



# Project Overview

As Mentioned in Previous Slide, I had to learn CRF, Feature
Functions and had to make a Pipeline from Chunking, POS to NER
all through CRF. I also had to do Annotations. I had to test on two
different data and check their results.

#### Contents

## Table Of Contents

01 How I proceeded?

Problems, Solutions and Learnings.

03 About Models Implementation

What Models I have made for Pipeline, and How they Works?

**02** About CRF and NER

What Different processes involved in their implementation

04 Results

How they Performed?



## How I Proceeded?

My Goal was to test use CRF Feature Functions for English which includes many Lexical properties for NER, such as Capitalization, Position in a sentence, Articles etc. with a rich premade-gazeteer.

But after Presentation, My Topic changed to Hindi, so I researched about CRF, Feature Functions, How they work, Formulaes, Linear CRF and How to Implement on Hindi.

# Summarise Learnings

- CRF is Conditional Random Field model, which works on its Template and Its Feature Functions, whereas Template provide information about which parts are dependent on which. Function provide probability the of current node in given nodes on which it is dependent.
- Functions are of form f(X,Yi-1,Yi,i), which have arguments current\_val,prev\_state,current\_state,weight(acc. to length).
- P(Y/X)=1/Z \* e^Sigma(F(X,Yi-1,Yi,i)), where F(X,Yi-1,Yi,i)= Sigma(wj\*fj)
- w is calculated while training the model.
- The Above Formulaes are for Linear CRF++, in which Current State is only dependent on Previous adjacent Stage.

## Problems

My Goal was to test use CRF Feature Functions for English which includes many Lexical properties for NER, such as Capitalization, Position in a sentence, Articles etc. with a rich premade-gazeteer, (Yup Ctrl+C, V). Whereas Hindi had no capitalization, It is Morphologically Rich, Having hard to find good gazeteer, No Particular Word Order, or Articles.

SO I DROPEED USING CRF FEATURE FUNCTIONS FOR NER



#### **NER**

Named Entity Recognition (NER) is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into predefined categories such as the names of persons, organizations, locations.

#### CRF

Conditional Random Fields (CRF) is a type of probabilistic graphical model used for structured prediction, especially in sequence labeling tasks such as named entity recognition (NER), part-of-speech (POS) tagging, and chunking.

# Process

### Chapter 2

1

#### **NER**

Got NER Data from Hugging Face and Trained Model 2

#### **POS**

Collected the
Data and
trained POS
Model. Also
anotated some
POS test.

3

#### **PIPELINE**

Connected
Pipeline so, If
Data is
provided, it will
return POS, NER
Annotated Data

4

#### **TOKENIZER**

To make It complete Text based, one System.



### NAME ENTITY RECOGNITION

#### STEPS:

IMPLEMENTED USING CRF++
COLLECTED TRAINING DATA WITH 150K LINES OF ANNOTATION

TRAINED MODEL ON PROVIDED DATA

TESTED ON PROVIDED COLLECTED TEST
TESTED ON NEWS DATA (MANUALLY)
TESTED ON STORY DATA (MANUALLY)

MADE PIPELINE TO PROVIDE TEXT->NER



### PARTS OF SPEECH

#### STEPS:

IMPLEMENTED USING CRF++
COLLECTED TRAINING DATA WITH 100K LINES OF ANNOTATION

TRAINED MODEL ON PROVIDED DATA

TESTED ON ANNOTATED TEST (MANUALLY)
TESTED ON NEWS DATA (MANUALLY)
TESTED ON STORY DATA (MANUALLY)

MADE PIPELINE TO PROVIDE TEXT->NER



### PIPELINE

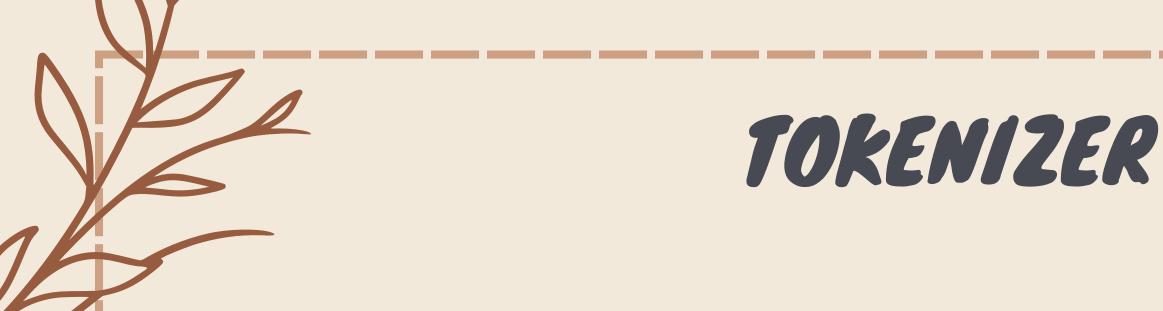
#### STEPS:

WROTE JUPYTER SCRIPT TO TRAIN MODEL WITH JUST PROVIDING ANNOTATED DATA.

WROTE JUPYTER SCRIPT TO ANNOTATE DATA WITH JUST PROVIDING TEXT

IF DATA IS ALREADY ANNOTATED, IT TESTS THE PRECISION, RECALL, F1 SCORE OF MODEL ON ANNOTATED DATA

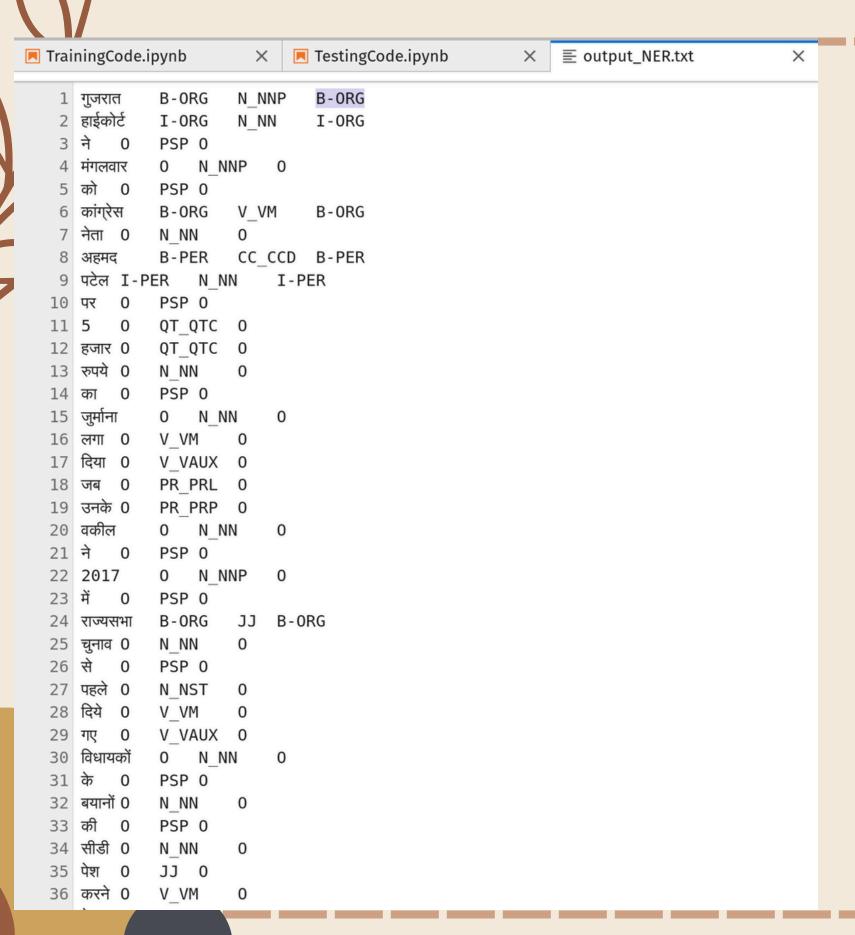
(PIPELINE CAN BE EXTRAPOLATED TO CHUNKING TOO, IF ANNOTATED DATA IS PROVIDED)



#### STEPS:

WROTE CODE TO CONVERT TEXT INTO SENTENCES
WROTE CODE TO CONVERT SENTENCES INTO WORD IN CONLL FORMAT FOR
TESTING/ANNOTATING DATA

(I HAVENT USED REGEX, IT WAS SIMPLE CODE, USING REGEX I CAN REMOVE PUNCTUATIONS OR ONE CAN DO AFTER TOKENIZATION, WITH JUST CTRL+F (ALT+ENTER)) \*\*Applied to remove Punctuation



### Given Test Data

NER: F1 Score:0.92 Precision:0.85

Matching count: 2874 Total entities: 3380

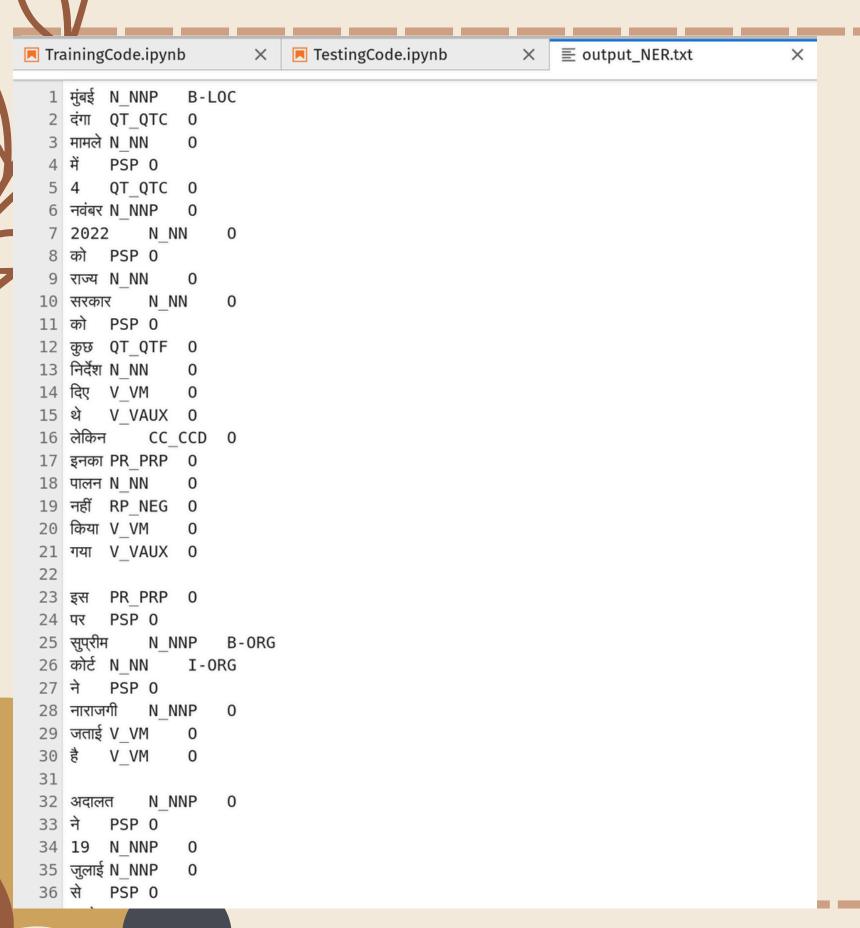
Precision: 0.8502958579881656

Recall: 1

F1 Score: 0.9190917812599936

I wonder O are so many so should we consider them in precision or not \*\*Fixed by just counting non-O

POS: F1 Score:0.93 Precision:0.88

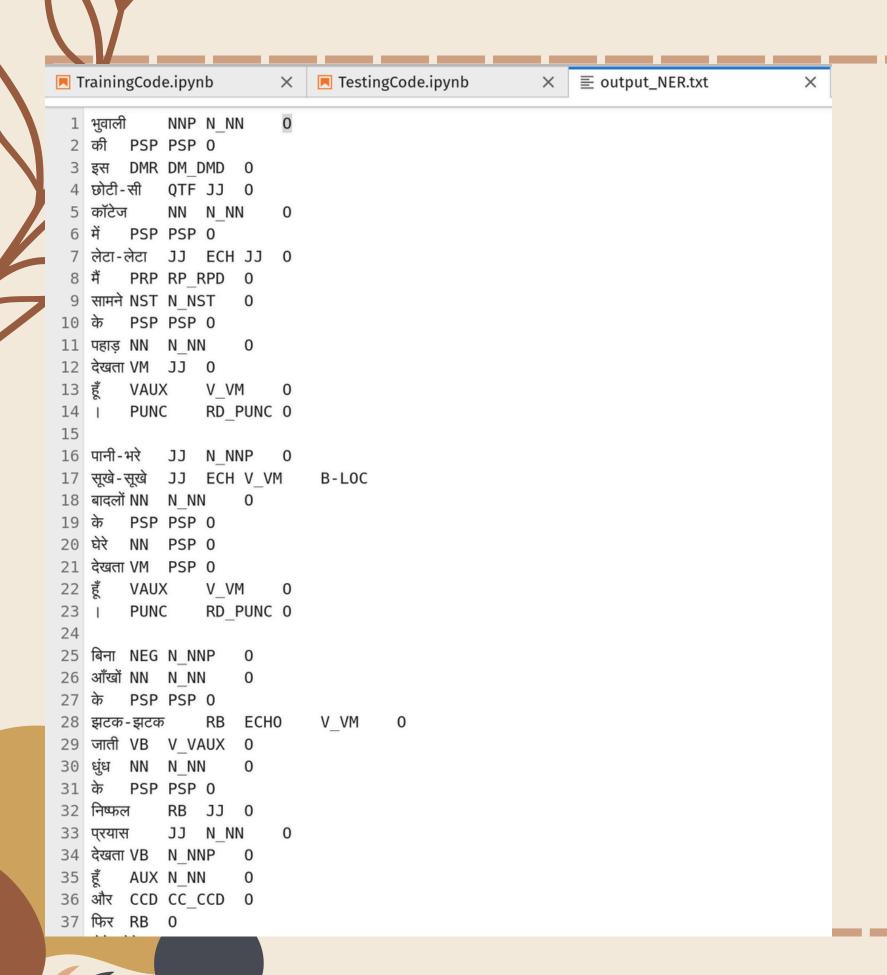


### News Test Data

NER: F1 Score:0.90 Precision:0.82

This One Feels
True having true
Precision

POS: F1 Score:0.85 Precision:0.91



## Story Test Data

NER: F1 Score:0.34 Precision:0.21 But the truth is there are only o tags, and most NER tags are wrong or missed. so Precision is around 0.1 \*\*Fixed

POS: F1 Score:0.80 Precision:0.67

Number(Correct_Tags)	352
Number(Wrong_Tags)	157
Precision	0.6692307692
Recall	1
F1 Score	0.801843318

# What else?

With Chunking Training Data, One can easily ass Chunk Annotator in Pipeline.

With Regex One make Tokenizer better\*\* Done a bit

BIO Tagging is with NER Tagging so no need to do another time

With Some more Time and Effort,
One can make complete Pipeline which takes text and annotate it

The Problem is Precision, It depends on whether Training Data include such type or not can be fixed by increasing amount of Training Data



