Hindi Named Entity Recognition.

Submitted in partial fulfillment of the requirements
of the degree of

B. E. Computer Engineering

By

Ankit Jaiswal	**56**	**172015**
Peter Richie Jacob	**57**	**172077**
Aishwarya Sreenivasan	**59**	**172006**

Guide:

Name of the Guide: Mrs Vincy Joseph
Designation: Assistant Professor

Department of Computer Engineering
St. Francis Institute of Technology
(Engineering College)

University of Mumbai
2019-2020

CERTIFICATE

This is to certify that the project entitled **"Hindi Named Entity Recognition."** is a bonafide work of **"Ankit Jaiswal"(56), "Peter Richie Jacob"(57) , "Aishwarya Sreenivasan"(59)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of B.E. in Computer Engineering

Ms. Vincy Joseph
Guide

Dr. Kavita Sonawane
Head of Department

Dr. Sincy George
Principal

Project Report Approval for B.E.

This project report entitled "**Hindi Named Entity Recognition.**" by (***Ankit Jaiswal ,Peter Richie Jacob , Aishwarya Sreenivasan***) is approved for the degree of ***B.E. in Computer Engineering.***

Examiners

1.----------

2.----------

Date:

Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Ankit Jaiswal 56)

(Peter Richie Jacob 57)

(Aishwarya Sreenivasan 53)

Date:

Abstract

Named-entity recognition (NER) (also known as entity identification, entity chunking and entity extraction) is a subtask of information extraction that seeks to locate and classify named entity mentioned in unstructured text into predefined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc. This project aims to create a novel NER system in Hindi which is one of the major Indo-Aryan languages of India. Processing of Hindi language and extraction of named entities is a challenging task. Named Entity Recognition System (NER) will include identification of names that refer to Weather type, Date etc. The NER system will be built using Customised Spacy Framework. The system will be useful for extracting Weather type, location, dates from given text as it is trained on customized dataset on weather.

Contents

List of Figures

List of Abbreviations

Sr. No.	Abbreviation	Expanded form
i	NER	Named Entity Recognition

Chapter 1
Introduction

Named-entity recognition (NER) (also known as named entity identification, entity chunking, and entity extraction) is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into predefined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.

1.1 Description

Named Entity Recognition (NER) is an important task in Natural Language Processing (NLP) applications like Information Extraction, Question Answering etc. In this project, an application is developed to recognize Hindi named entities like weather type, location name, date to determine conditions in various regions. A Hindi Customised dataset has been used for this system to train and test.

1.2 Problem Formulation

Reading complete Weather data and classifying like the location, places, weather type and dates mentioned in it becomes a tedious task to be done manually. A system which finds all these entities quickly can save a lot of reading time. The paragraphs of interest can be then read thoroughly by the user if the required entities exist in the text. The user can quickly understand the context of the entire text by referring only to the entities found and classify text as useful or not. Labelled paragraphs are much more useful than unlabeled text.

1.3 Motivation

For a student or professor studying weather conditions in various locations, reading large amounts of text from books and other sources which are not properly documented becomes a challenging task. For such undocumented data, NER system will help recognize key entities in the text and save a lot of time and energy. In the past few years and with the advent of massive digitization of textual data, NER has become of increasing interest in various sectors due to its potential for information extraction and analysis of large scale documents.

1.4 Proposed Solution

Spacy Framework and Fasttext embeddings are used to make the Hindi NER system. The dataset is made manually on a small scale to train and test the system as a prototype. spaCy also gives us the liberty to add arbitrary classes to the NER model, by training the model to update it with newer trained examples. The features considered are the tag of the current word, the previous word and the next word in addition to the words themselves.The input text also undergoes tagging before NER labelling for better results. The system is designed with an accuracy of 91.05%.

1.5 Scope of The Project

This model recognizes entities in a text which ensures every user gets at least the basic idea of the main roles in the text. The system is developed in Python with a Flask frontend. The system gives better performance on weather data in Hindi. The system is extremely simplified and the results are displayed as tags of Dates, weather-type, location, etc instead of showing the entire text with non-contextual words.

The system does not perform very well for previously unseen samples. The model can be improved to tag phrases instead of just words. The system can be used in summarization systems to give a brief summary of the text instead of just the important entities.

Chapter 2

Review of Literature

Named Entity Recognition (NER) is essential for some Natural Language Processing (NLP) tasks. Previous researchers gave a survey of NER in the statistical machine learning era, however, research on NER has already changed a lot in recent decades. On the one hand, more and more NER systems adopt deep learning, transfer learning, knowledgebase and other methods. On the other hand, multilingual and low resource languages NER researches increase rapidly.

Amongst the various classification models for Named Entity Recognition, spaCy model showed a better performance than any other methods. spaCy features an extremely fast statistical entity recognition system, that assigns labels to contiguous spans of tokens. The default model identifies a variety of named and numeric entities, including companies, locations, organizations and products. You can add arbitrary classes to the entity recognition system, and update the model with new examples.

Named entity is a “real-world object” that’s assigned a name — for example, a person, a country, a product or a book title. spaCy can recognise various types of named entities in a document, by asking the model for a prediction. Because models are statistical and strongly depend on the examples they were trained on, this doesn’t always work perfectly and might need some tuning later, depending on your use case.

To evaluate the quality of a NER system’s output, several measures have been defined. While accuracy on the token level is one possibility, it suffers from two problems: the vast majority of tokens in real-world text are not part of entity names as usually defined, so the baseline accuracy (always predict “not an entity”) is extravagantly high, typically >90%; and mis-predicting the full span of an entity name is not properly penalized (finding only a person’s first name when their last name follows is scored as ½ accuracy).

2.1 Related Work.

While NER has a rich literature, it was not until 2008, a lot of work for NER on Indian languages saw prominence. Initially, NER task was based on language specific designing rules and gazetteer lists. They designed language specific rules, made gazetteer lists to add knowledge to the data and performed NER on Indian Languages. Many approaches were proposed, for example, (Saha et al., 2008) used class specific language rules, gazetteer as features to their Maximum Entropy Model. (Ekbal and Bandyopadhyay, 2008) proposed pair wise multi-class decision method and second degree polynomial kernel function to perform classification on the text data using SVM. Another approach using CRFs (Conditional Random Field) (Gali et al., 2008) used rules similar to previous findings for designing the CRF features in addition to gazetteer lists. In 2013, (Das and Garain, 2014) also used gazetteer lists and linguistic features for classification of the text using a CRF model. With the advent of Deep Learning, in 2016 (Athavale et al., 2016) proposed a technique to identify named entities in a given piece of text without any language specific rules. They used a bidirectional Long Short Term Memory (BiLSTM) model that classified the words in the sentences into one of the required classes. Their model significantly outperformed previous approaches involving rule based systems or handcrafted features. In 2018, (Xie et al., 2018) proposed a method that translated the low-resource language to a resource rich language and then using the tools available for the rich language to perform a NER on the translated text. This approach makes use of the fact that there is a good language translation from the low-resource to rich resource language, which often is not the case. A model that combined deep learning architecture with knowledge-based feature extractors was explored in (Dadas, 2018). They use a vectorized representation of the word, which is constructed by concatenating (a) the output of a pre-trained word embedding, (b) a train-able character level encoder and (c) a set of one-hot vectors from the feature extraction module. Then a hidden word representation is computed by a number of BiLSTM layers. Finally, this representation is sent to a CRF output layer, which is responsible for predicting a sequence of labels for all the words in the sentence. We use the (Athavale et al., 2016) BiLSTM architecture as our initial model, which we also use as our base model to compare our proposed NER models. We propose a method to use autoencoders and conditioning LSTM on top of this model to improve the performance of the NER task for Hindi language. Word embeddings are representation of the words in any NLP task. We describe the specific word embeddings that we used in our experiments.

2.2 Word Embeddings.

Word Embeddings have proven to be efficient representation of words in several NLP tasks. Word2vec, a type of word embedding, takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the vector space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the embedded vector space (Mikolov et al., 2013). In our experiments, we choose two different pre-trained word embeddings (Fasttext and M-BERT) to represent the words in Hindi language for the NER task. Fasttext, proposed by Facebook (Bojanowski et al., 2016) provides pre-trained word embeddings of dimension 300 for Hindi (and many other languages) built on the skip-gram model, where each word is represented as a bag of character n-grams. They used Hindi Wikipedia dumps as the corpus for training the language model. They showed that the Fasttext model out performed other baseline models that did not take into account sub-word information, as well as methods relying on morphological analysis. Later on, they released pre-trained word embeddings for 157 languages including Hindi (Grave et al., 2018). Multilingual BERT (M-BERT) was proposed by Google (Pires et al., 2019) which has an em-bedding dimension of length 768 dimension, trained on a sub-word level using the Wikipedia corpus on multiple languages. Their model architecture consisted of a bidirectional transformer trained for the task of language modeling. They detail a novel technique they used which was called Masked LM (MLM) that enhanced the performance of their word embedding model for representation of the words as well as its use in other NLP tasks.

Chapter 3
System Analysis

Detailed analysis of NER System for Weather Data Recognition is performed. We specify the functional and non-functional (implicit and explicit) requirements of the system. We show the interaction of the system and the user is illustrated using use case diagrams.

3.1 Functional Requirements

Home Page:

1. User gives the Weather Data written in Hindi to the system as input.
2. On clicking submit, the user is redirected to the Results obtained by using the trained NER model.

Results:

Weather-type, location, date from the text are identified and displayed to the user in the tagged form.

3.2 Non Functional Requirements:

3.2.1 Performance Requirements

The project is web based and must run from a localhost as it is not deployed on any web services .

3.2.2 Software Quality Attributes

Users of this system are not expected to have a technical background, hence the system is easy to use.

3.3 Specific Requirements:

Hardware :

A simple computer(desktop/laptop) which can run a web browser and a python backend is sufficient. The project is software based and has no special hardware requirements.

Software :

The application must support major web browsers with convenient access to the backend.

1. Windows 7 or higher.
2. Web Browser: Google Chrome 1.0, Internet Explorer 4.0, Firefox 1.0, Safari 1.0
3. +Python
4. Flask
5. Text Editor

3.4 Use-Case Diagrams and description

In this section, the different use case scenarios in which the users interact with the system and each other are discussed.

Scenario 1: NER Text:

In the below Fig.3.1, the user enters text in Hindi format (UTF-8) into the text area provided. On clicking view results, the system processes the text and assigns tags to the words using spacy framework and the Fasttext embeddings for hindi. After this the named entities are recognized using their tags and results are displayed to the User.

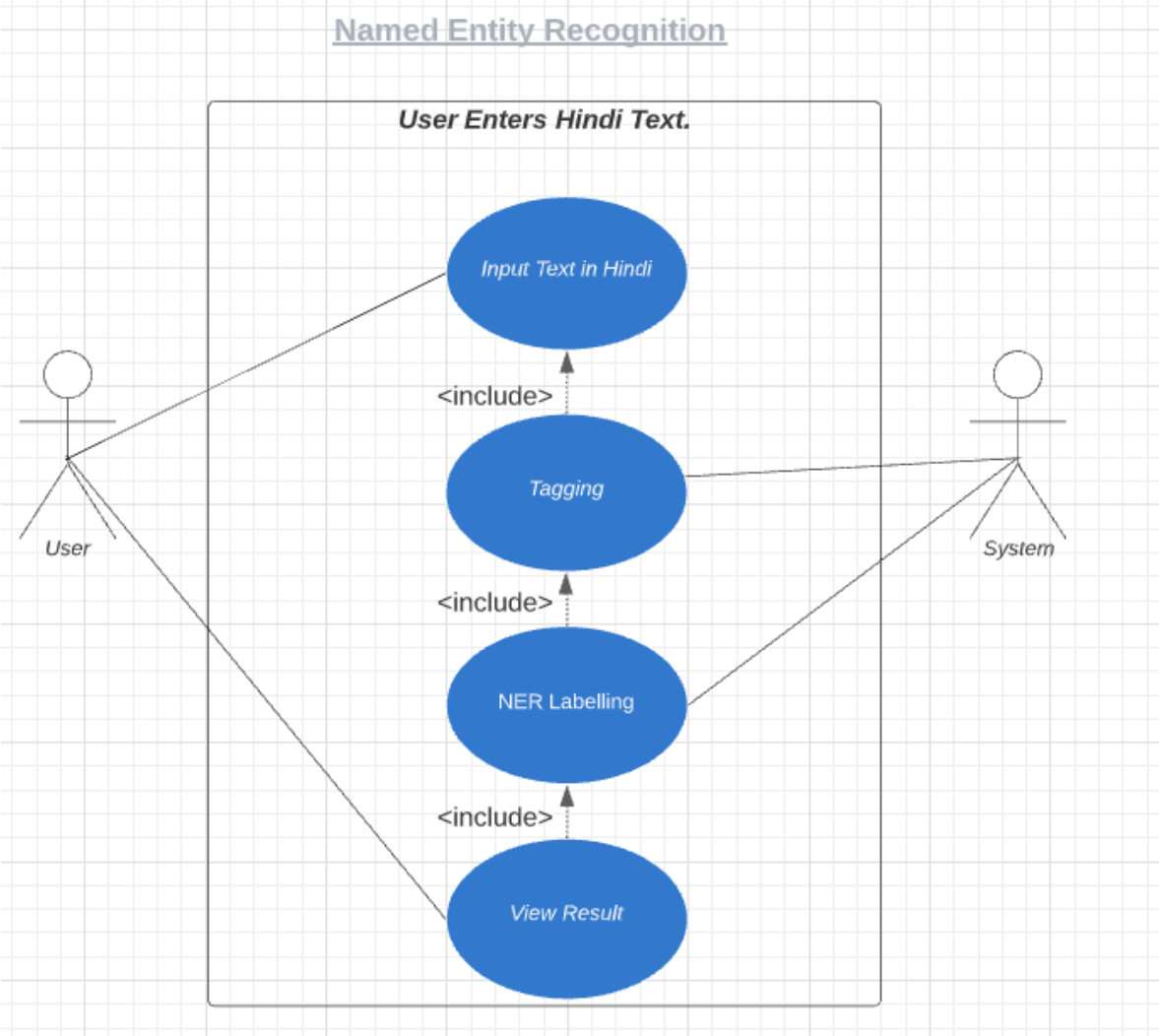


Fig. 3.4.1: Use Case Diagram for NER System in Hindi

Use Case: Input text

Brief Description: The system will accept text from the user.

Primary Actor: User

Main Flow: User provides input to the user.

Use Case: Tagging

Brief Description: System performs tagging using spacy where it is encoded with the help of CNN

Primary Actor: System

Main Flow: After the processing is done by the system, it will perform tagging using CNN algorithm.

Use Case: NER labelling

Brief Description: System performs NER labelling

Primary Actor: System

Main Flow: After the tagging is done by the system, it will perform NER labelling using spacy trained model.

Use Case: Gets output

Brief Description: The user can view the output on the screen

Primary Actor: User

Main Flow:

1.After the NER labelling is done by the system, it will generate an output according to the user requirements.
2.This output generated can be viewed by the user on the window

Chapter 4

Analysis Modeling

Analysis modeling deals with modeling and representation of data. In this chapter, we design different models in which information, functions and the behaviour of the system is defined and these are translated into the architecture.

4.1 Class Diagram and Activity Diagram

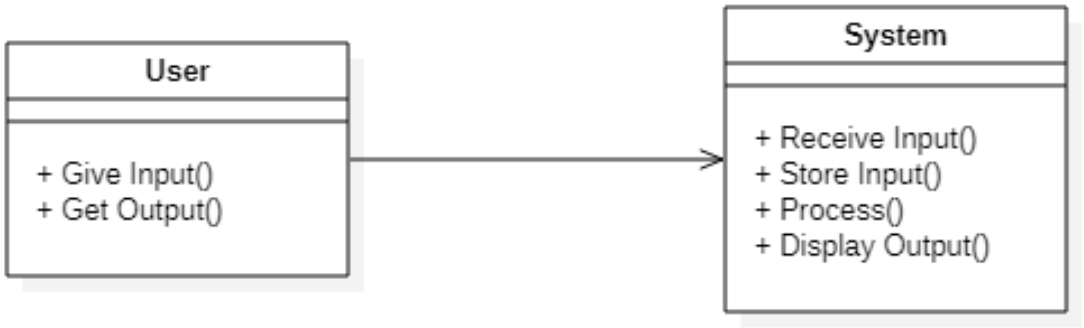
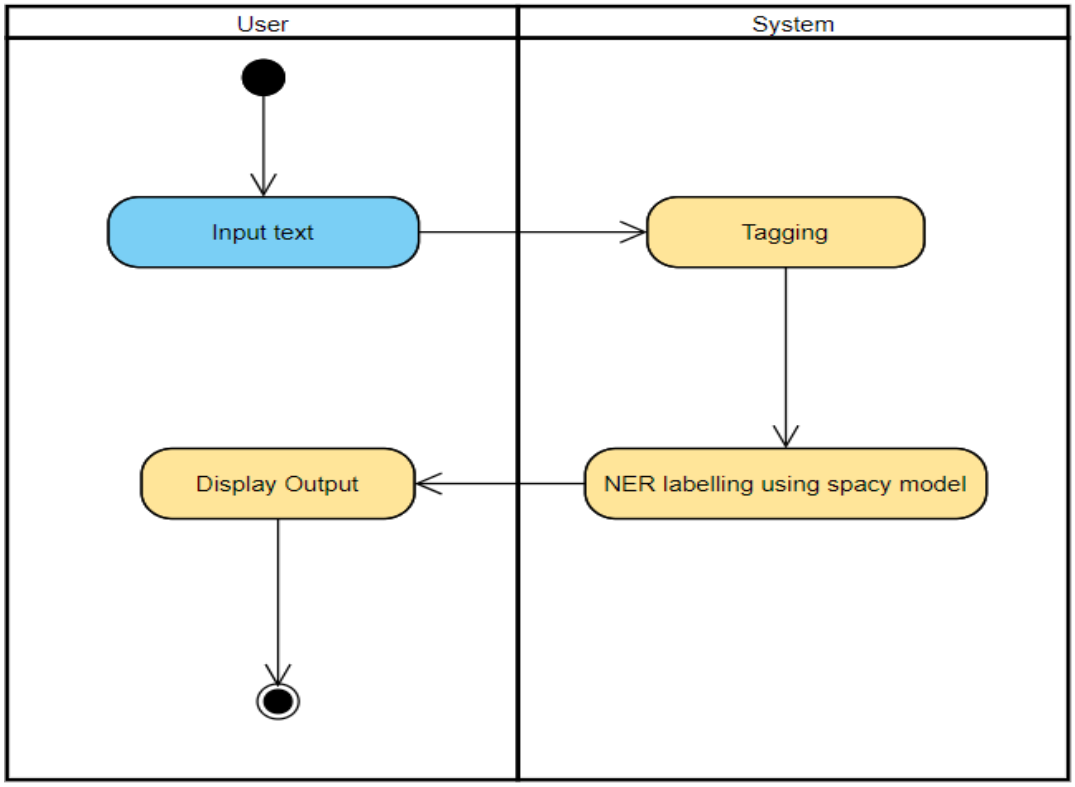


Fig 4.1.1: Class Diagram for NER System in Hindi

In this class diagram, NER labelling user input is performed and this is shown to the user.



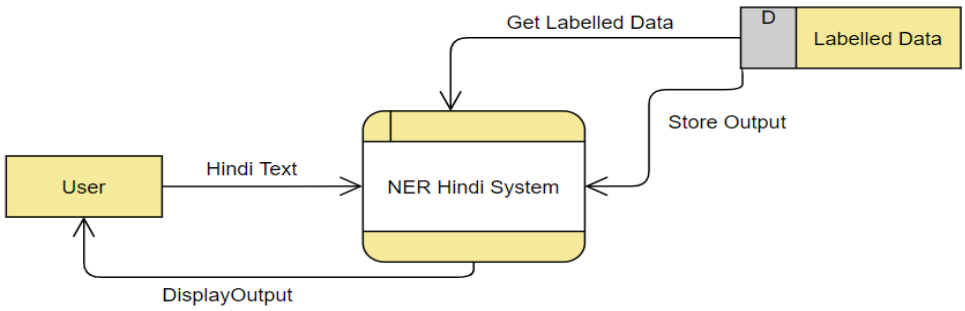
4.1.2 Activity Diagram for NER System in Hindi

User inputs data which is tagged. NER labelling is done based on the features and output is displayed to the user.

4.2 Functional Modelling

Data Flow Diagram

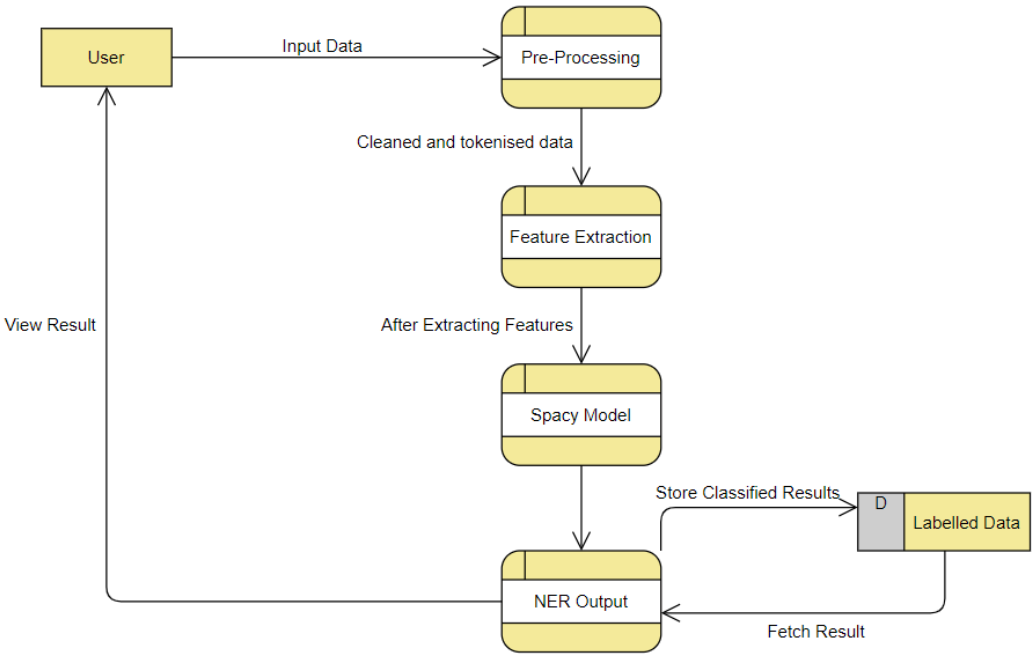
A data flow diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on flow of information, where data comes from, where it goes and how it gets stored.



DFD (level 0)

Fig.4.2.1 DFD (level -0)

The Hindi NER system takes input from the user and gives output labelled based on the training from labelled in data store Labelled data.



DFD (Level 1)

Fig 4.2.2 DFD (level-1)

User input data is cleaned, tokenized, POS tagged and given for feature extraction. Features extracted are fed to spacy model trained from labelled data and NER output is given which is viewed by the user

Chapter 5

Design

5.1 NER Workflow.

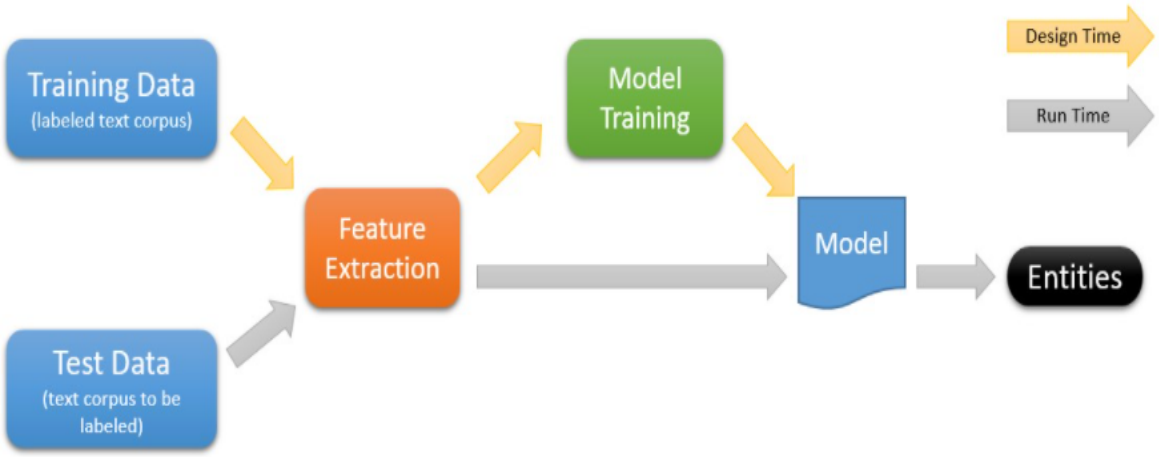


Fig. 5.1.1: NER Workflow

In the diagram above, the system is explained.

- The first thing to do is to find out a suitable dataset and their NER labels.
- After getting a suitable dataset, the next step is Data Preprocessing. In this, words are cleaned and tokenized.
- Features extracted from the tagged corpus.
- Corpus divided into train and test set
- After all the initial steps of data preprocessing is done, the Spacy model is trained using training data
- The model is tested on test data.
- Results evaluated based on Precision, Recall and F1 Score.

5.1.1 NER tagged Dataset

Suitable dataset with headline and content is created.

5.1.2 Data Preprocessing

The tagged text corpus is tokenized into words (symbols) and tags (states / classes). Each tag is assigned with a class label.

5.1.3 Feature Extraction

From the separated words (X) and tags (Y) in the preprocessing stage, extract the feature set.
The features are:

- Current Word: -POS tag of the word, If the word is a digit and the word itself.
- Previous two words:- Word, POS tag of the words
- Following two words:- Word, POS tag of the words

5.1.4 Spacy training and testing

Corpus separated into training and testing sets.
Training set used to train the Spacy model.
The trained model is used to label the test set and the result is evaluated.

5.1.5 Result and Evaluation

The label of the Corpus is compared to the label predicted by the Spacy model. The performance of the model is evaluated based on Precision, Recall and F1 Score.

Precision = No. of correct answers / Total No. of answers – produced

Recall = No. of correct answer / Total No. of possible-correct answers

F1-Measure = 2PR / (P + R)

5.2 User Interface Design

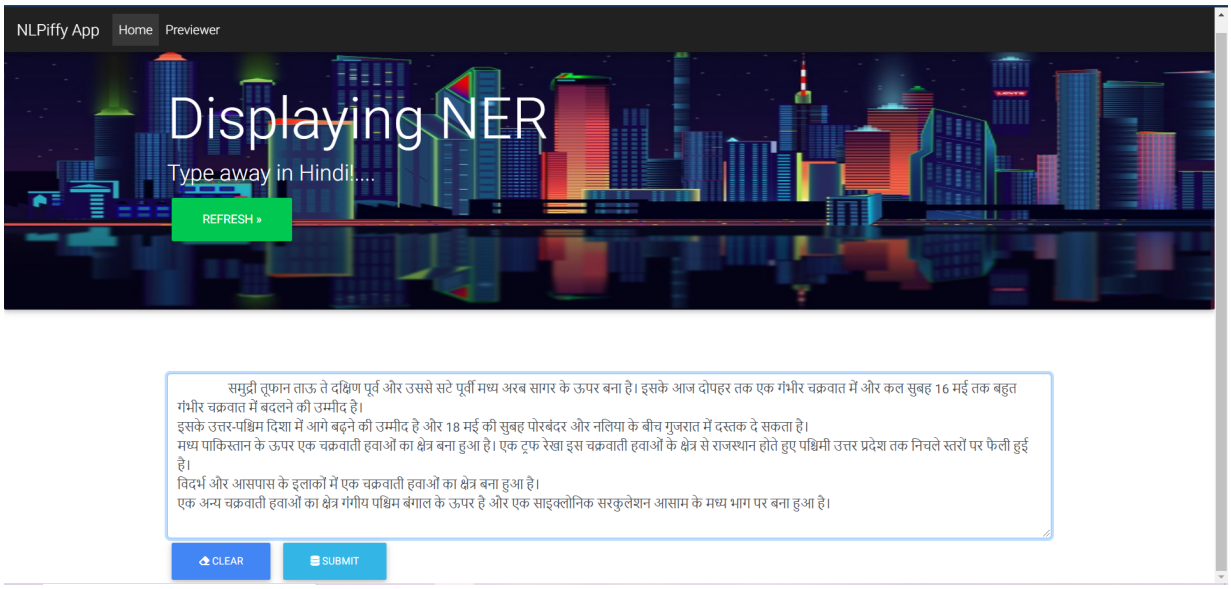


Fig 5.2.1:Accepting Input Text

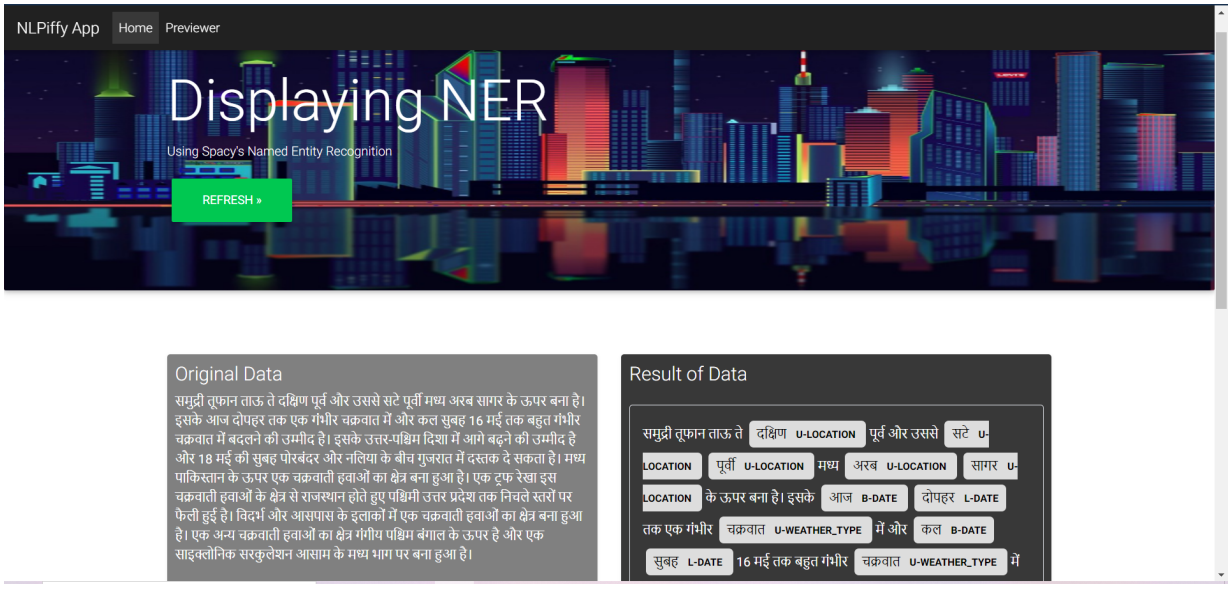


Fig 5.2.2: NER output

Chapter 6

Implementation

Hindi Named Entity Recognition using Conditional Random Fields for Custom Made Weather Text used to label user input. We use Spacy framework classification.

6.1 Working of the project

6.1.1 Dataset format to Spacy

```
import pickle

def savepkl(data,filename):
    output = open(filename + '.pkl', 'wb')

    # Pickle dictionary using protocol 0.
    pickle.dump(data, output)



def conll2spacy(data):
    sentList = []
    temp = []
    startloc = 0
    endloc = 0
    entity = []

    for word in data:
        if word == "":# new sent
            sent = " ".join(temp)
            e = {'entities':entity }
            finalSent = (sent,e)
            sentList.append(finalSent)
            temp = []
            entity = []
            startloc = 0
            endloc = 0

        else: # old sent
            w = word.split(" ")
            wordlen = len(w[0])
            if w[1] == 'O': # if normal increase both loc
```

```
            startloc += wordlen
            endloc += wordlen
        else:# if not normal then start and end diff
            endloc += wordlen
            tup = (startloc,endloc,w[1])
            entity.append(tup)
            startloc += wordlen
        temp.append(w[0])
        # including space
        startloc += 1
        endloc += 1
    return sentList
```

6.1.2 Wiki word vectors

```
lang = "hi"
vectors_loc = "/content/drive/MyDrive/mini-dataset/Data/fasttextwiki/wiki.hi.vec"
nlp = spacy.blank(lang)
with open(vectors_loc, "rb") as file_:
    header = file_.readline()
    nr_row, nr_dim = header.split()
    nlp.vocab.reset_vectors(width=int(nr_dim))
    for line in file_:
        line = line.rstrip().decode("utf8")
        pieces = line.rsplit(" ", int(nr_dim))
        word = pieces[0]
        vector = numpy.asarray([float(v) for v in pieces[1:]], dtype="f")
        nlp.vocab.set_vector(word, vector)  # add the vectors to the vocab
```

```
text = "भारी बारिश के कारण आज कार्यालय बंद रहेगा"
doc = nlp(text)
print("similarity btw",doc[0] , "and", doc[3]," :-", doc[0].similarity(doc[3]))
```

```
similarity btw भारी and कारण  :- 0.41513643
```

6.1.3 Spacy Model Training

```
def train_spacy(TRAIN_DATA,TEST_DATA,iterations,droprate = 0.5,modelName =
"modelTrained"):

    # loading hindi model and using vector from fasttext
    lang = "hi"
    vectors_loc = "/content/drive/MyDrive/mini-dataset/Data/fasttextwiki/wiki.hi.vec"
    modiner = spacy.blank(lang)
    with open(vectors_loc, "rb") as file_:
        header = file_.readline()
        nr_row, nr_dim = header.split()
        modiner.vocab.reset_vectors(width=int(nr_dim))
```

```
        for line in file_:
            line = line.rstrip().decode("utf8")
            pieces = line.rsplit(" ", int(nr_dim))
            word = pieces[0]
            vector = numpy.asarray([float(v) for v in pieces[1:]], dtype="f")
            modiner.vocab.set_vector(word, vector)  # add the vectors to the vocab

#     modiner = spacy.blank('en')  # create blank Language class

    # create the built-in pipeline components and add them to the pipeline
    # nlp.create_pipe works for built-ins that are registered with spaCy
    if 'ner' not in modiner.pipe_names:
        ner = modiner.create_pipe('ner')
        modiner.add_pipe(ner, last=True)

    # setting up f1score
    f1score = 0.0000


    # add labels that will be involved in training
    for _, annotations in TRAIN_DATA:
         for ent in annotations.get('entities'):
            ner.add_label(ent[2])


    # get names of other pipes to disable them during training
    other_pipes = [pipe for pipe in modiner.pipe_names if pipe != 'ner']
    with modiner.disable_pipes(*other_pipes):  # only train NER
        optimizer = modiner.begin_training()

        # --Iterations Starts--
        for itn in range(iterations):
            print("Starting iteration " + str(itn))
            #--Shuffling Traning Data--
            random.shuffle(TRAIN_DATA)
            losses = {}



            # batch Traning For better Training and Learning of model
            batches = minibatch(TRAIN_DATA, size=compounding(2.0, 16.0, 1.01))
            for batch in batches:
                texts, annotations = zip(*batch)
                modiner.update(
                    texts,  # batch of texts
                    annotations,  # batch of annotations
                    drop=droprate,  # dropout - make it harder to memorise data
                    losses=losses,
                )
```

```
        print(losses)


        # Evaluating the Current Model Score on test data
        results = evaluate(modiner, TEST_DATA)
        print("Current Score :-",results["ents_f"], "Precision  :-",results["ents_p"], "Recall
:-",results["ents_r"])



        # loading previous best saved model in start of traning
        if f1score == 0.00:
            try:
                pnlp = spacy.load(modelName)
                result = evaluate(pnlp, TEST_DATA) # calling evaluate function
                f1score = result["ents_f"]
            except:
                print("Previous Model not found")

        print("Best Sccore :- ",f1score)
        print("------------------------------------")
        # finding out the best score
        if f1score < results["ents_f"]:
            f1score = results["ents_f"]

            # Save our trained Model if the score if grater than best score else no change in
previous model
            modiner.to_disk(modelName)

    print("-----Best Model is Saved-----")
```

6.1.4 Evaluating

```
def evaluate(ner_model, examples):
    scorer = Scorer()

    #loading tags for each input and Evaluating them
    for input_, annotations in examples:
        tags = []
        # loading text
        doc_gold_text = ner_model.make_doc(input_)

        #loading all tags for that text
        for ent in annotations.get('entities'):
            tags.append(ent)

        # Evaluating the tags
        gold = GoldParse(doc_gold_text, entities=tags)
```

```
        pred_value = ner_model(input_)
        scorer.score(pred_value, gold)


    return scorer.scores
```

6.2 Result Analysis

```
def score(model,TEST_DATA):
    result = evaluate(model, TEST_DATA) # calling evaluate
function
    f1score = result["ents_f"]
    precision = result["ents_p"]
    recall = result["ents_r"]
    print("F1 score of Model is :-",f1score)
    print("Precision of Model is :-",precision)
    print("Recall of Model is :-",recall)

# loading the saved model
pnlp = loadNERModel("hindiNER")

# calculating the score of the model
score(pnlp,TEST_DATA)
```

```
F1 score of Model is :- 92.0
Precision of Model is :- 92.0
Recall of Model is :- 90.0
```

Chapter 7
Conclusion

The main requirement of NER labelling is tagging and feature extraction. Extracting only the relevant features is important for the accuracy of the model. NER model is obtained using Spacy Customized training model. The system is trained on weather data and is thus most suitable for labelling weather input. The NER system developed can be a part of various applications.

The system currently predicts labels for individual words and not phrases. The system has a lot of scope for further study with Noun phrases. The model can be customized for use in different fields using suitable datasets for the same.

References

1. Vinayak Athavale, Shreenivas Bharadwaj, Monik Pamecha, Ameya Prabhu, and Manish Shrivastava. 2016. Towards deep learning in hindi NER: an approach to tackle the labelled data sparsity. CoRR, abs/1610.09756.

2. Peng Sun, Xuezhen Yang, Xiaobing Zhao and Zhijuan Wang, '*An Overview of Named Entity Recognition*', 2018 International Conference on Asian Language Processing (IALP), 2018

3. Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. CoRR, abs/1607.04606.

4. Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? CoRR, abs/1906.01502.

Acknowledgements

Our group would like to express our sincere thanks to **Director Bro. Jose Thuruthiyil**, our Principal **Dr. Sincy George**and the HOD of our department, **Dr. KavitaSonawane**for the precious platform to evolve and grow as a student.

We are highly indebted to our Teacher, Guide and Mentor **Mrs. Vincy Joseph** for her constant guidance, knowledge, constructive feedback and motivation which have helped us achieve our objective for the report topic. We are thankful to the library staff of St. Francis Institute of Technology for their guidance and help to get the right information from the wide range of resources available.

We also want to thank our family members and friends for their patience and support throughout the project. We are also thankful to all the contributors of online forums, bloggers and YouTubers whose work and insights have helped us understand the perspectives of people around the globe.