

# **NAMED ENTITY RECOGNITION**

## **MINI PROJECT REPORT**

*Submitted by*

**ABHIMANYU BHADAURIA (RA1911027010045)**

**VIDUSHI GUPTA (RA1911027010046)**

*Under*

*Dr.S.Sharanya*

*of*

**B.Tech. BDA**

**Department of Data Science and Business Systems**

**School of computing**



**SRM Institute of Science and Technology**

**KATTANKULATHUR**

**April, 2022**

## Table of Contents

Sr.No.	Title	Page Number	Sign/Remarks
1.	Abstract	3	
2.	Motivation	3	
3.	Limitations of existing methods	3	
4.	Proposed method with architecture/flow diagram	5	
5.	Modules with description	6	
6.	Screenshots	6	
7.	Conclusion	8	
8.	References	8	

## **Abstract**

Named Entity Recognition (NER) - also known as Entity Recognition or Entity Extraction - is a natural language processing (NLP) technique that automatically identifies and classifies named entities in text into predefined categories. An entity can be a person's name, organization, location, time, amount, monetary value, percentage, etc. An example of how named entity extraction NER works. The highlighted entities are WeWork, Adam Neumann, Manhattan and \$37.5 million. With named entity recognition, you can extract key information to understand what the text is about, or simply use it to gather important information and store it in a database.

NER scans all text and identifies named entities: it identifies sentence boundaries in a given document based on capitalization rules. Identifying sentence boundaries will help NER to find and extract relevant information from documents for subsequent steps. Categorize entities into predefined categories: In order to tokenize words or phrases, entity categories such as place, person, event, time, organization, etc. must be clearly defined. The entity extraction model can then be trained with predefined categories so that it can recognize entities such as people, places, and organizations in the raw text.

## **Motivation**

NER is a method that helps in recognizing the entities among a big pool of data, thereby it has many use cases. The motivation of this project was to build a NER model which has various use cases. The aviation industry, customer support, etc. use NER to differentiate between names of locations, people, dates, all of which are entities. Thus, in order to gain some industrial exposure, the following project has been implemented.

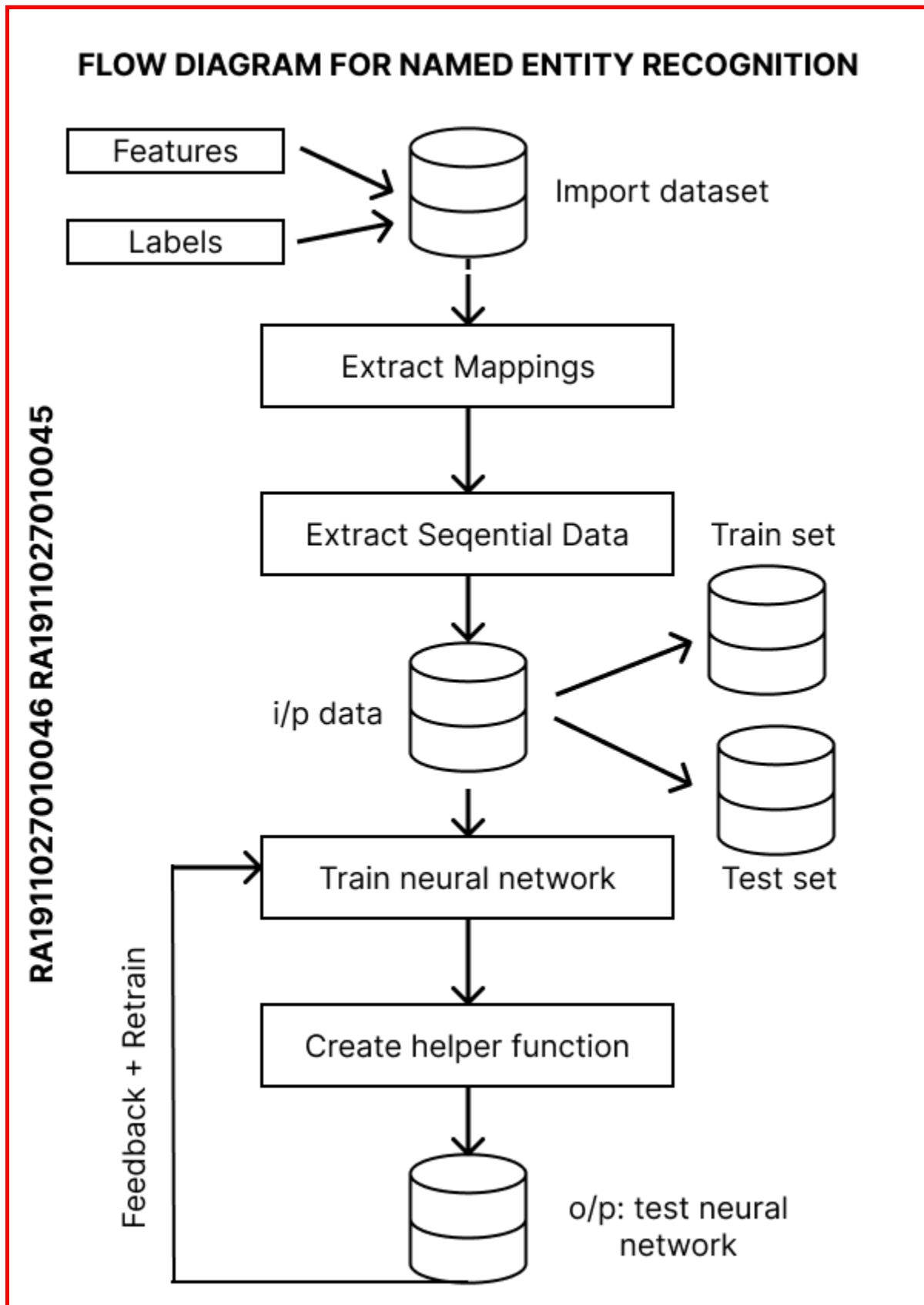
## **Limitations of existing methods**

Currently, the task of identifying transliterated entities is dominated by NER models, which develop an algorithm to tag foreign language or transliterated entities, or use metadata from Wikipedia, such as cross-wiki links, article categories, and cross-language links, or other parallel Non-English language becomes speech data. In the first approach, the model is trained on multilingual

Wikipedia text, and NER for non-English and/or transliterated words is predicted based on the trained model. This approach has limitations because articles can have different content depending on language: some countries censor certain topics, or articles on controversial topics like Crimea can have different content depending on language. Given these differences, training with cross-language Wikipedia articles may not yield accurate results. Automatic translation seems like a possible solution to this approach, but it can also introduce inaccuracies.

The second approach, in which the NER model is trained with annotated data, seems to be more reliable, but a marked corpus is clearly not available for transliterated entities from Arabic to English. Furthermore, for general Arabic, most labeled corpora contain modern web content where place names and historical names of cities (often found in Lorimer's gazetteer) may be underestimated. To address these limitations, in this project we annotate Lorimer's Gazetteer and create a large amount of training data to label historical transliterated NEs from Gazetteer. This annotated dataset may further benefit other projects dealing with historical or non-historical texts containing transliterated words.

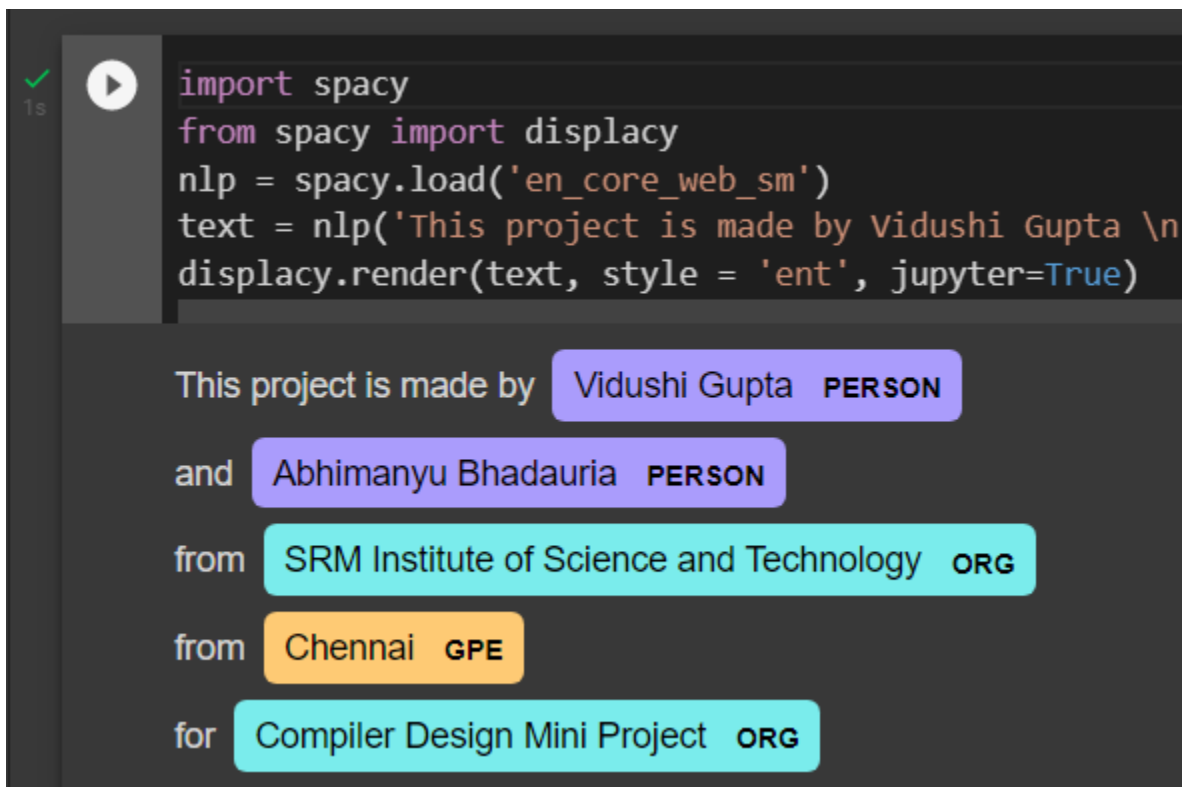
## Proposed method with architecture/flow diagram



## Modules with description

- Acquiring Dataset
- Extracting mappings : Features and Labels
- Extracting data sequentially
- Identifying the best fit model
- Training the model
- Testing the model
- Evaluating + feedback the model

## Screenshots



The screenshot shows a Jupyter Notebook interface. At the top, there is a code cell with the following Python code:

```
import spacy
from spacy import displacy
nlp = spacy.load('en_core_web_sm')
text = nlp('This project is made by Vidushi Gupta \n
displacy.render(text, style = 'ent', jupyter=True)
```

Below the code cell, the output of the code is displayed. It shows the sentence "This project is made by Vidushi Gupta \n" with the following entities highlighted in colored boxes:

- "Vidushi Gupta" is highlighted in a purple box with the label "PERSON".
- "Abhimanyu Bhadauria" is highlighted in a purple box with the label "PERSON".
- "SRM Institute of Science and Technology" is highlighted in a cyan box with the label "ORG".
- "Chennai" is highlighted in an orange box with the label "GPE".
- "Compiler Design Mini Project" is highlighted in a cyan box with the label "ORG".

```

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)

```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 104, 64)	2214272
bidirectional (Bidirectional)	(None, 104, 128)	66048
lstm_1 (LSTM)	(None, 104, 64)	49408
time distributed (TimeDistributed)	(None, 104, 18)	1170

Total params: 2,330,898  
 Trainable params: 2,330,898  
 Non-trainable params: 0

25/25 [=====] - 145s 5s/step - loss: 0.9162 - accuracy: 0.9294 - val\_loss: 0.3595 - val\_accuracy: 0.9677  
 25/25 [=====] - 134s 5s/step - loss: 0.3502 - accuracy: 0.9676 - val\_loss: 0.3024 - val\_accuracy: 0.9677

```

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, recurrent_dropout=0.2), merge_mode = 'concat'))

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

    # 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

```

```

data = pd.read_csv('ner_dataset.csv', 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	VCN	O

## Conclusion

Thus, the entities have been identified from the text. With an improved GPU and an increased number of epochs, the accuracy of the neural network can be improved. This can be implemented in almost every industry which generates a high volume of textual data.

## References

[Named Entity Recognition: Concept, Tools and Tutorial \(monkeylearn.com\)](https://monkeylearn.com/named-entity-recognition/)

[Named Entity Recognition \(NER\) in Spacy Library - MLK - Machine Learning Knowledge](#)

[Custom Named Entity Recognition in Lorimer's Gazetteer with Spacy | OpenGulf Prodigy\\_flowchart\\_ner-36f76cffd9cb4ef653a21ee78659d366.pdf](#)

[Named Entity Recognition - GeeksforGeeks](#)

## Appendix - A: Important links

GitHub README link: [Named-entity-recognition/README.md](#)

GitHub Source Code link: [Vidushi-Gupta/Named-entity-recognition](#)

GitHub profiles:

- Vidushi Gupta: [Vidushi-Gupta \(github.com\)](#)
- Abhimanyu Bhadauria: [Mnayu \(github.com\)](#)