International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 6 Issue 1, November-December 2021 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Custom Named Entity Recognition (CNER)

Prince Nathan S[1], Onkar Saudagar[2], Rutika Shinde[3]

[1]PG Student, Department of Data Science and Analytics,
[1]National Institute of Electronics and Information Technology, Chennai, Tamil Nadu, India
[2,3]Student, Department of Information Technology,
[2]Pune Institute of Computer Technology, Pune, Maharashtra, India
[3]Sinhgad Institute of Technology and Science, Pune, Maharashtra, India

ABSTRACT

Named Entity Recognition (NER) and Disambiguation are Natural Language Processing (NLP) subtasks that aim to recognise and categorise named entities in text into their appropriate categories. CNER enhances the capacity by assisting you in identifying new entity types that are not included in the preset generic entity types. This implies that, in addition to detecting entity categories like DISEASE, LOCATION, DATE, and PERSON, you can also examine documents and extract entities like product codes or business-specific entities that are relevant to your needs.

Due to a lack of large-scale labelled training data and domain expertise, biomedical named entity recognition (BioNER) is a difficult problem for interpreting bio- logical literature. In addition to employing sophisticated encoders (e.g., biLSTM and BioBERT) to solve the problem, one option is to use readily available supple- mentary information.

The goal of Bio-medical Diseased Named Recognition is to recognise various disorders. In this study, I'll use spaCy in Python to do custom named entity (Disease) recognition in clinical literature. In this project, I'll build a clinical named entity recognition model that can recognise disease names in clinical text.

KEYWORDS: *Custom named entity, Biomedical named entity recognition, NER, Bio-medical text, clinical text, SpaCy*

How to cite this paper: Prince Nathan S | Onkar Saudagar | Rutika Shinde "Custom Named Entity Recognition (CNER)" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-6 | Issue-1, December 2021, pp.1798-1806, URL: www.ijtsrd.com/papers/ijtsrd49180.pdf

IJTSRD49180

1. INTRODUCTION

Named entity recognition (NER) is a sub-task of information extraction (IE) that looks for and categorises specific entities in a body of texts. Entity identification, entity chunking, and entity extraction are all terms used to refer to NER. Named entity recognition(NER) is utilised in a variety of AI applications, including Natural Language Processing (NLP) and Machine Learning. Entities are terms in a text that denote a certain category of data. They might be nu- merical, like cardinal numbers; temporal, like dates; nominal, like people's and places' names; and political, like geopolitical entities (GPE). In other words, an entity may be anything that the designer wants to designate as an item in a text with a label. The process by which a system takes unstructured data (a text) and converts it into structured data, especially the identification of entities, is known as named entity recognition, or NER.

The task of Named Entity Recognition involves identifying and pointing out the strings that fall into some named predetermined entity class(es), in the text [1]. Although, attempts were made to create fixed rigid designators as entities for NER, in common practice one must deal with numerous referents that cannot be considered philosophically "rigid". This can be clearly shown by discussing the following example:

"It was an interesting time for the Ford Motor Company."

Here we can easily recognize that the string ***"Ford Motor Company"*** refers to the organization, yet we

must not overlook the fact that the word ***"Ford"*** can easily refer to many entities [1].

Evolution of NER [2] At the sixth Message Understanding Conference (MUC- 6), the term "Named Entity" (NE) was first used to describe the problem of identifying names of companies, persons, and geographic locations in text, as well as currency, time, and percentage expressions. Since MUC-6, there has been a surge in interest in NER, and a number of scientific conferences (e.g., CoNLL03, ACE, IREX, and TREC Entity Track) have focused on it.

"A NE is a proper noun that serves as a name for something or someone," says the problem definition. The significant percentage of proper nouns present in a corpus justifies this restriction. Some claimed that the term "Named" limited the task to entities with one or more *rigid designators* serving as the referent. Despite the various definitions of NEs, researchers have come to an agreement on the types of NEs that should be recognised. Generic NEs (e.g., person and location) and domain-specific NEs(e.g., proteins, enzymes, and genes) are the two types of NEs that we use . In this paper, we mainly focuson domain-specific NEs.

Custom named entity recognition (CNER) expands the functionality by assisting you in identifying new entity kinds that are not included in the pre- defined generic entity types. This means that, in addition to detecting entity categories such as DISEASE, LOCATION, DATE, and PERSON, you can also examine documents and extract entities such as product codes or business-specific entities that are relevant to your needs.

Biomedical named entity recognition (BioNER) is an important task for reading biomedical literature, but it might be difficult due to a lack of large- scale labelled training data and domain knowledge. To address the challenge, In addition to using powerful encoders (e.g., biLSTM and BioBERT) to solve the problem, one option is to use readily available supplementary knowledge. The goal of Bio-medical Diseased Named Recognition is to recognise various disorders. In this study, I'll use SpaCy in Python to do custom named entity (Disease) recognition in clinical text.

Clinical named entity recognition (CNER) is a crucial task for extracting patient data from electronic health records (EHRs) in order to facilitate clinical and translational research. CNER's major goal is to recognise and classify clinical terminology in electronic health records (EHRs), such as diseases, symptoms, therapies, tests, and body parts. A crucial step in any clinical Natural Language Processing (NLP) system is to identify these important clinical ideas.NER in the medical domain is more difficult than in the generic domain. On the one hand, clinical texts frequently use non-standard abbreviations or acronyms, as well as various versions of the same entity. Clinical notes, on the other hand, are noisier, more prone to grammatical errors, and include less context due to shorter and incomplete sentences. Furthermore, due to the diversity of EHRs, constructing such a CNER is a difficult undertaking [5].

SpaCy is a Python-based open-source library that performs advanced natural language processing. It is intended for production use and aids in the development of applications that process and "understand" massive amounts of text. It may be used to create data extraction and natural language understanding systems, as well as to pre-process text for deep learning. Tokenization, Parts-of-Speech (PoS) Tagging, Text Classification, and Named Entity Recognition are some of the functionalities provided by spaCy. In Python, SpaCy provides an extremely efficient statistical method for NER that can give labels to contiguous clusters of tokens. It comes with a default model that can recognise a wide range of named or numerical entities, such as people, organisations, languages, and events. Apart from these preset entities, spaCy also allows us to add additional classes to the NER model by updating the model with newer trained examples.

SpaCy NER already supports entity types such as- People, including fictional characters. Nationalities, religious, and political groups are all examples of this. Structures such as buildings, airports, highways, and bridges, and many others. Companies, agencies, and institutions, to name a few, Countries, cities, states, and other entities. Our goal is to refine this model so that it can account for our own custom entities in the dataset.

2. LITERATURE SURVEY

The Following table shows the literature survey by comparing techniques pro- pose in various references:

Sr No.	Paper	Summary	Gap
1	Automated Custom Named Entity Recognition and Disambiguation	1. Introduction on NLP (Natural language Processing). 2. The use of NER ranges from profanity detection to extracting meta-data from documents. 3. A novel fust approach (FastEnt) to tackle the task of identifying and detecting Custom Named Entities (CNE).	1. The system can be improved by the addition of Bidirectional LSTM models and more model training routines.
2	A Survey on Deep Learning for Named Entity Recognition	1. Introduction on Named entity recognition (NER). 2. How in early Days, NER system got a huge success in achieving good performance with the cost of human engineering in designing domain-specific features and rules. 3. The future directions in the area of NER(Named Entity Recognition) and the challenges faced by the present readers.	1. The greatest shortcoming of the classical NER models is the limited number of predefined classes that are set in the task (i.e. Person (PER), Location (LOC), Companies/institutions (ORG) etc.).
3	Named entity recognition on bio-medical literature documents using hybrid based approach	1. Extracting the drug names, diseases, symptoms, route of administration, species, and dosage forms from the textual document in the Natural Language Processing. 2. A new hybrid based approach is proposed to identify named entity from the medical literature documents by the blank Spacy machine learning model.	1. The average F1 score for five entities of the proposed hybrid-based approach is 73.79%. 2. The Quantity of entities is less. Enriching the dictionaries by adding more objects will give more accuracy.
4	Effect of Character and Word Features in Bidirectional LSTM-CRF for NER	1. Studied the impact of various features and their combination on the neural network model. 2. The model incorporates two layers of Bidirectional LSTM with CRF without preprocessing and lexicon that achieves the 91.10% F1 score on the CoNLL-2003 dataset.	1. The performance of the model can be further enhanced by using converting the dataset from IOB to IOBES tagging scheme. 3. It can be explored in multi-task learning approaches to combine more useful and correlated information among different NLP tasks.
5	Incorporating dictionaries into deep neural networks for the Chinese clinical named entity recognition	1. Proposed several effective approaches for the Chinese CNER by integrating dictionaries into neural networks. 2. Use of the Bi-LSTM layer to improve the system efficiency.	1. Since Bi-LSTM has double LSTM cell so it is costly. 2. Not good fir for speech Recognition. 3. Hard to train bit more accurate.

3. RELATED RESEARCH

3.1. NER Software Evaluation

3.1.1. NER: Concepts & Functioning

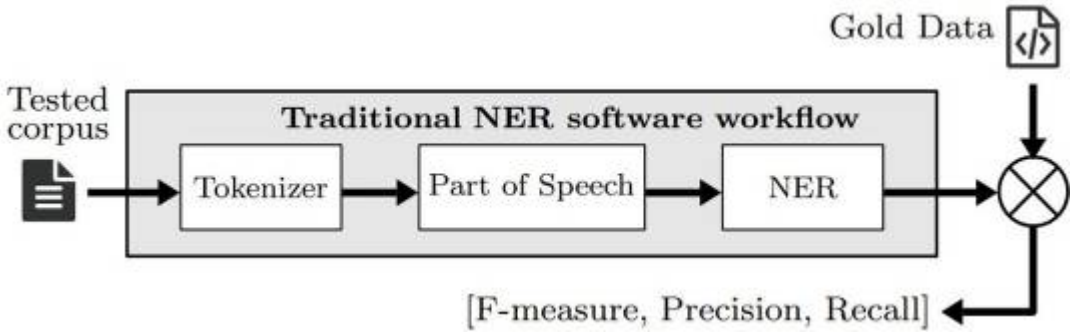

Figure 1: Traditional evaluation methodology of NER software performance

The goal of NER is to recognise and semantically classify words/entities in a text. As shown in Fig.1, traditional NER software consists of three main tasks. The workflow depicted in Fig.1 is commonly used to evaluate the performance of any given NER software. As inputs, both a corpus of reference and the associated "gold data" are used in order to evaluate the extent to which the identification/classification resulting from the NER software matches with the gold data [11].

3.1.2. NER Software Evaluation

Several studies in the literature have been conducted to determine which NER software performs the best. Fig.2 present an overview of the most notable evaluation studies in this field, including details on the corpus, software, and metrics used for evaluation[11].

- NER Software : First, it should be noted that NER software is either de- veloped for use in a specific domain (e.g., social media, healthcare) or in a generic manner, as evidenced by the set of evaluated NER software (i.e., being domain-independent). To date, StanfordNLP1, NLTK, SpaCy, and OpenNLP are the most well-known software for general use, with each having its own set of features and tuning. Stanford NLP is a JAVA toolkit that includes a variety of tools such as PoS, NER tagger, and others. SpaCy is known for its parsing speed, whereas NLTK provides a diverse set of libraries and modules for symbolic and statistical NLP.
- Corpus : The selection of the corpus, as discussed in section 4.1.1 , is an important step in the performance evaluation process. CoNLL 2003, MUC- 6, MUC-7 using newswire, or ACE 2005 using weblogs, broadcast news, newsgroups, and broadcast conversations are just a few examples of anno- tated corpora created specifically for scholars. Other corpora were created over time to expand or diversify the nature of already existing corpora like OntoNotes and WikiGold; or more specialised corpora like Ritter Twitter and UMBC.

Corpus	Software	License	Classifier	Version
	OpenNLP	Apache Software Lic.	N/A	N/A
	StanfordNLP	GNU GPL	CoNLL, ACE, MUC	**3.6.0**
CoNLL 2003	NLTK	Apache Lic. v2	ACE	N/A
Ritter	Pattern	BSD	N/A	N/A
MSM2013	TweetNLP	GPL v2	N/A	N/A
	TweeterNLP	GNU GPL	N/A	N/A
	TwitIE	N/A	N/A	N/A
	SpaCy	MIT License	N/A	N/A
Wikigold	StanfordNER	GNU GPL	CoNLL MUC6/7, ACE	v3.6
	NLTK	Apache Lic. v2	N/A	N/A
	Alias-i LingPipe	Royalty Free Lic. v1	MUC6	4.1
	Annie (Gate)	GPL v3	Gazetter	Gate v8
	StanfordNER	GNU GPL	CoNLL, ACE	N/A
	DBpedia Spotlight	Apache Lic. v2	N/A	N/A
MSM2013	Lupedia	N/A	N/A	N/A
Ritter	Ritter et al.	GPL v3	N/A	N/A
UMBC	Alchemy API	Non Commercial	N/A	N/A
	NERD-ML	GPL v3	N/A	N/A
	YODIE	N/A	N/A	N/A
	Zemanta	Non Commercial	N/A	N/A
	TextRazor	Non Commercial	N/A	N/A
CoNLL 2003	StanfordNLP	GNU GPL	N/A	N/A
	Annie (Gate)	GPL v3	N/A	N/A
OntoNotes 5	SpaCy	MIT License	Small, Medium, Large OntoNotes 5	v2.x
CoNLL 2003	StanfordNLP	GNU GPL	N/A	N/A

Figure 2: EXPERIMENTAL CONDITIONS OF NER SOFTWARE EVALUA- TION STUDIES IN THE LITERATURE.

3.2. Different Existing Approaches

The existing algorithms for the NER task can be divided into four categories: rule-based approaches, dictionary-based approaches, statistical machine learning approaches, and deep learning approaches, which have recently received more attention from the CNER community.

3.2.1. Rule-based Approach

The rule-based algorithm applies a set of rules in order to extract patterns, i.e., rule base for Malay NER [4]. Most of the rules are manually generated. The dictionaries are maintained separately for diseases, chemicals, genes, etc [3]. Rule-based approaches rely on heuristics and handcrafted rules to identify entities. They were the

dominant approaches in the early CNER systems [5] and some recent work. However, it is difficult to list all rules to model the structure of clinical named entities, and this kind of handcrafted approaches always leads to a relatively high system engineering cost.

3.2.2. Dictionary-based Approach

Dictionary-based approaches rely on existing clinical vocabularies to identify enti- ties [8]. They were widely used because of their simplicity and their performance. A dictionary-based CNER system can extract all the matched entities defined in a dictionary from a given clinical text. However, it cannot deal with entities which are not included in the dictionary, and usually causes low recalls.

3.2.3. Machine Learning Approach (Supervised or Unsupervised)

Statistical machine learning approaches consider CNER as a sequence labeling problem where the goal is to find the best label sequence for a given input sentence. Typical methods are Hidden Markov Models (HMMs), Maximum Entropy Markov Models (MEMMs), Conditional Random Fields (CRFs), and Support Vector Machines (SVMs) [5].

3.2.4. Deep Learning Approach

Recently, deep learning approaches [5], especially the methods based on Bidirectional Recurrent Neural Network (RNN) using CRF as the output interface (Bi-RNN-CRF) [9],achieve state-of-the-art performance in CNER tasks and out- perform the traditional statistical models [10].

With the emergence of the machine and deep learning, various models were pro- posed for the task. The machine learning approach involves the usage of structured and unstructured techniques, such as CRF that was implemented for DrugNER .

4. DIFFERENT EXISTING MODELS FOR NAMED ENTITY RECOGNITION

4.1. Hidden Markov Models

In speech and language processing, the Hidden Markov Model is one of the most important machine learning models. Hidden Markov Models (HMMs) are doubly embedded stochastic processes that are used to model a variety of situations characterised by the evolution of some events that are influenced by some internal factors. Internal factors are referred to as states, whereas events are referred to as observations.

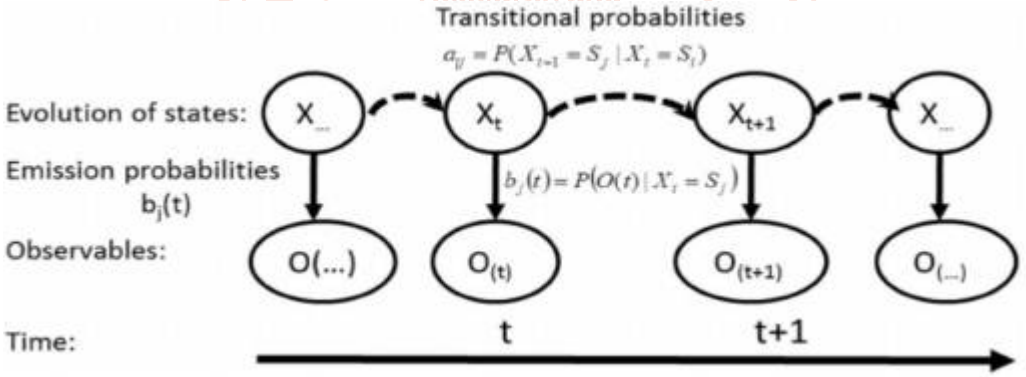

Figure 3: HMM model visualization - transition states

HMMs can be thought of as a closed system with n states that is required to reside in one of the states at a given fixed point in time and that can also make transitions between those n states with some predetermined probability while emitting observations with another predetermined probability set.

4.2. Conditional Random Fields

CRFs are a type of sequence modelling technique used for structured prediction, and they have found a useful application in NER. CRFs are a type of discriminative undirected probabilistic graph model that is used to encode known relationships between observations while generating consistent interpretations.

4.3. Neural Networks

Artificial Neural Networks (ANNs) are a popular tool for solving binary and multi- class classification problems. NN architectures such as Feed forward Neural Net- work (FNN - the basic ones), Recursive Neural Network (RNN), and Convolutional Neural Network (CNN) have been developed over several decades (CNN). In this section, we'll go over the fundamentals of the structures mentioned above.

The most important concept to grasp is a Deep Neural Network, which is made up of several layers of artificial neurons, each of which is a basic computational node in its own right.

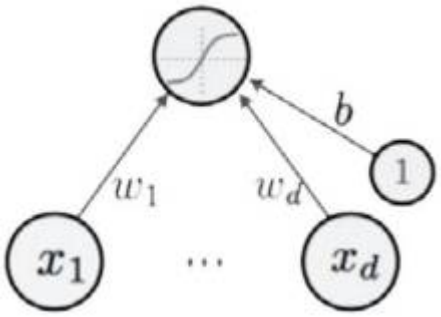

Figure 4: Single artificial neuron

A sequence of numbers x_i is fed into each computational node (neuron). A bias b is added to the sum of multiplications after each x_i is multiplied by its corresponding weighting coefficient. The neural node function receives this sum as an input, and the result is the neuron's final output. For a neuron function, there are several well-known choices, such as *tanh or sigmoid*. A single sigmoid neuron is limited to categorising data into two basic categories and setting a threshold. To make a more advanced classifier, multiple neurons are combined to form a layer, which is then combined with several layers to form a classic FNN. As shown below Fig.5, a standard FNN is made up of three types of neural layers: input, hidden, and output layers.

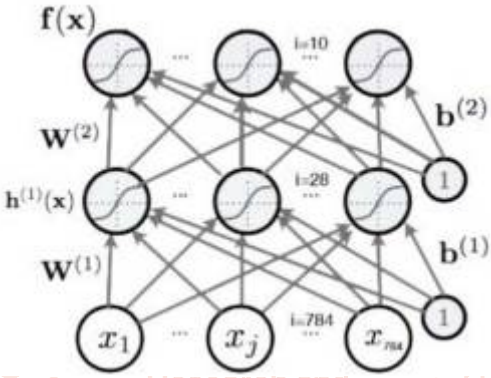

Figure 5: Multi-layer Neural Network

In order to solve the classification problem, the network is first initialised with random weights and biases for each neural cell, which are then optimised in terms of the error function. In the sense that it has more layers of convolution and pooling, the Convolutional NN represents a step forward. Convolutional layers differ from typical neural layers in that each neuron receives only a portion of the inputs as input. The core premise of CNN is to divide neurons into subgroups, generating a feature map, with each subgroup optimising its recognition of a given "feature" in the data. Finally, pooling layers are employed to filter away unnecessary feature map subgroups, attempting to rid our classifier of irrelevant "features."

5. METHODOLOGY

5.1. Overview

5.1.1. Custom Named Entity (Disease) Recognition in Clinical Text with SpaCy 2.0 in Python

I perform Custom Named Entity (Disease) Recognition in clinical text with spaCy in Python. Here I will be creating a clinical named entity recognition model which can recognize the disease names from clinical text.

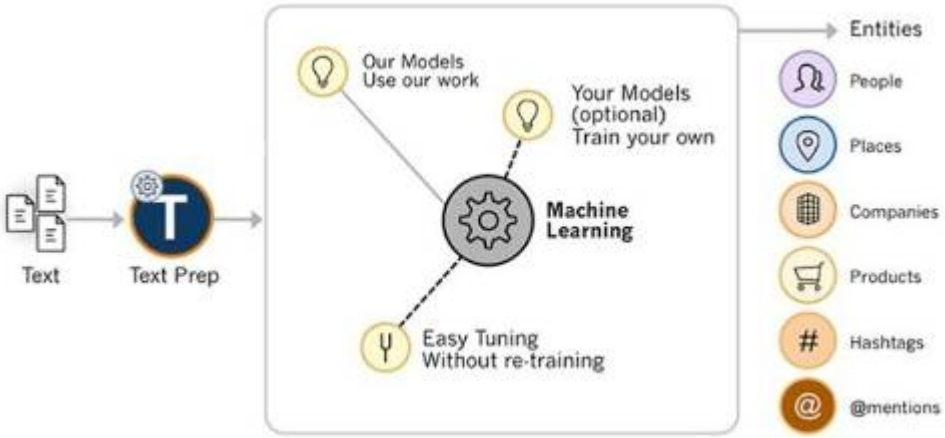

Figure 6: An example of CNER in action

The fixed number of entity classes used by current systems is one of the most limiting factors of NER. Indeed, the most commonly used CoNLL entity tags do not cover the vast majority of likely entities that a user might want to segment. This problem was solved by SpaCy by using a collection of classes.

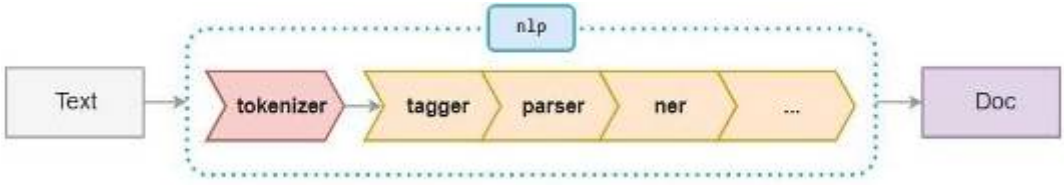

Figure 7: The Spacy NLP Pipeline

5.2. Workflow

5.2.1. Dataset and Pre-Processing Steps

6.2.1.1. Loading the Dataset into Workplace available on kaggle and in differnt plat- forms NER Dataset.
6.2.1.2. Each dataset having the following files: train.tsv, test.tsv, dev.tsv and de- vel.tsv.
6.2.1.3. These are tab separated files each word is annotated using the BIO format.
6.2.1.4. In BIO format : B stands for begin of entity , I stands for inside entity and O stand for outside entity or any other word.

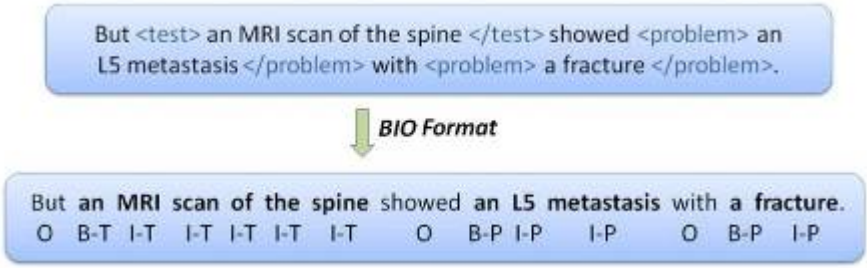

Figure 8: An example of BIO Format (T = Test, P = Problem)

5.2.2. Training a Custom Named Entity (Disease Recognition in Clini- cal Text)

➢ Let us define methods to compute Precision, Recall and F1-score

5.2.2.1. TP(True Positive) : Word predicted as either I Disease or B Disease and present in the data(train/test/validation) as either i Disease or B Disease.
5.2.2.2. FP(False Positive) : Word predicted as either I Disease or B Disease and not present in the data(train/test/validation) as either i Disease or B Disease.
5.2.2.3. FN(False Negative) : Word predicted in the data(train/test/validation) data as either I Disease or B Disease and not predicted as either i Disease or B Disease.

➢ Matrix:

1. Precision : Number of predicted entity string spans that line up exactly with spans in the evaluation data.

Precision = TP / (TP + FP)

2. Recall : Number of names in the evaluation data that appear at exactly the same location in the predictions.

Recall = TP / (TP + FN)

```
def calc_precision(pred, true):
    precision = len([x for x in pred if x in true]) / (len(pred) + 1e-20) # true positives / total pred
    return precision

def calc_recall(pred, true):
    recall = len([x for x in true if x in pred]) / (len(true) + 1e-20)    # true positives / total test
    return recall

def calc_f1(precision, recall):
    f1 = 2 * ((precision * recall) / (precision + recall + 1e-20))
    return f1
```

Figure 9: Precisio, Recall, F1-score

3. F1-score : This metric is the harmonic mean of Precision and Recall.

F1-score = 2 * Precision Recall / (Precision + Recall)

➢ I use an existing model ”en core web md”(English medium sized model).This is a CNN model. This model by default has POS tagger, Dependency parser and Named entity recognition functionalities. We only re-train the named entity recognition part of the model.

```
nlp = spacy.load("en_core_web_md")
# nlp = spacy.blank('en')
if 'ner' not in nlp.pipe_names:
    ner = nlp.create_pipe('ner')
    nlp.add_pipe(ner)
else:
    ner = nlp.get_pipe("ner")
# Add entity labels to the NER pipeline
for i in labels:
    ner.add_label(i)
# Disable other pipelines in SpaCy to only train NER
other_pipes = [pipe for pipe in nlp.pipe_names if pipe != 'ner']
 with nlp.disable_pipes(*other_pipes):
     #nlp.vocab.vectors.name = 'spacy_model' # without this, spaCy throws an "unnamed" error
     optimizer = nlp.create_optimizer()
     for itr in range(iterations):
         random.shuffle(train_data) # shuffle the training data before each iteration
         losses = {}
         batches = minibatch(train_data, size = compounding(16.0, 64.0, 1.5))
         for batch in batches:
             texts, annotations = zip(*batch)
             example = []
             # Update the model with iterating each text
             for i in range(len(texts)):
                 doc = nlp.make_doc(texts[i])
                 example.append(Example.from_dict(doc, annotations[i]))

             nlp.update(
                 example,
                 drop = dropout,
                 sgd = optimizer,
                 losses = losses)
         scores = evaluate(nlp,VALID_DATA)
         valid_f1scores.append(scores["textcat_f"])
```

Figure 10: Train a SpaCy NER model, which can be queried against with test data

- Dropout is a regularization technique for reducing over fitting in neural net- works by preventing complex co-adaptations on training data. The term dropout refers to randomly ”dropping out”, or omitting, units (both hidden and visible) during the training process of a neural network. In our case if dropout = 0.5 there is a 50% dropping out omitting units during training process of our model.

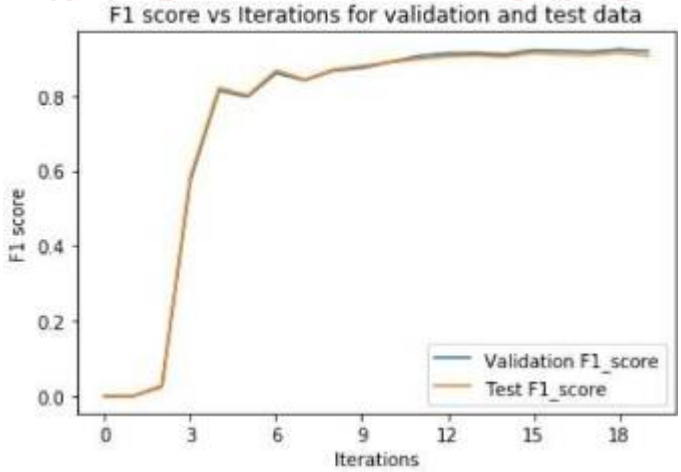

Figure 11: F1-score vs Validation data

- Here we can also see that in the initial phase of iterations basically the model is not learned something but after a couple of interactions it actually starts learning then as the iteration increases the F1- score is also increasing. The last interactions F1-score is very close to each other hence we can say the algorithm is performed really well in the learning process.

6. CONCLUSION

Named Entity Recognition (NER) plays a key role in the detection and classification of entities in NLP applications. An end-to-end framework for Custom Named Entity Recognition was developed during the research process. As a result, our model achieved state-of-the-art performance in terms of F1 score, precision and recall. I successfully trained a custom named entity recognition model in clinical text with spacy to detect your custom entities. Here I successfully detect the disease entities with 0.91 or 91% F1-score and validation data. Although our model requires a large amount of memory and time.

It must be noted that the modules in the system are mostly (only dataset generation requires explicit word vectors, which are initially supplied to be for English) language independent. Although a comprehensive routine for complete custom NER has been developed, the system can be improved by adding Bidirectional LSTM models and more model training routines.

References

[1] Arakelyan, Erik Bittlingmayer, Adam Stepanyan, Levon. (2017). Automated Custom Named Entity Recognition and Disambiguation.

[2] J. Li, A. Sun, J. Han and C. Li, March 2020, ”A Survey on Deep Learning for Named Entity Recognition,” , in IEEE Transactions on Knowledge and Data Engineering, doi: 10.1109/TKDE.2020.2981314.

[3] Ramachandran R, Arutchelvan K., March 2021, Named entity recognition on bio-medical literature documents using hybrid based approach. J Ambient Intell Humaniz Comput. 11:1-10. doi: 10.1007/s12652-021-03078-z. Epub ahead of print. PMID: 33723489; PMCID: PMC7947151.

[4] C. Ronran and S. Lee, 2020, ”Effect of Character and Word Features in Bidirectional LSTM-CRF for NER,”, IEEE International Conference on Big Data and Smart Computing (BigComp), 2020, pp. 613-616, doi: 10.1109/Big- Comp48618.2020.00132.

[5] J. Qiu, Y. Zhou, Q. Wang, T. Ruan and J. Gao,July 2019, ”Chinese Clinical Named Entity Recognition Using Residual Dilated Convolutional Neural Net- work With Conditional Random Field,” in IEEE Transactions on NanoBio- science, vol. 18, no. 3, pp. 306-315, doi:10.1109/TNB.2019.2908678.

[6] Erdmann, Alexander Wrisley, David Allen, Benjamin Brown, Christopher Cohen-Bod´en`es, Sophie Elsner, Micha Feng, Yukun Joseph, Brian JoyeuxPrunel, B´eatrice Marneffe, Marie-Catherine. (2019). Practical, Efficient, and Customizable Active Learning for Named Entity Recognition in the Digital Humanities. 2223-2234. 10.18653/v1/N19-1231.

[7] Segura-Bedmar, Isabel Su´arez-Paniagua, V´ıctor Martinez, Paloma. (2015). Exploring Word Embedding for Drug Name Recognition. 64-72. 10.18653/v1/W15-2608.

[8] Song, M., Yu, H. Han, WS. (2015) Developing a hybrid dictionary-based bio-entity recognition technique. BMC Med Inform Decis Mak 15, S9.

[9] Maryam Habibi, Leon Weber, Mariana Neves, David Luis Wiegandt, Ulf Leser, 15 July 2017, Deep learning with word embeddings improves biomed- ical named entity recognition, Bioinformatics, Volume 33, Issue 14, Pages i37–i48,

[10] X. Yang et al., ”Bidirectional LSTM-CRF for biomedical named entity recognition,” 2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2018, pp. 239-242, doi: 10.1109/FSKD.2018.8687117.

[11] Schmitt, Xavier Kubler, Sylvain Robert, J´er´emy Papadakis, Mike LeTraon, Yves. (2019). A Replicable Comparison Study of NER Software: Stanford NLP, NLTK, Open NLP, SpaCy, Gate. 338-343. 10.1109/SNAMS.2019.8931850.