Import Libraries and Reading Dataset

```
import pandas as pd
df = pd.read_csv('Summer Internship - Homework Exercise.csv')
df.head()
print(df.dataset.unique())
df_train = df[df['dataset']!="train"]
df_valid = df[df['dataset']=="validation"]
df_test = df[df['dataset']=="test"]
```

```
['train' 'validation' 'test']
```

Creating Dictionaries for each row

```
LABEL = "STORE_NUMBER"
import re
TRAIN_DATA = []
p = {}
start = 0
end = 0
for i,j in zip(df_train['transaction_descriptor'], df_train['store_number']):
  start = 0
  end = 0
  start = i.find(j)
  end = start + len(j) - 1
  #print(start, end)
  inner_list = [(start, end, "STORE_NUMBER")]
  p = {}
  p["entities"] = inner_list
  TRAIN_DATA.append((i, p))
print(TRAIN_DATA)
```

```
[('DEL TACO 833', {'entities': [(9, 11, 'STORE_NUMBER')]}), ('NNT BURLNGTON STORE472605
```

Getting the ner component

```
import spacy
nlp=spacy.load("en_core_web_sm")
```

```
# Getting the ner component
```

Add Labels

```
ner.add_label(LABEL)

# Resume training
optimizer = nlp.resume_training()
move_names = list(ner.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 nlp.pipe_names if pipe not in pipe_exceptions]
```

Creating Mini Batches and Training the model

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

# Begin training by disabling other pipeline components
with nlp.disable_pipes(*other_pipes) :

  sizes = compounding(1.0, 4.0, 1.001)
  # Training for 2000 iterations
  for itn in range(2000):
    # shuffle examples before training
    random.shuffle(TRAIN_DATA)
    # batch up the examples using spaCy's minibatch
    batches = minibatch(TRAIN_DATA, size=sizes)
    # ictionary to store losses
    losses = {}
    for batch in batches:
      texts, annotations = zip(*batch)
      # Calling update() over the iteration
      nlp.update(texts, annotations, sgd=optimizer, drop=0.5, losses=losses)
      print("Losses", losses)
```

```
Losses {'ner': 293.77224181152553}
Losses {'ner': 301.615347068083}
Losses {'ner': 312.8243426487562}
Losses {'ner': 318.0593963072416}
Losses {'ner': 328.2439369887826}
Losses {'ner': 336.7331905693528}
Losses {'ner': 345.114840897989}
Losses {'ner': 354.4643738598344}
```

```
Losses {'ner': 361.327305826616}
Losses {'ner': 372.4860576719758}
Losses {'ner': 380.3183844418046}
Losses {'ner': 390.50029817528696}
Losses {'ner': 397.68648270077676}
Losses {'ner': 410.8467765182969}
Losses {'ner': 412.92080328101866}
Losses {'ner': 417.3674579310521}
Losses {'ner': 431.82952845574465}
Losses {'ner': 436.84654853225794}
Losses {'ner': 445.01786819578257}
Losses {'ner': 452.1689451086625}
Losses {'ner': 5.727180459495377}
Losses {'ner': 13.923619248863051}
Losses {'ner': 25.11869023084242}
Losses {'ner': 30.086688807760456}
Losses {'ner': 36.71801649645445}
Losses {'ner': 52.06817840651152}
Losses {'ner': 63.05675633863089}
Losses {'ner': 72.72354137018797}
Losses {'ner': 82.6164283664572}
Losses {'ner': 95.65782128885863}
Losses {'ner': 104.71428488806364}
Losses {'ner': 116.6298173578131}
Losses {'ner': 123.81092188373682}
Losses {'ner': 134.01298085466502}
Losses {'ner': 137.7683327966976}
Losses {'ner': 151.22495968878744}
Losses {'ner': 159.5703251079855}
Losses {'ner': 168.47455606024837}
Losses {'ner': 176.12643310587978}
Losses {'ner': 186.42544409792995}
Losses {'ner': 191.73653224270916}
Losses {'ner': 197.80118611615276}
Losses {'ner': 203.96330294294265}
Losses {'ner': 213.36999091787246}
Losses {'ner': 221.67612394971755}
Losses {'ner': 234.60632059259322}
Losses {'ner': 243.86333731098082}
Losses {'ner': 252.18290368003753}
Losses {'ner': 261.52293041629696}
Losses {'ner': 271.09943428916836}
Losses {'ner': 280.23334500474834}
Losses {'ner': 287.8172868816366}
Losses {'ner': 300.6701351730337}
Losses {'ner': 313.7105896799078}
Losses {'ner': 328.9973912565222}
Losses {'ner': 338.42086711807156}
Losses {'ner': 348.33543703241253}
Losses {'ner': 361.9370386330595}
```

```
#df_valid
```

```
# for i in range(len(df_valid[["transaction_descriptor"]])):
```

```
#   print(df_valid["transaction_descriptor"].iloc[i])
```

RESULTS

```
result = []
test_text = "DOLRTREE 2257 00022574 ROSWELL"
#test_text = df_valid.iloc[5]["transaction_descriptor"]
for i in range(len(df_test[["transaction_descriptor"]])):
  doc = nlp(df_test["transaction_descriptor"].iloc[i])
  print("Entities in '%s'" %df_test["transaction_descriptor"].iloc[i])
  result.append(doc.ents)
    #result.append(ent)
    #print(ent)
#result
```

```
Entities in 'NST BEST BUY #48   072393'
Entities in 'NST BEST BUY #231  160037'
Entities in 'WALGREENS #11332'
Entities in 'NST ROSS STORES #16482149'
Entities in 'MCDONALD'S F33735'
Entities in 'MCDONALD'S F671'
Entities in 'MCDONALD'S F11370 0000000'
Entities in 'WM SUPERCENTER #50'
Entities in 'BURGER KING #11820 Q07'
Entities in 'DENNY'S #6619 ON'
Entities in 'DOMINO'S 6102'
Entities in 'MCDONALD'S F2383 972-231-3337 TX'
Entities in 'TACO BELL #733780'
Entities in 'NST ROSS STORES #62132001'
Entities in 'MCDONALD'S F33124'
Entities in 'WALGREENS #15392'
Entities in 'NNT HIBBETT SPORTS 860977'
Entities in 'MCDONALD'S F122'
Entities in 'PAPA JOHN'S #0982'
Entities in 'NST BEST BUY #188  871025'
Entities in 'TACO BELL 15843'
Entities in 'CIRCLE K #2742643'
Entities in 'NNT FAMOUS FOOTWEAR001261'
Entities in 'EXPRESS#0813'
Entities in 'BURGER KING #7414'
Entities in 'SUNOCO 039962380'
Entities in 'NST BEST BUY #392  080590'
Entities in 'ARCO #66165'
Entities in 'WAL-MART #0647'
Entities in 'H&R BLOCK #14788'
Entities in 'FOOTACTION 57331 TAMPA, FL (2340)'
Entities in '7-ELEVEN 34493'
Entities in 'HOLIDAY STNSTORE 408'
Entities in 'STARBUCKS #10101'
Entities in 'SUNOCO 0837208800        STATEN ISLANDNY'
```

```
Entities in 'NST BEST BUY #405  562536'
Entities in 'PANERA BREAD #601128'
Entities in 'BURGER KING #4633  Q07'
Entities in 'WAL-MART #1997'
Entities in 'NST ROSS STORES #21000236'
Entities in 'NNT FAMOUS FOOTWEAR730376'
Entities in 'MARATHON PETRO170928   MIAMI'
Entities in 'SUNOCO 0104235700  QPS'
Entities in 'NNT FAMOUSFOOTWEAR#132427'
Entities in 'STARBUCKS STORE 11966'
Entities in 'SUBWAY        03317963'
Entities in 'NST BEST BUY #401  000948'
Entities in 'NST BEST BUY #51   672842'
Entities in 'PAPA MURPHY'S UT044 OLO'
Entities in 'BP#1003300DANADA SQUS'
Entities in 'SUBWAY        00032128'
Entities in 'TEXACO 00303733'
Entities in 'THE BUCKLE #513'
Entities in 'MCDONALD'S F2151'
Entities in 'NST BEST BUY #1403 332411'
Entities in 'CVS/PHARMACY #06689'
Entities in 'BANANA REPUBLIC #8109'
Entities in 'BOSTON MARKET 0443'
```

```
result_1 = []
for i in range(len(result)):
  try:
    result_1.append(result[i][0])
  except:
    result_1.append(' ')
len(result_1)
```

```
100
```

```
df_test_1 = df_test.copy()
```

```
df_test_1["pred"] = result_1
```

```
df_test_1
```

1 to 25 of 100 entries Filter

index	transaction_descriptor	store_number	dataset	pred
200	IN-N-OUT BURGER #242	242	test	242
201	BP#9442088LIBERTYVILLE B	9442088	test	
202	JCPENNEY 1419	1419	test	1419
203	ROSS STORES #1019	1019	test	1019
204	WM SUPERCENTER #38	38	test	38
205	TUESDAY MORNING # 0673 06	673	test	
206	IHOP 629 WHITE HOUSE TN	629	test	629
207	LBOUTLETS#4249 1475 N BUR	4249	test	
208	WINN DIXIE #2505 VALRICO, FL (3454)	2505	test	
209	BURLINGTON STORES 825	825	test	825
210	WM SUPERCENTER #2923	2923	test	2923
211	BUFFALO WILD WINGS 058 CARSON CITY NV	58	test	058
212	BOB EVANS REST #2039	2039	test	2039
213	JIMMY JOHNS # 382 - E	382	test	382
214	PENSKE TRK LSG 012260	12260	test	012260
215	AEROPOSTALE # 864	864	test	864
216	GIANT 0338	338	test	0338

```
def accuracy(actual , predicted):
    count = 0
    for i, j in zip(actual, predicted):
        if str(i)==str(j):
            count+=1
        else:
            pass
    return (count/len(actual))*100
```

Show 25 per page

```
acc = accuracy(df_test_1['store_number'], df_test_1['pred'])
print(f"The accuracy of the Entity Recognition Model is {acc}%")
```

```
The accuracy of the Entity Recognition Model is 68.0%
```

✓ 0s completed at 03:06

Entity Extraction Model

Goal: Build an entity extractor model to extract the store_number from the transaction_descriptor.

1. The dataset given has three columns: transaction_descriptor, store_number, and dataset.
2. To build an entity extraction model with the given dataset, I used an inbuilt python library called spaCy that allows us to train named entity recognition models.
3. spaCy has in-built pipeline “ner” for Named recognition. Though it performs well, it’s not always completely accurate for our text.
4. To overcome this problem, I used an existing pre-trained spacy model and update it with newer examples.
5. To do this, we need example texts and the character offsets and labels of each entity contained in the texts.
6. Ex: ('DEL TACO 833', {'entities': [(9, 11, 'STORE_NUMBER')]}) where 9 is the starting index of the store_number and 11 is the ending index.
7. Then I have added these labels to the ner model using ner.add_label() method of pipeline. Now the model is ready to be trained.
8. But before I trained the model, I disabled the other components of the library as they should not be affected by the training.
9. After training, the model does not memorize the training examples as it should learn from them and generalize it to new examples.
10. Once I found the performance of the model satisfactory, I saved the updated model.