Named Entity Recognition

```
# Loading Dataset

import pandas as pd
data = pd.read_csv('/content/ner-dataset.txt', encoding= 'unicode_escape')
data.head()
```

	Sentence #	Word	POS	Tag
0	Sentence: 1	Thousands	NNS	O
1	NaN	of	IN	O
2	NaN	demonstrators	NNS	O
3	NaN	have	VBP	O
4	NaN	marched	VBN	O

Data Preparation for Neural Networks

```
from itertools import chain
def get_dict_map(data, token_or_tag):
    tok2idx = {}
    idx2tok = {}

    if token_or_tag == 'token':
        vocab = list(set(data['Word'].to_list()))
    else:
        vocab = list(set(data['Tag'].to_list()))

    idx2tok = {idx:tok for  idx, tok in enumerate(vocab)}
    tok2idx = {tok:idx for  idx, tok in enumerate(vocab)}
    return tok2idx, idx2tok
token2idx, idx2token = get_dict_map(data, 'token')
tag2idx, idx2tag = get_dict_map(data, 'tag')


data['Word_idx'] = data['Word'].map(token2idx)
data['Tag_idx'] = data['Tag'].map(tag2idx)
data_fillna = data.fillna(method='ffill', axis=0)
# Groupby and collect columns
data_group = data_fillna.groupby(
['Sentence #'],as_index=False
)['Word', 'POS', 'Tag', 'Word_idx', 'Tag_idx'].agg(lambda x: list(x))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: FutureWarning: Indexing
  import sys
```

Split the data into training and test sets

```
from sklearn.model_selection import train_test_split
from keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
import spacy
from spacy import displacy

def get_pad_train_test_val(data_group, data):

    #get max token and tag length
    n_token = len(list(set(data['Word'].to_list())))
    n_tag = len(list(set(data['Tag'].to_list())))

    #Pad tokens (X var)
    tokens = data_group['Word_idx'].tolist()
    maxlen = max([len(s) for s in tokens])
    pad_tokens = pad_sequences(tokens, maxlen=maxlen, dtype='int32', padding='post', value= r

    #Pad Tags (y var) and convert it into one hot encoding
    tags = data_group['Tag_idx'].tolist()
    pad_tags = pad_sequences(tags, maxlen=maxlen, dtype='int32', padding='post', value= tag2i
    n_tags = len(tag2idx)
    pad_tags = [to_categorical(i, num_classes=n_tags) for i in pad_tags]

    #Split train, test and validation set
    tokens_, test_tokens, tags_, test_tags = train_test_split(pad_tokens, pad_tags, test_size
    train_tokens, val_tokens, train_tags, val_tags = train_test_split(tokens_,tags_,test_size

    print(
        'train_tokens length:', len(train_tokens),
        '\ntrain_tokens length:', len(train_tokens),
        '\ntest_tokens length:', len(test_tokens),
        '\ntest_tags:', len(test_tags),
        '\nval_tokens:', len(val_tokens),
        '\nval_tags:', len(val_tags),
    )

    return train_tokens, val_tokens, test_tokens, train_tags, val_tags, test_tags

train_tokens, val_tokens, test_tokens, train_tags, val_tags, test_tags = get_pad_train_test_v
```

```
train_tokens length: 24408
train_tokens length: 24408
test_tokens length: 3617
```

```
test_tags: 3617
val_tokens: 8137
val_tags: 8137
```

Training Neural Network for Named Entity Recognition (NER)

```
import numpy as np
import tensorflow
from tensorflow.keras import Sequential, Model, Input
from tensorflow.keras.layers import LSTM, Embedding, Dense, TimeDistributed, Dropout, Bidirec
from tensorflow.keras.utils import plot_model
from numpy.random import seed
seed(1)
tensorflow.random.set_seed(2)


input_dim = len(list(set(data['Word'].to_list())))+1
output_dim = 64
input_length = max([len(s) for s in data_group['Word_idx'].tolist()])
n_tags = len(tag2idx)


def get_bilstm_lstm_model():
    model = Sequential()

    # Add Embedding layer
    model.add(Embedding(input_dim=input_dim, output_dim=output_dim, input_length=input_length

    # Add bidirectional LSTM
    model.add(Bidirectional(LSTM(units=output_dim, return_sequences=True, dropout=0.2, recurr

    # Add LSTM
    model.add(LSTM(units=output_dim, return_sequences=True, dropout=0.5, recurrent_dropout=0.

    # Add timeDistributed Layer
    model.add(TimeDistributed(Dense(n_tags, activation="relu")))

    #Optimiser
    # adam = k.optimizers.Adam(lr=0.0005, beta_1=0.9, beta_2=0.999)

    # Compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    model.summary()

    return model


def train_model(X, y, model):
    loss = list()
    for i in range(25):
        # fit model for one epoch on this sequence
        hist = model.fit(X, y, batch_size=1000, verbose=1, epochs=1, validation_split=0.2)
        loss.append(hist.history['loss'][0])
    return loss


results = pd.DataFrame()
model_bilstm_lstm = get_bilstm_lstm_model()
plot_model(model_bilstm_lstm)
results['with_add_lstm'] = train_model(train_tokens, np.array(train_tags), model_bilstm_lstm)
```

```
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the crit
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the crit
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the crit
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet the cr
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding (Embedding)       (None, 104, 64)           1979328   
                                                                 
 bidirectional (Bidirectiona  (None, 104, 128)         66048     
 l)                                                              
                                                                 
 lstm_1 (LSTM)               (None, 104, 64)           49408     
                                                                 
 time_distributed (TimeDistr  (None, 104, 17)          1105      
 ibuted)                                                         
                                                                 
=================================================================
Total params: 2,095,889
Trainable params: 2,095,889
Non-trainable params: 0
_________________________________________________________________
20/20 [==============================] - 41s 1s/step - loss: nan - accuracy: 0.9196 - v
20/20 [==============================] - 27s 1s/step - loss: 0.4259 - accuracy: 0.9679
20/20 [==============================] - 26s 1s/step - loss: 0.3575 - accuracy: 0.9680
20/20 [==============================] - 27s 1s/step - loss: 0.2838 - accuracy: 0.9680
20/20 [==============================] - 26s 1s/step - loss: 0.2374 - accuracy: 0.9680
20/20 [==============================] - 26s 1s/step - loss: 0.2188 - accuracy: 0.9680
20/20 [==============================] - 26s 1s/step - loss: 0.1994 - accuracy: 0.9681
20/20 [==============================] - 27s 1s/step - loss: 0.2061 - accuracy: 0.9680
20/20 [==============================] - 26s 1s/step - loss: 0.1928 - accuracy: 0.9680
20/20 [==============================] - 27s 1s/step - loss: 0.2273 - accuracy: 0.9680
20/20 [==============================] - 26s 1s/step - loss: 0.2090 - accuracy: 0.9680
20/20 [==============================] - 26s 1s/step - loss: 0.1493 - accuracy: 0.9682
20/20 [==============================] - 27s 1s/step - loss: 0.1426 - accuracy: 0.9681
20/20 [==============================] - 26s 1s/step - loss: 0.1310 - accuracy: 0.9682
20/20 [==============================] - 26s 1s/step - loss: 0.1299 - accuracy: 0.9682
20/20 [==============================] - 27s 1s/step - loss: 0.1228 - accuracy: 0.9681
20/20 [==============================] - 27s 1s/step - loss: 0.1152 - accuracy: 0.9682
20/20 [==============================] - 26s 1s/step - loss: 0.1120 - accuracy: 0.9683
20/20 [==============================] - 26s 1s/step - loss: 0.1086 - accuracy: 0.9684
20/20 [==============================] - 26s 1s/step - loss: 0.1090 - accuracy: 0.9684
```

```
20/20 [==============================] - 26s 1s/step - loss: 0.1077 - accuracy: 0.9684
20/20 [==============================] - 27s 1s/step - loss: 0.1030 - accuracy: 0.9685
20/20 [==============================] - 27s 1s/step - loss: 0.1022 - accuracy: 0.9686
20/20 [==============================] - 27s 1s/step - loss: 0.0991 - accuracy: 0.9687
20/20 [==============================] - 26s 1s/step - loss: 0.0963 - accuracy: 0.9688
```

```
nlp = spacy.load('en_core_web_sm')
text = nlp('Suraj Gupta is working on Google Inc')
displacy.render(text, style = 'ent', jupyter=True)
```

Suraj Gupta **PERSON** is working on Google Inc **ORG**

```
import pickle
pickle.dump(model_bilstm_lstm, open('ner-model.pkl','wb'))
```

```
ng is not possible, pass the object in the `custom_objects` parameter of the load functi
ng is not possible, pass the object in the `custom_objects` parameter of the load functi
ng is not possible, pass the object in the `custom_objects` parameter of the load functi
```

```
print(model_bilstm_lstm)
```

```
<keras.engine.sequential.Sequential object at 0x7f6d56464850>
```

✓ 0s completed at 7:59 AM ● ×