniv
bupropion
guanfacine
Focalin
Desoxyn
lisdexamfetamine
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

- contains list of illness

illness.txt - Notepad

File Edit Format View Help

```
Acne
ADHD
AIDS/HIV
Allergies
Alzheimer's
Angina
Anxiety
Arthritis
Asthma
Bipolar Disorder
Bronchitis
Cancer
Cholesterol
Cold
Flu
Constipation
COPD
Covid 19
Depression
Diabetes (Type 1)
Diabetes (Type 2)
Diarrhea
Eczema
Erectile Dysfunction
Fibromyalgia
Gastrointestinal
GERD (Heartburn)
Gout
Hair Loss
Hayfever
Heart Disease
Herpes
Hypertension
Hypothyroidism
IBD (Bowel)
Incontinence
Insomnia
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

Code:

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import re
import random

df = pd.read_csv('sample_data/dataset.txt', sep='\t',
names=["info"])

df.head()

import spacy
from spacy import displacy
nlp = spacy.load('en_core_web_sm')

for sentence in df['info']:
  print("Sentence is: ", sentence)
  sentence_doc = nlp(sentence)
  displacy.render(sentence_doc,style='ent',jupyter=True)

df_drug = pd.read_csv('sample_data/drugs.txt', sep='\t',
names=["drug"])

df_drug.head()

df_illness = pd.read_csv('sample_data/illness.txt',
sep='\t', names=["illness"])

df_illness.head()

import spacy
from spacy.util import minibatch, compounding
```

```
nlp0 = spacy.load('en_core_web_sm')

nlp0.pipe_names

ner0 = nlp0.get_pipe('ner')

all_drugs = df_drug['drug'].unique().tolist()
all_drugs = [x.lower() for x in all_drugs]

all_drugs

all_illness = df_illness['illness'].unique().tolist()
all_illness = [x.lower() for x in all_illness]

all_illness

df['info']

def process_review(review):
    processed_token = []
    for token in review.split():
        token = ''.join(e.lower() for e in token if
e.isalnum())
        processed_token.append(token)
    return ' '.join(processed_token)

# Training examples in the required format
count = 0
TRAIN_DATA = []
for _,item in df.iterrows():
    ent_dict = {}
    if count < 1000:
        review = process_review(item['info'])
        visited_item = []
        entities = []
        for token in review.split():
          if token in all_drugs:
            for i in re.finditer(token, review):
                if token not in visited_item:
                  entity = (i.span()[0], i.span()[1],
'DRUG')
```

```
                    visited_item.append(token)
                    entities.append(entity)
            elif token in all_illness:
              for i in re.finditer(token, review):
                  if token not in visited_item:
                    entity = (i.span()[0], i.span()[1],
'ILLNESS')
                    visited_item.append(token)
                    entities.append(entity)
          if len(entities) > 0:
              ent_dict['entities'] = entities
              train_item = (review, ent_dict)
              TRAIN_DATA.append(train_item)
              count+=1

TRAIN_DATA

# Add the new label to ner
ner0.add_label(LABEL1)
ner0.add_label(LABEL2)

# Resume training
optimizer = nlp0.resume_training()
move_names = list(ner0.move_names)

# List of pipes you want to train
pipe_exceptions = ["ner", "trf_wordpiecer",
"trf_tok2vec"]

# List of pipes which should remain unaffected in
training
other_pipes = [pipe for pipe in nlp0.pipe_names if pipe
not in pipe_exceptions]

n_iter = 100
def train_ner(training_data):
    TRAIN_DATA = training_data
    nlp1 = spacy.blank("en")
    print("Created blank 'en' model")

    if "ner" not in nlp1.pipe_names:
```

```
      print("ner is created")
      ner = nlp1.create_pipe("ner")
      nlp1.add_pipe(ner, last=True)

    else:
        ner = nlp1.get_pipe("ner")

    for _,annotations in TRAIN_DATA:
        for ent in annotations.get("entities"):
            ner.add_label(ent[2])
            # print(ner)
    nlp1.begin_training()
    for itn in range(n_iter):
        random.shuffle(TRAIN_DATA)
        losses = {}
        batches = minibatch(TRAIN_DATA,
size=compounding(4.0, 32.0, 1.001))
        for batch in batches:
            texts,annotations = zip(*batch)
            nlp1.update(
                texts,
                annotations,
                drop=0.5,
                losses=losses,
            )
        print("Losses",losses)
    return nlp1

nlp3 = train_ner(TRAIN_DATA)

from spacy import displacy



sentence="Paracetamol, also known as acetaminophen, is a
medication used to treat fever and mild to moderate
pain."
print("Sentence is: ", sentence)
sentence_doc = nlp3(sentence)
colors = {"DRUG": "coral", "ILLNESS": "GREEN"}
options = {"ents": ["DRUG", "ILLNESS"], "colors": colors}
```

```
displacy.render(sentence_doc,style='ent',jupyter=True,opt
ions=options)

for sentence in df['info']:
  print("Sentence is: ", sentence)
  sentence_doc = nlp3(sentence)
  colors = {"DRUG": "coral", "ILLNESS": "GREEN"}
  options = {"ents": ["DRUG", "ILLNESS"], "colors":
colors}

displacy.render(sentence_doc,style='ent',jupyter=True,opt
ions=options)

def find_terms(text):
    terms = []
    review = process_review(text)
    for token in review.split():
          if token in all_drugs:
            terms.append(token)
    return terms

# apply function
df['drug_terms'] = df['info'].apply(find_terms)

def find_terms_ill(text):
    terms = []
    review = process_review(text)
    for token in review.split():
          if token in all_illness:
            terms.append(token)
    return terms

# apply function
df['illness_terms'] = df['info'].apply(find_terms_ill)

df.head()

sensitive = pd.DataFrame(columns=['Content','Drug_Term'])
unsensitive = pd.DataFrame(columns=['Content'])
row_list1 = []
row_list2 = []
```

```
for i in range(len(df)):
    if len(df.loc[i,'drug_terms'])!=0:
      for k in df.loc[i,'drug_terms']:

row_list1.append({'Content':df.loc[i,'info'],'Drug_Term':
k})
    else:
      row_list2.append({'Content':df.loc[i,'info']})
sensitive = pd.DataFrame(row_list1)
unsensitive = pd.DataFrame(row_list2)

sensitive.head()

unsensitive.head()

drug_suffix = {"azole":"antifungal (except
metronidazole)",
"caine":"anesthetic",
"cillin":"antibiotic(penicillins)",
"mycin":"antibiotic",
"micin":"antibiotic",
"cycline":"antibiotic",
"oxacin":"antibiotic",
"ceph":"antibiotic(cephalosporins)",
"cef":"antibiotic (cephalosporins)",
"dine":"h2 blockers (anti-ulcers)",
"done":"opiod analgesics",
"ide":"oral hypoglycemics",
"lam":"anti-anxiety",
"pam":"anti-anxiety",
"mide":"diuretics",
"zide":"diuretics",
"nium":"neuromuscular blocking agents",
"olol":"beta blockers",
"tidine":"h2 antagonist",
"tropin":"pituitary hormone",
"zosin":"alpha blocker",
"ase":"thrombolytics",
"plase":"thrombolytics",
"azepam":"anti-anziety(benzodiazepine)",
```

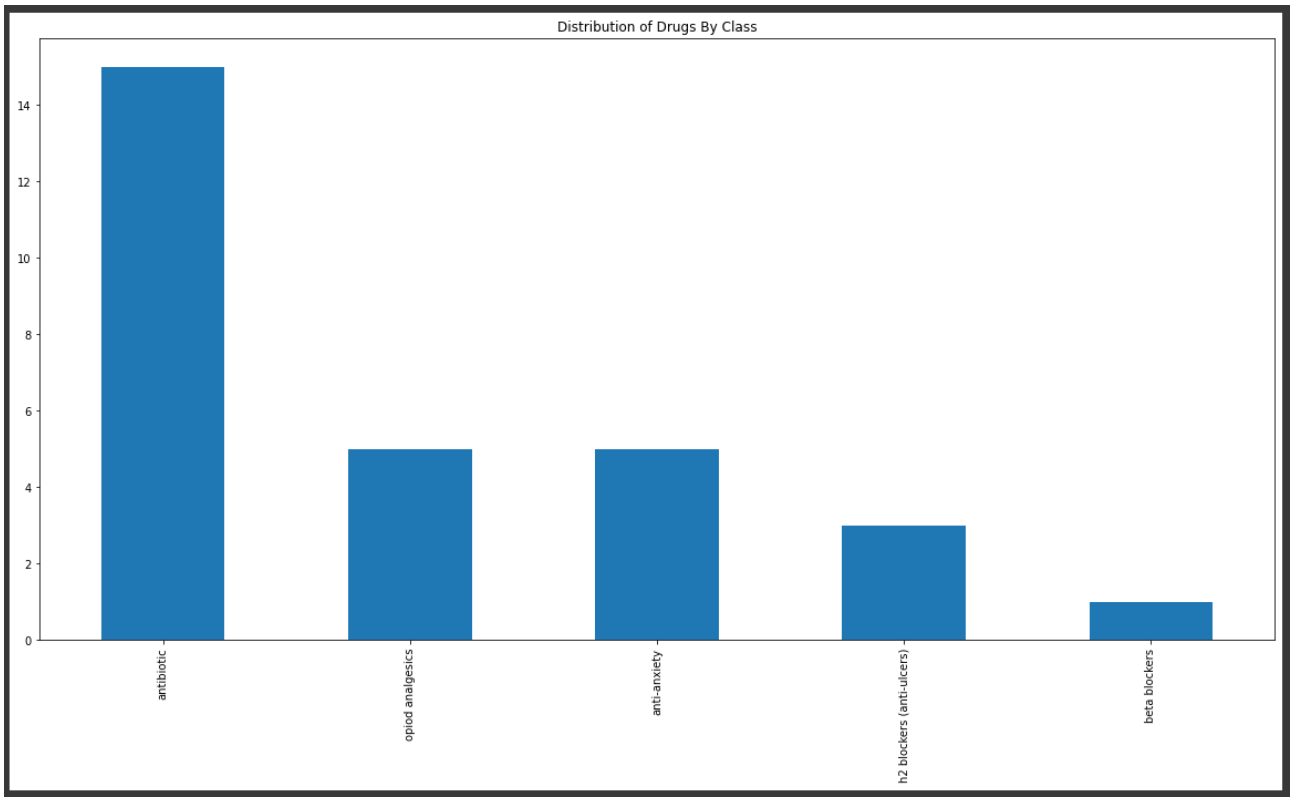
```
"azine":"antipyschotics (phenothiazine)",
"barbital":"barbiturate",
"dipine":"calcium channel blocker",
"lol":"beta blocker",
"zolam":"cns depressants",
"pril":"ace inhibitor",
"artan":"arb blocker",
"statins":"lipid-lowering drugs",
"parin":"anticoagulants",
"sone":"corticosteroid (prednisone)"}

def classify_drug(drugname):
    for i in drug_suffix.keys():
        if drugname.endswith(i):
            return drug_suffix[i]

sensitive['Drug_Class'] =
sensitive['Drug_Term'].apply(classify_drug)

sensitive

sensitive['Drug_Class'].unique().tolist()

sensitive['Drug_Class'].value_counts()

plt.figure(figsize=(20,10))
sensitive['Drug_Class'].value_counts().plot(kind='bar')
plt.title("Distribution of Drugs By Class")
plt.show()
```

Results

- Using default spaCy corpus
  - Successful tagging
  - But drugs and illnesses not tagged explicitly

- Using the drug and illness corpora
  - modified to show only entities tagged '*DRUG*' and '*ILLNESS*'
  - Successfully separated drugs and illnesses from data set

♦ Using the drug class to sort using usage statistics



Code

https://colab.research.google.com/drive/18zX-4MKtloWGu_DCL6NEpn5rXTCJoMHQ?usp=sharing

Conclusion

- Till now, we have managed to implement acronym and text extraction on our current database.
- We have successfully labelled the DRUG and ILLNESS entities till now, i.e. we can now recognize 22,000+ drug names in the data set and 5000+ illnesses.
- Here NER is used to associate the drug with the illness and is extremely useful for research and analysis
- We have also added multiple statistical methods of data visualization to display the distribution of each drug by class for uses like post market analysis.

References

♦ Text Mining: Applications and Theory - Michael W. Berry Jacob Kogan

♦ Text Mining: The state of the art and the challenges - Ah-Hwee Tan

♦ What Is Text Mining? - Marti Hearst

♦ Biomedical Text Mining and Its Applications - Raul Rodriguez-Esteban

https://doi.org/10.1371/journal.pcbi.1000597

♦ A survey of current work in biomedical text mining - Aaron M. Cohen, William R. Hersh

https://doi.org/10.1093/bib/6.1.57

♦ https://www.udemy.com/course/python-for-data-science-and-machine-learning-bootcamp/learn/lecture/5733574#overview

♦ https://colab.research.google.com/github/WomenWhoCode/WWCodeDataScience/blob/master/Intro_to_NLP/2_NLP_DeepDive1.ipynb

♦ https://archive.ics.uci.edu/ml/datasets/Drug+Review+Dataset+%28Drugs.com%2

♦ https://www.machinelearningplus.com/spacy-tutorial-nlp/

♦ https://github.com/Jcharis/data-science-projects/tree/master/exploratory_data_analysis_in_python_drug_reviews_dataset

♦, "Named Entity Recognition from Biomedical Text Using SVM," - Z. Ju, J. Wang and F. Zhu

https://doi.org/10.1109/icbbe.2011.5779984

♦ A Neural Named Entity Recognition and Multi-Type Normalization Tool for Biomedical Text Mining - Donghyeon Kim, Jinhyuk Lee

https://doi.org/10.1109/ACCESS.2019.2920708