Linear Chain Conditional Random Fields for Named Entity Recognition

ELL884: Deep Learning for Natural Language Processing
Assignment 2 Report

Aastha A K Verma (2022CS11607)

March 2025

1 Introduction

Conditional Random Fields (CRFs) are a probabilistic method for structured predictions. This document presents Named Entity Recognition (NER) implemented using CRFs. For more details, refer to my GitHub repository[1].

1.1 General CRFs

We wish to predict a vector $\mathbf{y} = \{y_0, y_1, ..., y_T\}$ of random variables given an observed feature vector x. In NLP-related applications like tagging, y_s is the tag of the word at position s, and the input $\mathbf{x}$ is divided into feature vectors $\{x_0, x_1, ..., x_T\}$. Each $\mathbf{x}_s$ contains various information about the word at position s, such as its identity, ortho- graphic features such as prefixes and suffixes, etc. [2]
We use CRFs to model the conditional probability $p(\mathbf{y}|\mathbf{x})$ to be able to use classification on top of the model. An advantage is that a conditional model has a much simpler structure than a joint model (i.e, $p(\mathbf{y}, \mathbf{x})$).

Undirected Graphical Models

An undirected graphical model is a family of probability distributions that factorize according to given collection of scopes.

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \prod_{a \in \mathcal{F}} \Psi_a(\mathbf{x}_a, \mathbf{y}_a)$$

$$Z = \sum_{\mathbf{x}, \mathbf{y}} \prod_{a \in \mathcal{F}} \Psi_a(\mathbf{x}_a, \mathbf{y}_a).$$

The quantity Z, considered as a function of the set $\mathcal{F}$ of factors, is called the **partition function**, which is a normalization factor to ensure that the probabilities sum to 1.

Directed Graphical Models

Whereas the local functions in an undirected model need not have a direct probabilistic interpretation, a *directed graphical model* describes how a distribution factorizes into local conditional probability distributions. Let $G = (V, E)$ be a directed acyclic graph, in which $\pi(v)$ are the parents of v in G. A directed graphical model is a family of distributions that factorize as:

$$p(\mathbf{y}, \mathbf{x}) = \prod_{v \in V} p(y_v | \mathbf{y}_{\pi(v)}). \tag{1}$$

1.2 Named Entity Recognition (NER)

Named-entity recognition (NER) is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc. [3]

[1] `https://github.com/loadthecode0/crf-ner`

IOB tagging

Short for **Inside-Outside-Beginning** tagging, it is a common NER tagging scheme. [1]

- The **I-prefix** before a tag indicates that the tag is inside a chunk.
- An **O** tag indicates that a token belongs to no chunk.
- The **B-prefix** before a tag indicates that the tag is the beginning of a chunk that immediately follows another chunk of the same type without O tags between them. It is used only in that case: when a chunk comes after an O tag, the first token of the chunk takes the I- prefix.

1.3 Linear Chain CRFs

One approach to NER is to classify each word independently as one of either Person, Location, Organization, or Other (meaning not an entity). The problem with this approach is that it assumes that given the input, all of the named-entity labels are independent. In fact, the named-entity labels of neighboring words are dependent; for example, while New York is a location, New York Times is an organization. One way to relax this independence assumption is to arrange the output variables in a linear chain. This is what we do with HMMs. Linear chain CRFs are the discriminative analogue of HMMs, which predict only the labels give the features (as opposed to learning the joint probability of the labels and the features).

2 Mathematical Formulation of Linear Chain CRFs for NER

Notation

In this report, we use the following notation:

- $\mathbf{Y} = \{y_1, y_2, ..., y_N\}$: the observation sequence of NER labels.
- $\mathbf{X} = \{x_1, x_2, ..., x_N\}$: the input sequence of tokens.
- $\mathbf{G} = \{g_1, g_2, ..., g_N\}$: the input sequence of POS tags.
- $\mathbf{T}$: the length of the sequence.
- $\mathbf{N}$: Number of NER labels.
- P = number of unique POS tags
- O = number of observation feature functions
- $f_{\text{score}}(\mathbf{X}, \mathbf{G}, \mathbf{Y})$: score of the given sequence triplet.
- $f_{\text{partition}}(\mathbf{X}, \mathbf{G})$: partition function value, i.e, sum of scores with all possible sequences $\mathbf{Y}$.
- $\mathbf{W}^o$: Observation function weights.

2.1 Sequence Scoring Function

Look at Section 4.2 for class-weighing scheme.

2.1.1 Transition score

$$T(\tilde{y}_{t-1}, \tilde{y}_t) \propto \text{normalised-class-weight}(y_t) \times \log P(y_t = \tilde{y}_t | y_{t-1} = \tilde{y}_{t-1})$$

2.1.2 Emission score

$$E(\tilde{g}, \tilde{y}_t) \propto \log P(g_t = \tilde{g}_t | y_t = \tilde{y}_t)$$

2.1.3 Observation Function

$$o_i(\tilde{\mathbf{X}}, \tilde{\mathbf{G}}, \tilde{y}_t) = w_i^o \cdot \mathbb{1}_{\{i^{th} \text{ obs. feature is true}\}}$$

So,

$$O(\tilde{\mathbf{X}}, \tilde{\mathbf{G}}, \tilde{y}_t) = \mathbf{W}^o \cdot \mathbf{F}(\tilde{\mathbf{X}}, \tilde{\mathbf{G}}, \tilde{y}_t, t)$$

where $\mathbf{F}$ is the feature vector constructed for the given timestep t.

2.1.4 Overall Sequence Score, f_{score}

For the t^{th} timestep in the sequence,

$$f_{\text{score},t}(\tilde{\mathbf{X}}, \tilde{\mathbf{G}}, \tilde{y}_t) = \lambda_T \cdot T(\tilde{y}_{t-1}, \tilde{y}_t) + \lambda_E \cdot E(\tilde{g}, \tilde{y}_t) + O(\tilde{\mathbf{X}}, \tilde{\mathbf{G}}, \tilde{y}_t)$$

I have considered different weighing factors for $T(\cdot, \cdot)$ and $E(\cdot, \cdot)$ since they can possibly contribute on different scales to the loss functions. They are hyperparameters to be tuned.
For the whole sequence,

$$f_{\text{score}}(\tilde{\mathbf{X}}, \tilde{\mathbf{G}}, \tilde{\mathbf{Y}}) = \sum_{t=1}^{T} f_{\text{score},t}(\tilde{\mathbf{X}}, \tilde{\mathbf{G}}, \tilde{y}_t)$$

Note that we add them, because the probabilities would be multiplied, but right now we are working in the log-space.

2.2 Partition Function, $f_{\text{partition}}$

For the partition function, we need the joint probability of $\mathbf{X}$ and $\mathbf{Y}$. To marginalize over the space of all possible N-length $\mathbf{Y}$ sequences, there would be an exponential number of combinations (N^T). To overcome this limitation, we use a dynamic programming algorithm, i.e, the **Forward Algorithm**.

Algorithm 1 Forward Algorithm for Computing Partition Function

Require: Observation sequence $\mathbf{x} = (x_1, \ldots, x_T)$, parameters θ, state space $\mathcal{Y}$
Ensure: Partition function $Z(\mathbf{x})$
1: Initialize $\alpha_1(y) = (\psi_1(y, x_1, \emptyset))$ for all $y \in \mathcal{Y}$
2: **for** $t = 2$ to T **do**
3: **for** each $y \in \mathcal{Y}$ **do**
4: $\alpha_t(y) = \sum_{y' \in \mathcal{Y}} \alpha_{t-1}(y') + (\psi_t(y, x_t, y'))$
5: **end for**
6: **end for**
7: $Z(\mathbf{x}) = \sum_{y \in \mathcal{Y}} \alpha_T(y)$
8: **return** $Z(\mathbf{x})$

2.3 Learning

The weights to be learnt will model the transition scores, emission scores, the observation weights. We will use pre-implemented L-BFGS algorithm from PyTorch for optimization.

2.4 Optimizer

Limited-memory BFGS (L-BFGS or LM-BFGS) is an optimization algorithm in the family of quasi-Newton methods that approximates the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS) using a limited amount of computer memory.

2.5 Losses

Other details are further described in Sections 3.4 and 3.5

3 Implementation

3.1 Data and Problem Setup

The training and test data were provided in the following format:

- **Sentence #** : A numeric ID for the sentence.
- **Sentence** : A list of tokens (unchunked).
- **POS** : The list of POS tags mapped to each token in the sentence.
- **Label** : The list of NER labels mapped to each token in the sentence.

Training example:

```
Sentence # : 41607,

Sentence : Mr. Tsang is expected to win the July vote .

POS : ['NNP', 'NNP', 'VBZ', 'VBN', 'TO', 'VB', 'DT', 'NNP', 'NN', '.'],

Labels : ['B-per', 'I-per', 'O', 'O', 'O', 'O', 'O', 'B-tim', 'I-tim', 'O']
```

Sentence	Mr.	Tsang	is	expected	to	win	the	July	vote	.
POS	NNP	NNP	VBZ	VBN	TO	VB	DT	NNP	NN	.
Labels	B-per	I-per	O	O	O	O	O	B-tim	I-tim	O

Table 1: Tokenized Sentence with POS Tags and NER Labels

3.1.1 Data Exploration

Number of training examples $\approx 40K$, **Number of test examples** $\approx 10K$

There were 4 training examples in which there was a mismatch between the lengths of the tokenised sentence and number of labels. These can be ignored while training since it is a very small quantity.
Mismatch Found in Lengths:
Words=37, POS=38, NER=38
Words=28, POS=29, NER=29
Words=30, POS=29, NER=29
Words=52, POS=54, NER=54

Table 2: Dataset Statistics

Metric	Value
NER Categories in Train	17
NER Categories in Test	17
POS Categories in Train	42
POS Categories in Test	41
NER Overlap (Train ∩ Test)	17
POS Overlap (Train ∩ Test)	41
Unique NER in Train (not in Test)	0
Unique NER in Test (not in Train)	0
Unique POS in Train (not in Test)	1
Unique POS in Test (not in Train)	0

Table 3: NER Label Distribution

NER Label	Train Count	Test Count
O	710885	176877
B-geo	29979	7664
B-tim	16284	4049
B-org	16229	3913
I-per	13805	3445
B-per	13600	3389
I-org	13469	3315
B-gpe	12695	3175
I-geo	5964	1450
I-tim	5228	1300
B-art	316	86
B-eve	248	60
I-art	239	58
I-eve	200	53
I-gpe	159	39
B-nat	151	50
I-nat	39	12

POS Label	Train	Test
NN	116630	29152
NNP	105260	26160
IN	96788	24197
DT	78737	19704
JJ	62826	15575
NNS	60816	15017
.	38265	9562
VBD	31642	7735
,	26335	6414
VBN	25943	6382
VBZ	19817	5137
CD	19814	4880
VB	19273	4931
CC	19021	4688
TO	18444	4612
RB	16175	4072
VBG	15363	3759
VBP	12945	3209
PRP	10709	2601
POS	9031	2226
PRP$	6853	1801

POS Label	Train	Test
MD	5544	1427
WDT	2976	722
“	2962	761
JJS	2445	589
JJR	2390	574
NNPS	2049	472
WP	2031	511
RP	2019	470
WRB	1764	420
$	913	236
RBR	850	205
:	637	157
RRB	542	137
LRB	541	137
EX	527	136
RBS	234	61
;	159	55
PDT	117	30
WP$	82	17
UH	20	4
FW	1	0

3.1.2 Insights

- There is no missing NER or POS category in the train data, i.e, there is no label in the testing data that is unseen or unpredictable in theory.
- There is just 1 extra POS label in the train data (FW) which we need not worry about.
- The label distributions in both the training and testing sets are similar, i.e, the label counts in the test set for almost every label is roughly 20-25% of those in the train data.
- The POS labels are much better distributed than the NER labels. The number of occurrences of the NER label ‘O’ majorly outweighs all other NER labels. This becomes a problem, as we see later, and will be treated accordingly.
- If the token is a punctuation mark k, then token = POS = k, and NER = ‘O’.

3.1.3 Features and Parameters

Note: We will follow the following convention: N = number of unique NER labels P = number of unique POS tags O = number of observation feature functions
Our goal is to learn a weight matrix modelling the following probabilities:

- The transition probabilities between NER labels ($N \times N$ parameters).
- The initial and final probabilities (probabilities for the the NER label being the at the start or end of a sentence respectively, $2N$ parameters).
- The emission probability of emitting POS p given NER label y ($N \times P$ parameters)
- The probability of observing feature i given NER label y ($N \times P$ parameters)

Hence the total number of learnt parameters is $N^2 + 2N + NP + NO = \boxed{\boldsymbol{N(N+2+P+O)}}$.
While learning, we flatten the matrices and concatenate the four parts to be fed into the objective function minimizer.

3.2 Feature Extraction

Apart from the transition and POS emission probabilities, other binary features are also modelled:

```
obs_funcs = [
    start_cap, # if the token starts with a capital letter
    end_ing, # if the token starts with a 'ing'
    is_punct, # if the token is a punctuation mark
    is_digit, # if the token is composed of digits
    is_start, # if the token occurs at the start of the sentence
    is_end # if the token occurs at the end of the sentence
]
```

3.3 Parameter Initialization

The weights are initialized from a random (Normal) distribution with mean 0.0 and std 0.01.

3.4 Log Likelihood

Recall how the probability of the (hidden) NER label sequence Y given (observable) token sequence X and POS tag sequence Q is defined.
The scores are all calculated in the log-space (to account for underflow issues and the speed of addition compared to multiplication). Hence,

$$P(\mathbf{Y} = \tilde{\mathbf{Y}} | \mathbf{X} = \tilde{\mathbf{X}}, \mathbf{G} = \tilde{\mathbf{G}}) = \frac{P(\mathbf{Y} = \tilde{\mathbf{Y}}, \mathbf{X} = \tilde{\mathbf{X}}, \mathbf{G} = \tilde{\mathbf{G}})}{P(\mathbf{X} = \tilde{\mathbf{X}}, \mathbf{G} = \tilde{\mathbf{G}})} = \frac{e^{f_{\text{score}}(\tilde{\mathbf{X}}, \tilde{\mathbf{G}}, \tilde{\mathbf{Y}})}}{\sum_{\mathbf{Y'} \in S^T} e^{f_{\text{score}}(\tilde{\mathbf{X}}, \tilde{\mathbf{G}}, \mathbf{Y'})}}$$

$$\implies \log P(\mathbf{Y} = \tilde{\mathbf{Y}} | \mathbf{X} = \tilde{\mathbf{X}}, \mathbf{G} = \tilde{\mathbf{G}}) = f_{\text{score}}(\tilde{\mathbf{X}}, \tilde{\mathbf{G}}, \tilde{\mathbf{Y}}) - \log \sum_{\mathbf{Y'} \in S^T} e^{f_{\text{score}}(\tilde{\mathbf{X}}, \tilde{\mathbf{G}}, \tilde{\mathbf{Y}})}$$

Note that the last term is exactly the partition function.

$$\implies \boxed{\text{log-likelihood}(\tilde{\mathbf{Y}} | \tilde{\mathbf{X}}, \tilde{\mathbf{G}}) = f_{\text{score}}(\tilde{\mathbf{X}}, \tilde{\mathbf{G}}, \tilde{Y}) - f_{\text{partition}}(\tilde{\mathbf{X}}, \tilde{\mathbf{G}})}$$

where S = set of all NER labels encountered, N = length of input sequence.

3.5 Loss Function

Since L-BFGS is a convex optimization algorithm that minimizes the objective function, we want a loss function to be fed into the algorithm, minimizing which would lead to optimal weights.
So we define the loss function from the negative log-likelihood.
For the i^{th} training example,

$$l_i(\theta) = -\text{log-likelihood}(\mathbf{Y}_i^{\text{train}} | \mathbf{X}_i^{\text{train}}, \mathbf{G}_i^{\text{train}})$$

and overall loss over training batch:

$$L(\theta) = \frac{\sum l_i(\theta)}{N_{train}}$$

$$\implies \boxed{L(\theta) = \frac{1}{N_{train}} \sum (f_{\text{partition}}(\mathbf{X}_i^{\text{train}}, \mathbf{G}_i^{\text{train}}) - f_{\text{score}}(\mathbf{X}_i^{\text{train}}, \mathbf{G}_i^{\text{train}}, \mathbf{Y}_i^{\text{train}}))}$$

The weights are trained using L-BFGS optimizer from PyTorch.

3.6 Inference

Similar to the Forward Algorithm, we use DP for decoding. Instead of the total, we store the maximum probabilities and backtrace.

Algorithm 2 Viterbi Algorithm for CRF Decoding

Require: Observation sequence $\mathbf{x} = (x_1, \ldots, x_T)$, parameters θ, state space $\mathcal{Y}$
Ensure: Most likely label sequence $\mathbf{y}^* = (y_1^*, \ldots, y_T^*)$
Initialize $\delta_1(y) = \psi_1(y, x_1, \emptyset)$ for all $y \in \mathcal{Y}$
Initialize $\psi_1(y) = 0$ for all $y \in \mathcal{Y}$
for $t = 2$ to T **do**
    for each $y \in \mathcal{Y}$ **do**
        $\delta_t(y) = \max_{y' \in \mathcal{Y}} \{\delta_{t-1}(y') + \psi_t(y, x_t, y')\}$
        $\psi_t(y) = \arg\max_{y' \in \mathcal{Y}} \{\delta_{t-1}(y') + \psi_t(y, x_t, y')\}$
    end for
end for
$y_T^* = \arg\max_{y \in \mathcal{Y}} \delta_T(y)$
for $t = T - 1$ down to 1 **do**
    $y_t^* = \psi_{t+1}(y_{t+1}^*)$
end for
return $\mathbf{y}^* = (y_1^*, \ldots, y_T^*)$

4 Training

There was a lot of experimentation and finetuning done during training, and some problems were encountered.

4.1 Major Problems Encountered

1. **Skewedness of NER label counts:** [3] states that:

   > *One overly simple method of measuring accuracy is merely to count what fraction of all tokens in the text were correctly or incorrectly identified as part of entity references (or as being entities of the correct type). This suffers from at least two problems: first, the vast majority of tokens in real-world text are not part of entity names, so the baseline accuracy (always predict "not an entity") is extravagantly high, typically* $> 90\%$*; and second, mispredicting the full span of an entity name is not properly penalized (finding only a person's first name when his last name follows might be scored as* $\frac{1}{2}$ *accuracy).*

   This simply means that occurrences of the label 'O' in the training data are a lot more compared to other labels, and hence naively training the model results in most of the predictions to be tagged as 'O'. This results in extremely high recall for 'O' and boosts accuracy (and likelihood) without providing any actually useful information. To curb this, I used a **NER class-wise weighing** technique (see Section 4.2), in which scores derived from highly occurring labels are given less importance.

2. **Sequential training time:** Since the Forward algorithm for calculating the partition function and the Viterbi algorithm are both sequential algorithms (the score at each timestep depends on the previous one), the computation can't be fully parallelized. Training on all examples (of the training split) was taking an unreasonable amount of time using the SciPy optimize library. The PyTorch optimizer showed much improved performance in terms of training speed, hence it was possible to run the training loop for more epochs and get lesser loss.

4.2 NER class-wise weighing

To curb the problem of skewed label distribution, we assign class weights to each NER label y.

$$\boxed{\text{class-weight}(y) = \frac{\text{Total count of all labels in training set}}{\text{Total count of label } y \text{ in training set}}}$$

These weights are then normalized by dividing all of them by the maximum of these.

$$\boxed{\text{normalised-class-weight}(y) = \frac{\text{class-wt}(y)}{\max_y \text{class-wt}(y)}}$$

For the BOS and EOS tags, a dummy value of 0.01 is used.

5 Trial Configurations

Table 4: Performance of Trial Configurations

Configuration	Accuracy
Random initialization, no training	0.033%
Tag all tokens as 'O's, featureless classification (*Baseline*)	84.30%
Not conditioning POS emissions and observations on NER label	84.53%
Not conditioning observations on NER label	85.69%
Conditioning every feature, no NER class weighing	85.71%
Log-smoothed NER class weighing	83.46%
Equal contribution of emission and transition scores ($\lambda_i = 1.0$)	84.44%
Double Class weighing for transitions	85.93%
Unsmoothed NER class weighing, conditioned emissions, $\lambda_E = 0.5$, $\lambda_T = 2.0$ **(Final)**	87.98%

5.1 Points to note:

- *Not conditioning features* means that their is a single vector describing the weight allotted to the corresponding features. *Conditioning features* means that the weights are in a matrix and each weight corresponds to the current feature and the current NER label while training.
- *Log smoothing:* For smoothing a value x, it simply means replacing it by $\log(1 + \gamma x)$ for some smoothing factor γ. It makes the distribution of the weights less convoluted (less varying).
- *Double class weighing* means multiplying by the weights of both the previous and current NER, as opposed to only one of them.

5.2 Visualization for intermediate suboptimal training configurations

No conditioning and weighing (less training examples)

We can see how skewed the distribution is.

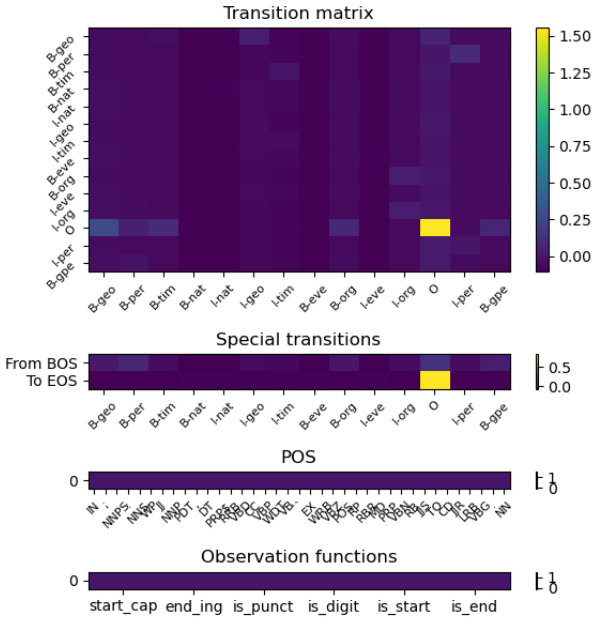

Figure 1: No Conditioning of emissions and observations

Weighing, No conditioning (less training examples)

Introduces a smoother distribution.

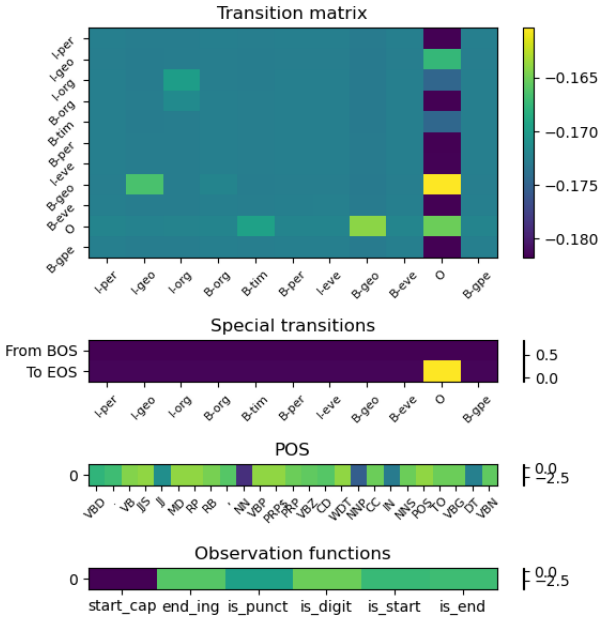

Figure 2: Weighted transitions

Weighing all features

We can see that this leads to over-smoothing of the distribution, it reduces accuracy.

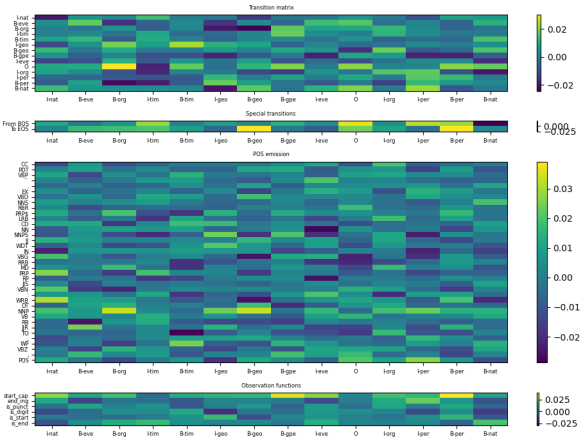

Figure 3: Weighing all transitions

6 Final Configuration and Evaluation

6.1 Metrics

For this labeling task, the following metrics are reported on the test set:

- Overall Accuracy
- Labelwise Precision
- Labelwise Recall
- Labelwise F1-score

6.2 Results

6.2.1 Model

Final Model Configurations

Epochs: 30
Learning Rate: 0.1
Optimizer: L-BFGS
Final Accuracy: 87.98%
$\lambda_E = 0.50$
$\lambda_T = 2.00$
Number of observation functions: 6
Parameter count: N(N+P+O+2)
Training Set Size: 40K examples
Testing Set Size: 10K examples
Regularization: Unregularized explicitly, regularised through NER class weighing

6.3 Labelwise Performance

Table 5: Labelwise Performance

Label	Precision	Recall	F1-Score
O	0.9490	0.9897	0.9690
B-GEO	0.2820	0.8730	0.4263

The remaining tags have very low scores.

6.4 Examples

Correct predictions

Sentence	Aid	agencies	say	five	million	of	Malawi	's	12	million	people	need	food	aid	.
POS	JJ	NNS	VBP	CD	CD	IN	NNP	POS	CD	CD	NNS	VBP	NN	NN	.
Labels	O	O	O	O	O	O	B-geo	O	O	O	O	O	O	O	O
Predictions	O	O	O	O	O	O	B-geo	O	O	O	O	O	O	O	O

Table 6: Example 1

Sentence	He	replied	that	Taiwan	is	“	probably	the	biggest	.	“
POS	PRP	VBD	IN	NNP	VBZ	“	RB	DT	JJS	.	“
Labels	O	O	O	B-geo	O	O	O	O	O	O	O
Predictions	O	O	O	B-geo	O	O	O	O	B-art	O	O

Table 7: Example 2

Sentence	There	has	been	no	public	comment	from	Syria	.
POS	EX	VBZ	VBN	DT	JJ	NN	IN	NNP	.
Labels	O	O	O	O	O	O	O	B-geo	O
Predictions	O	O	O	O	O	O	O	B-geo	O

Table 8: Example 3

Incorrect predictions

Sentence	He	trails	Alonso	85	to	111	.
POS	PRP	VBZ	NNP	CD	IN	CD	.
Labels	O	O	B-per	O	B-tim	I-tim	O
Predictions	O	O	B-geo	O	O	O	O

Table 9: Example 4

Sentence	A	Sunni	Muslim	group	,	Jundollah	,	claimed	responsibility	for	that	attack	.
POS	DT	NNP	NNP	NN	,	NNP	,	VBD	NN	IN	DT	NN	.
Labels	O	B-per	I-per	O	O	B-org	O	O	O	O	O	O	O
Predictions	O	B-geo	B-geo	O	O	B-geo	O	O	O	O	O	O	O

Table 10: Example 5

Insights

We can see that

- 'O' and 'B-geo' are mostly prevalent in the predictions because they are the highest occurring in the dataset.
- the occurrence of named entities is captured well but the distinguishing power is low.

7 Discussion

7.1 Determinism of the Algorithm

The inference process is deterministic in the sense that it picks the maximum probable sequence during Viterbi decoding. This makes sense mathematically, but in addition to the skewed label distribution in the training data, this results in mispredictions also skewed towards the top highly occurring labels (''O' and 'B-geo' here).

7.2 Diversity of Predicted Labels

Again due to the determinism, there is skewedness and thus less diversity in predictions. Trying to alter the class weighing scheme results in more diverse predictions, but increases the chance of misprediction, and thus decreases recall and accuracy. Hence, *there is a trade-off between prediction diversity and accuracy.*

7.3 Analysing correlations between trained weights and real-world language patterns

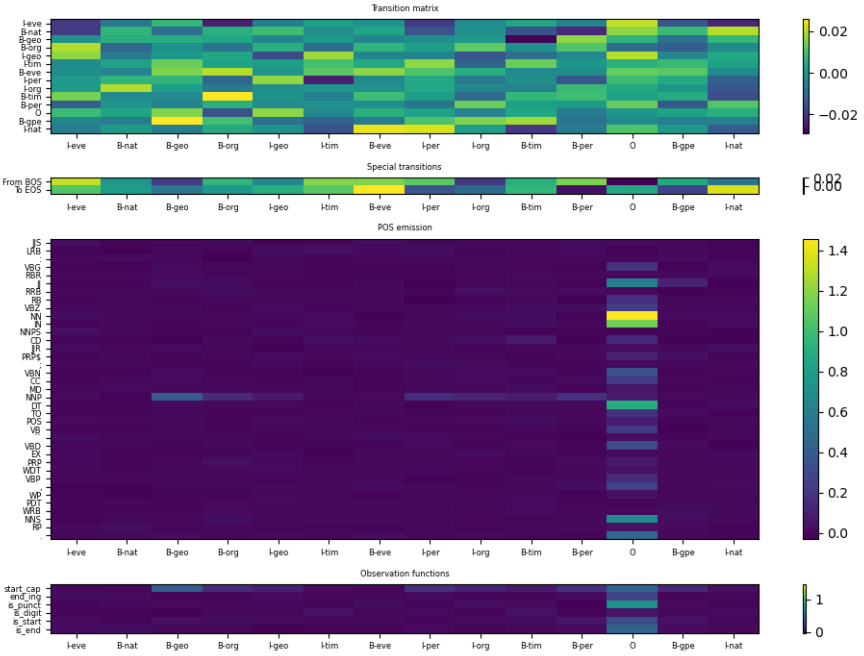

Figure 4: Final Trained Weights Heat Map

- The tags B-PER, I-PER, B-GEO, I-GEO, B-GPE, are highly correlated with the feature `start_cap`.
- For the NNP tag (proper nouns), we can see good correlation with different named entity tags that are not O, which should be expected, especially with 'B-GEO'.
- NNS, NN, DT, N contribute heavily to 'O', which is also expected.
- `is_punct` contributes heavily to 'O' because all punctuation marks are classified as 'O', which is a good sign that the model recognizes such almost 'hard-coded' patterns.
- It is clear that the scaling of the values are different for the three sets of weights, hence it is logical to use the different λ-contributions to the final score.

7.4 Analysing mispredictions

Building up on the premise of the tradeoff between diversity and F1-score, the most prevalent tags tend to be predicted more.
We can see that

- 'O' and 'B-geo' are mostly prevalent in the predictions because they are the highest occurring in the dataset.
- the occurrence of named entities is captured well but the distinguishing power is low.
- The numbers are mostly labelled as 'O' in the training data, so they end up taged as 'O' rather than, for example, 'B-TIME'.
- The high correlation of NNP with 'B-GEO' is apparent in both the trained weights and the predictions/mispredictions.

7.5 Scope for improvement

We see that the accuracy caps at 88% for our model at 6 features. If we have more compute resources and training time, we can do extensive feature engineering (taking next, previous POS, using word embeddings, using larger set of suffixes etc.) to increase the accuracy.

References

[1] Eric Brill. "Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging". In: *Computational Linguistics* 21.4 (1995). arXiv:cmp-lg/9505040, pp. 543–565.

[2] Charles Sutton and Andrew McCallum. "An introduction to conditional random fields". In: *arXiv preprint arXiv:1011.4088* (2010). URL: https://arxiv.org/abs/1011.4088.

[3] Wikipedia contributors. *Named-entity recognition*. Wikipedia, The Free Encyclopedia. Accessed: 19 March 2025. 2025. URL: https://en.wikipedia.org/wiki/Named-entity_recognition.