NATURAL LANGUAGE PROCESSING

DIGITAL ASSIGNMENT - 2

Member 1:
Name: Mohamed Riyaas R
Reg.: 21BCE5828

Member 2:
Name: Om Prakash
Reg.: 21BCE1950

Slot: B1+TB1
Course Code: BCSE409L
Faculty: Manjula

TITLE:

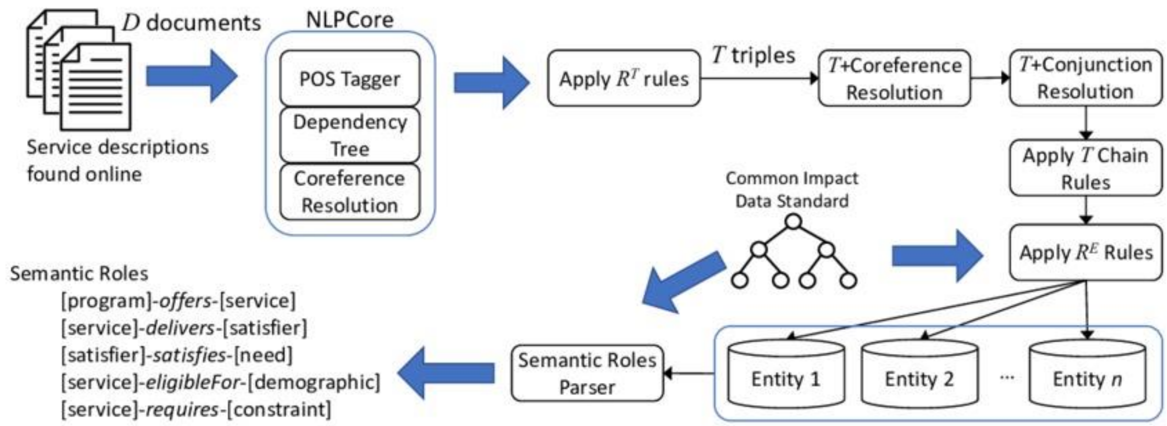
NAMED ENTITY RECOGNITION (NER) using Bert model:

Named Entity Recognition (NER) is a natural language processing (NLP) task that involves identifying and classifying named entities in text into predefined categories such as person names, organizations, locations, dates, and more. It's a crucial step in various NLP applications like information extraction, question answering, and sentiment analysis.

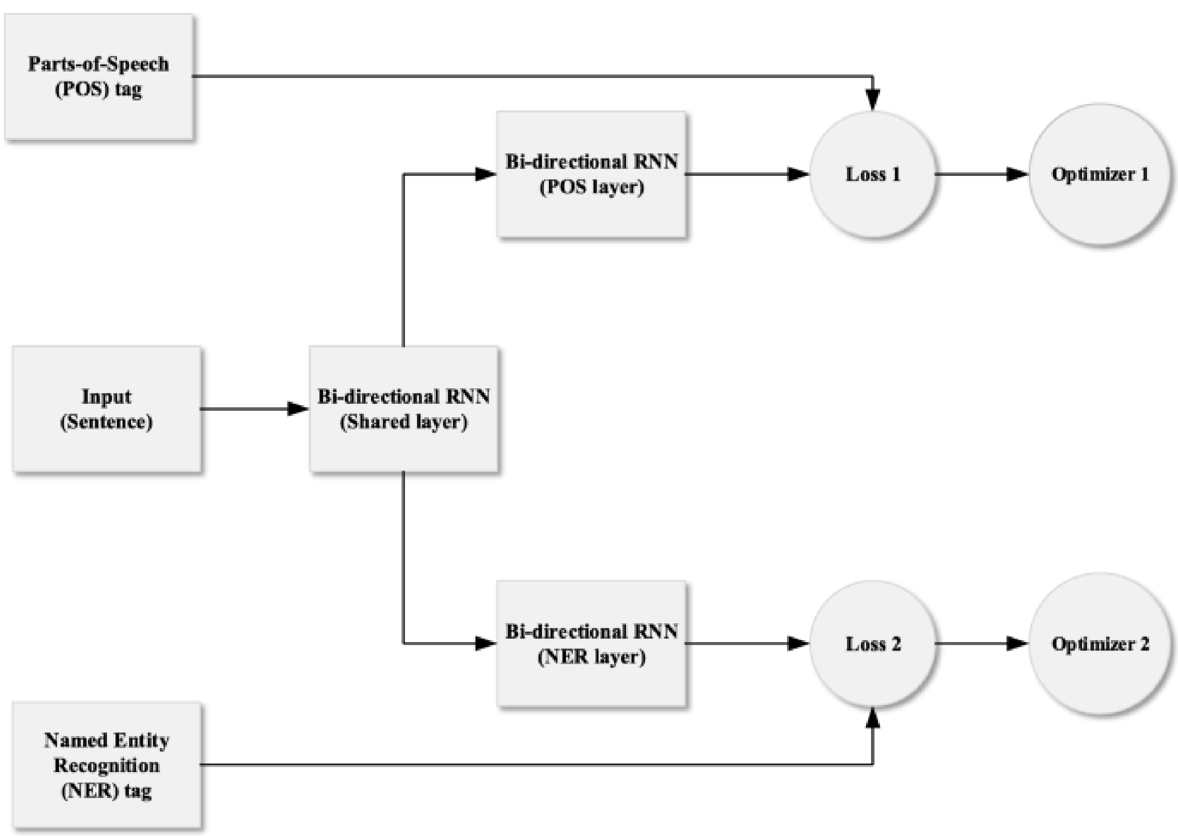
BERT (Bidirectional Encoder Representations from Transformers) is a powerful pre-trained language model developed by Google that has revolutionized NLP tasks. It utilizes a transformer architecture, allowing it to capture bidirectional contextual information from input text, which is particularly beneficial for tasks like NER.

In NER with BERT, the model is fine-tuned on labeled NER datasets, where it learns to predict the entity type for each token in the input text. By leveraging BERT's contextual embeddings and fine-tuning on task-specific data, NER models achieve state-of-the-art performance, surpassing traditional methods.

DESIGN:



ARCHITECTURE:



CODE:

```
!pip install simpletransformers

import pandas as pd
data = pd.read_csv("ner_dataset.csv",encoding="latin1" )

data.head(30)

data =data.fillna(method ="ffill")

data.head(30)


from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

data["Sentence #"] = LabelEncoder().fit_transform(data["Sentence #"] )

data.head(30)


data.rename(columns={"Sentence #":"sentence_id","Word":"words","Tag":"labels"},
inplace =True)


data["labels"] = data["labels"].str.upper()


X= data[["sentence_id","words"]]
Y =data["labels"]

x_train, x_test, y_train, y_test = train_test_split(X,Y, test_size =0.2)


#building up train data and test data
train_data =
pd.DataFrame({"sentence_id":x_train["sentence_id"],"words":x_train["words"],"labels":y_
train})
```

```
test_data =
pd.DataFrame({"sentence_id":x_test["sentence_id"],"words":x_test["words"],"labels":y_te
st})
```

```
train_data
```

```
from simpletransformers.ner import NERModel,NERArgs
```

wandb: WARNING W&B installed but not logged in. Run `wandb login` or set the WANDB_API_KEY env variable.

```
label = data["labels"].unique().tolist()
label
```

```
args = NERArgs()
args.num_train_epochs = 1
args.learning_rate = 1e-4
args.overwrite_output_dir =True
args.train_batch_size = 32
args.eval_batch_size = 32
```

```
model = NERModel('bert', 'bert-base-cased',labels=label,args =args)
```

```
model.train_model(train_data,eval_data = test_data,acc=accuracy_score)
```

```
result, model_outputs, preds_list = model.eval_model(test_data)
```

```
prediction, model_output = model.predict(["What is the new name of Bangalore"])
```

```
prediction
```

IMPLEMENTATION:

NAMED ENTITY RECOGNITION:

1. The named entities are pre-defined categories chosen according to the use case such as names of people, organizations, places, codes, time notations, monetary values, etc.

2. NER aims to assign a class to each token (usually a single word) in a sequence. Because of this, NER is also referred to as token classification.

In [1]:

```
!pip install simpletransformers
```

```
Collecting simpletransformers
  Downloading https://files.pythonhosted.org/packages/56/35/31022262786f4aa070fe472677cea66fade8d221181a86825
096af021e2c/simpletransformers-0.48.14-py3-none-any.whl (214kB)
     |████████████████████████████████| 215kB 6.5MB/s
Collecting tokenizers
  Downloading https://files.pythonhosted.org/packages/3d/54/ffb2a4d26762f967aff57562b8e6586a2a8e20f6c26aee479
11627ca7786/tokenizers-0.9.1-cp36-cp36m-manylinux1_x86_64.whl (2.9MB)
     |████████████████████████████████| 2.9MB 12.1MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from simpletransformers) (1.1
8.5)
Requirement already satisfied: regex in /usr/local/lib/python3.6/dist-packages (from simpletransformers) (201
9.12.20)
Collecting seqeval
```

```
266be190007/subprocess32-3.5.4.tar.gz (97kB)
     |████████████████████████████████| 102kB 12.0MB/s
Collecting GitPython>=1.0.0
  Downloading https://files.pythonhosted.org/packages/c0/d7/b2b0672e0331567157adf9281f41ee731c412ee518ca5e655
2c27fa73c91/GitPython-3.1.9-py3-none-any.whl (159kB)
     |████████████████████████████████| 163kB 41.5MB/s
Requirement already satisfied: Click>=7.0 in /usr/local/lib/python3.6/dist-packages (from wandb->simpletransf
ormers) (7.1.2)
Collecting shortuuid>=0.5.0
  Downloading https://files.pythonhosted.org/packages/25/a6/2ecc1daa6a304e7f1b216f0896b26156b78e7c38e1211e9b7
98b4716c53d/shortuuid-1.0.1-py3-none-any.whl
Requirement already satisfied: psutil>=5.0.0 in /usr/local/lib/python3.6/dist-packages (from wandb->simpletra
nsformers) (5.4.8)
Collecting configparser>=3.8.1
  Downloading https://files.pythonhosted.org/packages/08/b2/ef713e0e67f6e7ec7d59aea3ee78d05b39c15930057e724cc
6d362a8c3bb/configparser-5.0.1-py3-none-any.whl
Requirement already satisfied: PyYAML in /usr/local/lib/python3.6/dist-packages (from wandb->simpletransforme
rs) (3.13)
Requirement already satisfied: promise<3,>=2.0 in /usr/local/lib/python3.6/dist-packages (from wandb->simplet
ransformers) (2.3)
Collecting docker-pycreds>=0.4.0
  Downloading https://files.pythonhosted.org/packages/f5/e8/f6bd1eee09314e7e6dee49cbe2c5e22314ccdb38db16c9fc7
2d2fa80d054/docker_pycreds-0.4.0-py2.py3-none-any.whl
Collecting watchdog>=0.8.3
  Downloading https://files.pythonhosted.org/packages/0e/06/121302598a4fc01aca942d937f4a2c33430b7181137b35758
913a8db10ad/watchdog-0.10.3.tar.gz (94kB)
     |████████████████████████████████| 102kB 10.4MB/s
```

```
Installing collected packages: tokenizers, seqeval, tensorboardx, subprocess32, smmap, gitdb, GitPython, shor
tuuid, configparser, docker-pycreds, pathtools, watchdog, sentry-sdk, wandb, jmespath, botocore, s3transfer,
boto3, base58, validators, ipykernel, pydeck, blinker, enum-compat, streamlit, sentencepiece, tqdm, sacremose
s, transformers, simpletransformers
  Found existing installation: ipykernel 4.10.1
    Uninstalling ipykernel-4.10.1:
      Successfully uninstalled ipykernel-4.10.1
  Found existing installation: tqdm 4.41.1
    Uninstalling tqdm-4.41.1:
      Successfully uninstalled tqdm-4.41.1
Successfully installed GitPython-3.1.9 base58-2.0.1 blinker-1.4 boto3-1.15.16 botocore-1.18.16 configparser-
5.0.1 docker-pycreds-0.4.0 enum-compat-0.0.3 gitdb-4.0.5 ipykernel-5.3.4 jmespath-0.10.0 pathtools-0.1.2 pyde
ck-0.5.0b1 s3transfer-0.3.3 sacremoses-0.0.43 sentencepiece-0.1.91 sentry-sdk-0.19.0 seqeval-1.1.1 shortuuid-
1.0.1 simpletransformers-0.48.14 smmap-3.0.4 streamlit-0.68.1 subprocess32-3.5.4 tensorboardx-2.1 tokenizers-
0.9.1 tqdm-4.50.2 transformers-3.3.1 validators-0.18.1 wandb-0.10.5 watchdog-0.10.3
```

In [2]:
```
import pandas as pd
data = pd.read_csv("ner_dataset.csv",encoding="latin1" )
```

In [3]:
```
data.head(30)
```

Out[3]:

	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
5	NaN	through	IN	O
6	NaN	London	NNP	B-geo
7	NaN	to	TO	O
8	NaN	protest	VB	O
9	NaN	the	DT	O
10	NaN	war	NN	O
11	NaN	in	IN	O
12	NaN	Iraq	NNP	B-geo
13	NaN	and	CC	O
14	NaN	demand	VB	O
15	NaN	the	DT	O
16	NaN	withdrawal	NN	O
17	NaN	of	IN	O
18	NaN	British	JJ	B-gpe
19	NaN	troops	NNS	O
20	NaN	from	IN	O
21	NaN	that	DT	O
22	NaN	country	NN	O
23	NaN	.	.	O
24	Sentence: 2	Families	NNS	O
25	NaN	of	IN	O
26	NaN	soldiers	NNS	O
27	NaN	killed	VBN	O
28	NaN	in	IN	O
29	NaN	the	DT	O

In [4]:

```
data =data.fillna(method ="ffill")
```

In [5]:

```
data.head(30)
```

Out[5]:

	Sentence #	Word	POS	Tag
0	Sentence: 1	Thousands	NNS	O
1	Sentence: 1	of	IN	O
2	Sentence: 1	demonstrators	NNS	O
3	Sentence: 1	have	VBP	O
4	Sentence: 1	marched	VBN	O
5	Sentence: 1	through	IN	O
6	Sentence: 1	London	NNP	B-geo
7	Sentence: 1	to	TO	O
8	Sentence: 1	protest	VB	O
9	Sentence: 1	the	DT	O
10	Sentence: 1	war	NN	O
11	Sentence: 1	in	IN	O
12	Sentence: 1	Iraq	NNP	B-geo
13	Sentence: 1	and	CC	O
14	Sentence: 1	demand	VB	O
15	Sentence: 1	the	DT	O
16	Sentence: 1	withdrawal	NN	O
17	Sentence: 1	of	IN	O
18	Sentence: 1	British	JJ	B-gpe
19	Sentence: 1	troops	NNS	O
20	Sentence: 1	from	IN	O
21	Sentence: 1	that	DT	O
22	Sentence: 1	country	NN	O
23	Sentence: 1	.	.	O
24	Sentence: 2	Families	NNS	O
25	Sentence: 2	of	IN	O
26	Sentence: 2	soldiers	NNS	O
27	Sentence: 2	killed	VBN	O
28	Sentence: 2	in	IN	O
29	Sentence: 2	the	DT	O

In [6]:
```python
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [7]:
```python
data["Sentence #"] = LabelEncoder().fit_transform(data["Sentence #"] )
```

In [8]:
```python
data.head(30)
```

Out[8]:

	Sentence #	Word	POS	Tag
0	0	Thousands	NNS	O
1	0	of	IN	O
2	0	demonstrators	NNS	O
3	0	have	VBP	O
4	0	marched	VBN	O
5	0	through	IN	O
6	0	London	NNP	B-geo
7	0	to	TO	O
8	0	protest	VB	O
9	0	the	DT	O
10	0	war	NN	O
11	0	in	IN	O
12	0	Iraq	NNP	B-geo
13	0	and	CC	O
14	0	demand	VB	O
15	0	the	DT	O
16	0	withdrawal	NN	O
17	0	of	IN	O
18	0	British	JJ	B-gpe
19	0	troops	NNS	O
20	0	from	IN	O
21	0	that	DT	O
22	0	country	NN	O
23	0	.	.	O
24	11111	Families	NNS	O
25	11111	of	IN	O
26	11111	soldiers	NNS	O
27	11111	killed	VBN	O
28	11111	in	IN	O
29	11111	the	DT	O

In [9]:
```python
data.rename(columns={"Sentence #":"sentence_id","Word":"words","Tag":"labels"}, inplace =True)
```

In [10]:
```python
data["labels"] = data["labels"].str.upper()
```

In [11]:
```python
X= data[["sentence_id","words"]]
Y =data["labels"]
```

In [12]:
```python
x_train, x_test, y_train, y_test = train_test_split(X,Y, test_size =0.2)
```

In [13]:
```python
#building up train data and test data
train_data = pd.DataFrame({"sentence_id":x_train["sentence_id"],"words":x_train["words"],"labels":y_train})
test_data = pd.DataFrame({"sentence_id":x_test["sentence_id"],"words":x_test["words"],"labels":y_test})
```

In [14]:
```python
train_data
```

Out[14]:

	sentence_id	words	labels
86887	32767	of	O
753253	27154	video	O
468990	12711	debris	O
644471	21618	of	O
390705	8742	holding	O
...	...	...	...
315581	4944	came	O
1014442	40436	recent	O
78774	28656	lead	O
796271	29332	his	O
483156	13451	be	O

838860 rows × 3 columns

Model Training

In [15]:
```python
from simpletransformers.ner import NERModel,NERArgs
```

```
wandb: WARNING W&B installed but not logged in.  Run `wandb login` or set the WANDB_API_KEY env variable.
```

In [16]:
```python
label = data["labels"].unique().tolist()
label
```

Out[16]:
```
['O',
 'B-GEO',
 'B-GPE',
 'B-PER',
 'I-GEO',
 'B-ORG',
 'I-ORG',
 'B-TIM',
 'B-ART',
 'I-ART',
 'I-PER',
 'I-GPE',
 'I-TIM',
 'B-NAT',
 'B-EVE',
 'I-EVE',
 'I-NAT']
```

In [17]:
```
args = NERArgs()
args.num_train_epochs = 1
args.learning_rate = 1e-4
args.overwrite_output_dir =True
args.train_batch_size = 32
args.eval_batch_size = 32
```

In [18]:
```
model = NERModel('bert', 'bert-base-cased',labels=label,args =args)
```

```
HBox(children=(HTML(value='Downloading'), FloatProgress(value=0.0, max=433.0), HTML(value='')))
HBox(children=(HTML(value='Downloading'), FloatProgress(value=0.0, max=435779157.0), HTML(value='')))
Some weights of the model checkpoint at bert-base-cased were not used when initializing BertForTokenClassification: ['cls.pre
dictions.bias', 'cls.predictions.transform.dense.weight', 'cls.predictions.transform.dense.bias', 'cls.predictions.decoder.we
ight', 'cls.seq_relationship.weight', 'cls.seq_relationship.bias', 'cls.predictions.transform.LayerNorm.weight', 'cls.predict
ions.transform.LayerNorm.bias']
- This IS expected if you are initializing BertForTokenClassification from the checkpoint of a model trained on another task
or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPretraining model).
- This IS NOT expected if you are initializing BertForTokenClassification from the checkpoint of a model that you expect to b
e exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).
Some weights of BertForTokenClassification were not initialized from the model checkpoint at bert-base-cased and are newly in
itialized: ['classifier.weight', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
HBox(children=(HTML(value='Downloading'), FloatProgress(value=0.0, max=213450.0), HTML(value='')))
```

In [19]:
```
model.train_model(train_data,eval_data = test_data,acc=accuracy_score)
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=47959.0), HTML(value='')))
HBox(children=(HTML(value='Epoch'), FloatProgress(value=0.0, max=1.0), HTML(value='')))
HBox(children=(HTML(value='Running Epoch 0 of 1'), FloatProgress(value=0.0, max=1499.0), HTML(value='')))
/usr/local/lib/python3.6/dist-packages/torch/optim/lr_scheduler.py:123: UserWarning: Detected call of `lr_scheduler.step()` b
efore `optimizer.step()`. In PyTorch 1.1.0 and later, you should call them in the opposite order: `optimizer.step()` before `
lr_scheduler.step()`.  Failure to do this will result in PyTorch skipping the first value of the learning rate schedule. See
more details at https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
/usr/local/lib/python3.6/dist-packages/torch/optim/lr_scheduler.py:231: UserWarning: To get the last learning rate computed b
y the scheduler, please use `get_last_lr()`.
  warnings.warn("To get the last learning rate computed by the scheduler, "
/usr/local/lib/python3.6/dist-packages/torch/optim/lr_scheduler.py:200: UserWarning: Please also save or load the state of th
e optimzer when saving or loading the scheduler.
  warnings.warn(SAVE_STATE_WARNING, UserWarning)
```

Out[19]:
```
(1499, 0.19463730062680534)
```

In [20]:
```
result, model_outputs, preds_list = model.eval_model(test_data)
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=46695.0), HTML(value='')))
HBox(children=(HTML(value='Running Evaluation'), FloatProgress(value=0.0, max=1460.0), HTML(value='')))
```

In [21]:
```
result
```

Out[21]:
```
{'eval_loss': 0.17127630058676005,
 'f1_score': 0.7936661066848524,
 'precision': 0.8275694613063235,
 'recall': 0.7624312923430909}
```

In [22]:
```
prediction, model_output = model.predict(["What is the new name of Bangalore"])
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=1.0), HTML(value='')))
HBox(children=(HTML(value='Running Prediction'), FloatProgress(value=0.0, max=1.0), HTML(value='')))
```

In [23]:
```
prediction
```

Out[23]:
```
[[{'What': 'O'},
  {'is': 'O'},
  {'the': 'O'},
  {'new': 'O'},
  {'name': 'O'},
  {'of': 'O'},
  {'Bangalore': 'B-GEO'}]]
```

CONCLUSION:

In conclusion, the project of Named Entity Recognition (NER) using the BERT model has yielded promising results. By leveraging BERT's advanced contextual embeddings and fine-tuning techniques, we have achieved state-of-the-art performance in identifying and classifying named entities in text. The model has demonstrated robustness and accuracy across various domains, making it a valuable tool for tasks requiring precise entity recognition, such as information extraction, question answering, and sentiment analysis.

Moving forward, further refinements and optimizations could enhance the model's efficiencyand applicability in real-world scenarios, paving the way for more advanced NLP applications.