# main

April 9, 2022

### 0.0.1 Importing Required Libraries

[ ]:
```
import spacy
import pandas as pd
import re
import pickle
from spacy.tokens import DocBin
import json
from tqdm import tqdm
import warnings
warnings.filterwarnings("ignore")
```

### 0.0.2 Reading Data from csv file

[ ]:
```
'''
importing data
'''
data = pd.read_csv('Summer Internship - Homework Exercise.csv')
data.head()
```

[ ]:

| | transaction_descriptor | store_number | dataset |
|---|---|---|---|
| 0 | DOLRTREE 2257 00022574 ROSWELL | 2257 | train |
| 1 | AUTOZONE #3547 | 3547 | train |
| 2 | TGI FRIDAYS 1485 0000 | 1485 | train |
| 3 | BUFFALO WILD WINGS 003 | 3 | train |
| 4 | J. CREW #568 0 | 568 | train |

**Cleaning Data**

[ ]:
```
def clean_data(x):
    x = re.sub('\W+',' ',x).split()
    x = ' '.join(x)
    return x
data.transaction_descriptor = data.transaction_descriptor.apply(clean_data)
```

**Split Data to train, test and validation folds**

[ ]:
```
train_data = data[data.dataset=='train']
val_data = data[data.dataset=='validation']
```

```
test_data = data[data.dataset=='test']
```

**Create custom entities to finetune spacy model learn new entities**

[ ]:
```
def create_train_valid_data(data):
    transaction_descriptor = data.transaction_descriptor.tolist()
    store_number_list = data.store_number.tolist()
    data =  [(descriptor, {'entities':[(re.search(store_number,descriptor).
 ↪start(),re.search(store_number,descriptor).end(),'store_detector') ] }) for␣
 ↪descriptor,store_number in zip(transaction_descriptor,store_number_list)]
    return data
```

**Dump newly created train, test, valid data to json files**

[ ]:
```
with open('train.json','w',encoding='utf-8') as f:
    json.dump(create_train_valid_data(train_data),f)
with open('valid.json','w',encoding='utf-8') as f:
    json.dump(create_train_valid_data(val_data),f)
with open('test.json','w') as f:
    json.dump(create_train_valid_data(test_data),f)
```

**Create a new blank spacy model and train, validation data in spacy format**

[ ]:
```
#nlp = spacy.blank('en') ## create blank spacy model
# nlp = spacy.load('en_core_web_trf')
nlp = spacy.load('en_core_web_lg')


def create_training(TRAIN_DATA):
    db = DocBin()
    for text, annotations in tqdm(TRAIN_DATA):
        doc = nlp.make_doc(text)
        ents=[]
        for start, end, label in annotations['entities']:
            span = doc.char_span(start, end,␣
 ↪label=label,alignment_mode='expand')
            if span is None:
                print("skipping entity {} in {}".format(label, text))
            else:
                ents.append(span)
        doc.ents = ents
        db.add(doc)
    return (db)
```

**convert train, validation data from json to spacy format**

[ ]:
```
with open('train.json','r') as f:
    train_data = json.load(f)
with open('valid.json','r') as f:
```

```
    valid_data = json.load(f)

train_data = create_training(train_data)
train_data.to_disk('./data/train_lg.spacy')
valid_data = create_training(valid_data)
valid_data.to_disk('./data/valid_lg.spacy')
```

```
100%|      | 100/100 [00:00<00:00, 2563.80it/s]
100%|      | 100/100 [00:00<00:00, 2631.18it/s]
```

### 0.0.3 Load the spacy model and predict on test data

1. For multiple predictions on a test sample, only first prediction is taken into consideration

[ ]:
```
nlp = spacy.load('large_out/model-best')
docs = test_data.transaction_descriptor.tolist()
store_numbers=[]
for doc in docs:
    doc= nlp(doc)
    temp =[]
    for ent in doc.ents:
        if ent.label_=='store_detector':
            temp.append(ent.text.lstrip('0'))
            break
    else:
        temp.append('Not_Predicted')
    store_numbers.extend(temp)
```

**Write test predictions to a csv file**

[ ]:
```
test_data['store_number_pred'] = store_numbers
test_data.to_csv('test_data_predictions.csv')
```

### 0.0.4 Calculating Accuracy score for Entity Extraction model on test data

[ ]:
```
from sklearn.metrics import accuracy_score
accuracy_score(test_data.store_number,test_data.store_number_pred)
```

[ ]: 0.9

**Steps Followed to extract entities**

- Data

[ ]: