

Computational Linguistics Project Report

Ayush Kumar Gupta

May 2024

1 Introduction

This report presents a comprehensive approach to linguistic analysis, encompassing Named Entity Recognition (NER), Part-of-Speech (POS) tagging, and tokenization. Leveraging CRF++ for NER and POS tagging, along with a custom Python tokenizer using Regex, we convert raw text into CoNLL format for efficient processing. The pipeline model facilitates seamless training and testing, while rigorous evaluation metrics like precision, recall, and F1 score ensure the accuracy and robustness of our analysis. Through testing on diverse datasets, we validate the versatility of our approach, aiming to advance the field of natural language processing with practical, impactful solutions.

1.1 Name Entity Recognition

Named Entity Recognition (NER) is a crucial task in natural language processing (NLP) that involves identifying and categorizing named entities within unstructured text data. These entities can include names of people, organizations, locations, dates, numerical expressions, and more. NER plays a vital role in various NLP applications such as information retrieval, question answering, and sentiment analysis.

1.2 Parts of Speech Tagging

Part-of-Speech (POS) tagging is a fundamental task in natural language processing (NLP) that involves assigning grammatical categories (such as noun, verb, adjective) to each word in a sentence. Applications include machine translation, text summarization, and sentiment analysis.

1.3 Conditional Random Field (CRF++)

CRF++ is a widely used toolkit for training and applying Conditional Random Fields (CRF) models, particularly in sequence labeling tasks such as Named Entity Recognition (NER) and Part-of-Speech (POS) tagging. It offers efficient training algorithms and supports various features, allowing for robust and accurate modeling of sequential data.

2 Working of CRF++

Templates in CRF provide essential information about the dependencies between different parts of the sequence. They define the structure of the model and specify which parts of the sequence are dependent on each other. For example, in a POS tagging task, a template might indicate that the current POS tag depends on the previous and next POS tags.

Feature functions, on the other hand, compute the probability of a particular state given the observed input and the previous and current states. These functions take into account various features of the input sequence, such as word identities, word shapes, and context information.

The probability of a sequence Y given an input sequence X is calculated using the following formula:

$$P(Y|X) = \frac{1}{Z} \exp \left(\sum_i \sum_j w_j \cdot f_j(X, Y_{i-1}, Y_i, i) \right)$$

Here, Z is the normalization factor ensuring that the probabilities sum up to 1, w_j are the weights associated with each feature function f_j , and i ranges over the positions in the sequence.

Each feature function $f_j(X, Y_{i-1}, Y_i, i)$ calculates a score based on the input sequence X and the current and previous states Y_{i-1} and Y_i . These scores are summed over all feature functions and weighted by their corresponding weights w_j to compute the final probability of the sequence.

In a linear CRF, the current state depends only on the previous adjacent state, leading to a simpler and more computationally efficient model. This formulation allows CRF to effectively capture the dependencies between adjacent elements in a sequence, making it well-suited for sequence labeling tasks in NLP.

3 Problems with Feature Functions for Hindi

The application of CRF feature functions for tasks like Named Entity Recognition (NER) and Part-of-Speech (POS) tagging poses significant challenges when dealing with languages like Hindi. Unlike English, Hindi lacks certain linguistic features such as capitalization and definite articles, making it difficult to leverage these properties as feature functions in CRF models.

One major obstacle is the absence of capitalization in Hindi. In English, capitalization patterns often provide valuable cues for identifying proper nouns, which are essential for NER tasks. However, in Hindi, where capitalization is not employed in the same way, this feature becomes irrelevant and cannot be effectively utilized as a feature function.

Furthermore, the morphological richness of Hindi presents additional complexities. Hindi words undergo various morphological changes based on factors such as tense, gender, and case. This morphological diversity adds a layer of

complexity that is not easily captured by simple feature functions in CRF models.

Moreover, the lack of a standardized word order in Hindi further complicates the task of designing effective feature functions. Unlike English, which generally follows a subject-verb-object order, Hindi sentences can have a more flexible word order, making it challenging to rely on positional information as a reliable feature.

Additionally, the absence of definite articles in Hindi poses a challenge for POS tagging. In English, definite and indefinite articles play a crucial role in determining the grammatical function of words, aiding in POS tagging. However, Hindi does not have explicit articles like "the" or "a," making it challenging to utilize article-related features in CRF models effectively.

Lastly, the scarcity of comprehensive gazetteers for Hindi further hinders the feasibility of using CRF feature functions effectively. Unlike English, where rich premade gazetteers are readily available, compiling a comprehensive gazetteer for Hindi requires extensive effort and resources, making it impractical for many NLP applications.

In conclusion, the inherent linguistic differences between English and Hindi, such as the absence of capitalization, morphological richness, flexible word order, and lack of definite articles, make it challenging to utilize CRF feature functions effectively for tasks like NER and POS tagging in Hindi. As a result, When I had to do the Tasks on Hindi, I preferred CRF with No Feature Functions.

4 Models and Pipeline

4.1 Named Entity Recognition Model

Leveraging data obtained from Hugging Face, a model is meticulously trained using CRF++ methodology. The training dataset comprises 150,000 annotated lines (which was around 0.75%). Rigorous testing ensues, encompassing both annotated pre-made test data and manually curated news and story datasets.

4.2 Part-of-Speech Tagging Model

Employing CRF++ methodology, a model is crafted and fine-tuned using a dataset consisting of 100,000 annotated lines. Testing procedures mirror those of NER, encompassing annotated test data as well as manually curated news and story datasets. The development of a streamlined pipeline is a cornerstone, facilitating the provision of text input and retrieval of POS annotated data.

4.3 Pipeline Development

We delve into development of a cohesive pipeline, seamlessly weaving together the intricate workings of Named Entity Recognition (NER) and Part-of-Speech (POS) tagging models. This streamlined Training and Annotation Process using

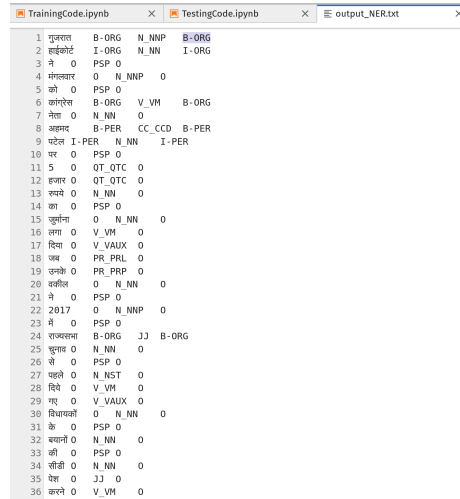
Jupyter scripts. What sets this pipeline apart is its adaptability; whether handling annotated or unannotated data. Moreover, the pipeline provides precision, recall, and F1 score metrics on annotated datasets, and also tags non-annotated datasets (but then provides wrong metrics as had to do on our own, manually). Pipeline can be fully customised by changing inputs for Training and Testing the data.

4.4 Tokenizer Implementation

An essential component in the preprocessing pipeline, the tokenizer is designed to convert raw text into structured sentences and subsequently into word tokens in CoNLL format. Used Regex to remove the Punctuation. With Tokenization, We had complete Pipeline where user can enter Text which will be POS Tagged and then NER Tagged and stored in output CoNLL format.

5 Results

5.1 NER Test Data (From HuggingFace Annotated)



Line	Word	POS	NER
1	गुरुलाल	B-ORG	N_NNP
2	हॉइमोट	I-ORG	N_NNP
3	३	PSP	O
4	संरक्षार	O	N_NNP
5	को	PSP	O
6	कॉन्फेस	B-ORG	V_VM
7	नेता	O	N_NNP
8	अपार	B-PER	CC_CCD
9	रंजन	I-PER	N_NNP
10	र	PSP	O
11	5	OT_OTC	O
12	हमर	OT_OTC	O
13	समे	N_NNP	O
14	जा	PSP	O
15	कुर्मा	O	N_NNP
16	ला	V_VM	O
17	रिवा	V_VAUX	O
18	अ	PR_PRL	O
19	उम्मे	PR_PRL	O
20	दकील	O	N_NNP
21	ने	PSP	O
22	2017	O	N_NNP
23	३	PSP	O
24	राज्यपाल	B-ORG	JJ
25	भुवा	O	N_NNP
26	ने	PSP	O
27	वहने	N_NST	O
28	रिने	V_VM	O
29	ग	V_VAUX	O
30	विद्यार्थी	O	N_NNP
31	के	PSP	O
32	बगनी	O	N_NNP
33	सी	PSP	O
34	सीडी	O	N_NNP
35	पे	JJ	O
36	करने	V_VM	O

Figure 1: NER Test Data Tagged Sample

For POS Tagging (Manually Calculated),

Precision: 0.88

Recall: 1

F1 Score: 0.93

Matching count: 2874
 Total entities: 3380
 Precision: 0.8502958579881656
 Recall: 1
 F1 Score: 0.9190917812599936

Figure 2: NER Test Data NER Metrics *Updated

Line	Text	NER Tag	Score
1	भुलानी	NNP N_NN	0
2	पुल	PSP PSP	0
3	इस	DMR DM_DMD	0
4	अपनी-नी	QTP JJ	0
5	कॉन्ट्र	NN N_NN	0
6	मैं	PSP PSP	0
7	लेटा-लेटा	JJ ECH JJ	0
8	मैं	PRP RP_RPD	0
9	सामने	NST N_NST	0
10	के	PSP PSP	0
11	पहाड़	NN N_NN	0
12	वेकता	VM JJ	0
13	हैं	VAUX V_VM	0
14	पुल	PUNC RD_PUNC	0
15			
16	पुल-सुखे	JJ N_NNP	0
17	बादलों	JJ ECH V_VM	B-LOC
18	के	PSP PSP	0
19	रे	NN PSP	0
20	वेकता	VM PSP	0
21	हैं	VAUX V_VM	0
22	पुल	PUNC RD_PUNC	0
23			
24			
25	मिल	NEG N_NNP	0
26	औरी	NN N_NN	0
27	के	PSP PSP	0
28	हॉटक-हॉटक	RB ECHO V_VM	0
29	जाली	VB V_VAUX	0
30	सुख	NN N_NN	0
31	के	PSP PSP	0
32	निम्न	RB JJ	0
33	परसल	JJ N_NN	0
34	वेकता	VB N_NNP	0
35	हैं	AUX N_NN	0
36	और	CCD CC_CCD	0
37	निर	RB	0

Figure 3: Story Test Data Tagged Sample

5.2 Story Test Data

For NER Tagging (Manually Calculated),

Precision: 0.21

Recall: 1

F1 Score: 0.34

5.3 News Test Data

For NER Tagging (Calculated Manually)

Precision: 0.82

Recall: 1

F1 Score: 0.90

For POS Tagging (Calculated Manually)

Precision: 0.91

Recall: 1

F1 Score: 0.85

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

Figure 4: Story Test Data POS Metrics

Line	Text	POS Tag	BIOES Label
1	सुबह	N_NNP	B-LOC
2	रात	OT_OTC	O
3	मामले	N_NN	O
4	ने	PSP	O
5	4	OT_OTC	O
6	नंबर	N_NNP	O
7	2022	N_NN	O
8	को	PSP	O
9	राज्य	N_NN	O
10	सरकार	N_NN	O
11	को	PSP	O
12	द्वारा	OT_OTC	O
13	निर्देश	N_NN	O
14	लिए	V_VM	O
15	से	V_VAUX	O
16	लेखन	CC_CCD	O
17	इलाका	PR_PR	O
18	पारल	N_NN	O
19	नहीं	RP_NEG	O
20	किया	V_VM	O
21	गया	V_VAUX	O
22			
23	इस	PR_PR	O
24	पर	PSP	O
25	सुरक्षा	N_NNP	B-ORG
26	कोर्ट	N_NN	I-ORG
27	ने	PSP	O
28	नाराज	N_NNP	O
29	जमाना	V_VM	O
30	ने	V_VM	O
31			
32	अदालत	N_NNP	O
33	ने	PSP	O
34	19	N_NNP	O
35	जुलाई	N_NNP	O
36	से	PSP	O

Figure 5: News Test Data Annotation

5.4 Metrics

5.4.1 Precision

Precision measures the accuracy of positive predictions made by the model. It is calculated as the ratio of true positives (TP) to the sum of true positives and false positives (FP).

$$Precision = \frac{TP}{TP + FP}$$

5.4.2 Recall

Recall, also known as sensitivity or true positive rate, measures the ability of the model to capture all positive instances. It is calculated as the ratio of true positives (TP) to the sum of true positives and false negatives (FN).

$$Recall = \frac{TP}{TP + FN}$$

*****For Our Model, It always gives some tag, Therefore False Negative was 0, Therefore Recall was always 1.***

5.4.3 F1 Score

The F1 Score is the harmonic mean of Precision and Recall. It balances both Precision and Recall and provides a single metric to evaluate the model's performance. It is calculated as follows:

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

6 Conclusion

From the above mentioned results, We observed that How Results can vary depending on Type of Testing and Training data used. For NER The Training data was general having more similarity with News data so performed quite well having a Good Precision and F1 Score. Whereas Story data had complete different vocabulary and Gazeteer, so Its Testing NER gave lots of False Positives decreasing Precision and F1 Score. It can be improved by training data on Story. Same goes with POS too, as testing data was from articles and news instead of Stories or Novels.

7 Advancements

- We have Pipeline for Training and Testing data, Using Pipeline One can train many annotated data. And Test The Metrics Score for the given Annotated Data in the Same way or can Tag Non annotated Data with variable precision according to Data Types used for Training and Annotation.
- One can add Chunking in Pipeline to further completing it for all NLP Tagging usages(For Syntax and Semantics Operations).
- If enough of annotated data is provided for above task with some time, all mentioned advancement would have been implemented.

8 Github Link for Repo

https://github.com/AKGIITH/NER_CRF_Hindi
https://github.com/AKGIITH/NER_CRF_Hindi

9 Sources

NER Test and Train Data: <https://huggingface.co/ai4bharat>
POS Train Data: Hindi_Tourism_Corpora

POS Test Data:
Do Bail: Munshi Premchand
News: AajTak, Amar Ujala