Usage of Datasets for NLP & Treebank Annotation for Linguistic Description

Usage of Datasets in NLP

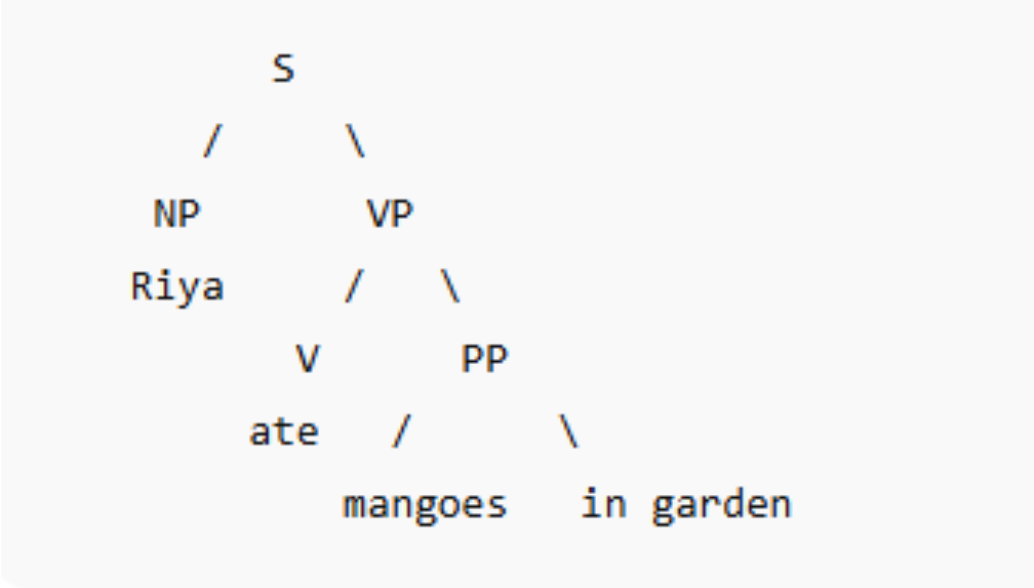
- A dataset in NLP is a **collection of text** that has been organized and labelled so that a computer can learn language patterns.

Why NLP needs datasets?

Task	Example	What Dataset Teaches the Computer
Sentiment Analysis	"This movie is awesome!"	Positive/Negative tone
Translation	"Good Morning → Suprabhat"	Meaning across languages
Speech to Text	Voice typing	How words sound and appear in text
Chatbots	Customer support bot	How humans ask questions

What is a Treebank?

- Riya ate mangoes in the garden



Explain roles

- **NP** = Noun Phrase → "Who?"
- **VP** = Verb Phrase → "What action?"
- **PP** = Prepositional Phrase → "Where / When?"

Treebank = Sentence → broken like a branch→ to show relationships.

Why Treebank Are Used

Purpose	Explanation
Linguistic Research	Understand sentence grammar
NLP Model Training	Teach computer how sentences are structured
Grammar Checking Tools	Helps identify incorrect sentences
Speech Assistants	Helps detect meaning and intent

POS tagging = labeling word type

Treebank annotation = showing **how** words connect

How HMM Works (Hidden Markov Model)

1. We are trying to find something that we **cannot see** (Hidden State)

2. Using something we **can see** (Observed Output).

Real-life Analogy

- Imagine a **friend texting you**, but you don't see their face.
- You try to guess their **mood (hidden)** based on their **words (observable)**.

Message (Observed)	Guess Mood (Hidden)
"Yessss 👏"	Happy 😊
"Ok."	Sad 😔
"I don't care"	Angry 😡

Components of HMM

Term	Means	Example
Hidden States	Things we want to predict	POS tags (Noun, Verb) or Emotions
Observations	Words we see in sentence	"Riya eats mangoes"
Transition Probability	How likely one hidden state follows another	Noun → Verb likely
Emission Probability	How likely a word comes from a hidden state	"eats" is likely a Verb

Maximum Entropy Markov Model (MEMM)

- HMM chooses labels based on **probabilities of transitions + word likelihoods**, but:
- It **cannot use many features** (context, suffix, capitalization, etc.)
- It assumes everything relies only on the previous state (Markov property)
- So researchers needed a model that:
  ✅ Uses **more clues/features**
  ✅ Is **more flexible**
  ✅ Doesn't assume strict probability constraints

where **MEMM** comes in.

What is MEMM?

- MEMM is a **sequence labelling model** that predicts the **current label using:**
- The **previous label**
- **Features from the current word**
- It uses **Maximum Entropy (Logistic Regression)** for prediction instead of conditional probability tables.
- **MEMM is:**
  "HMM + Logistic Regression + Flexible Features"

Real-Life Analogy

- Identifying the **part of speech** of words while reading a sentence.
- In HMM, We just look at the previous word.
- But in MEMM You also check:

1. Does the word start with a capital letter?
2. Does it end with -ing?
3. Is it near a verb?

So, MEMM is like:

- "Use **as many clues as possible** before deciding."

Difference

Feature	HMM	MEMM
Uses probabilities	Yes	No
Uses features (suffixes, capitalization, context)	No	Yes ✅
Classifier	Markov Chains	**Logistic Regression / MaxEnt**
Flexibility	Low	**High**

MEMM Example (POS Tagging)

Sentence: **He plays football**

We want to assign POS labels:

Word	Possible POS
He	Pronoun
plays	Verb / Noun
football	Noun

MEMM considers features like

Word Feature	Example
Word shape	capital letter?
Suffix	"-s" → verb?
Previous tag	Pronoun → Verb likely
Word frequency	plays = verb often

Step-by-Step Decision:

1. **He** → No suffix → Starts sentence → Pronoun
2. **plays** → Ends in "-s" → Previous label was Pronoun → Likely a **Verb**
3. **football** → No capital, common noun → **Noun**

```
He        → PRON
plays     → VERB
football  → NOUN
```

MEMM Logic (One Line)

- **Current Tag = f(Previous Tag + Word Features)**
  (using **Maximum Entropy / Logistic Regression**)

We want to find the **Part-of-Speech (POS)** of each word.

Word	What we See (Observed)
Riya	Word
eats	Word
mangoes	Word

But POS **tags are hidden:**

Word	Hidden Tag
Riya	Noun
eats	Verb
mangoes	Noun

What MEMM Does

- MEMM predicts the **current word's tag** using:

1. The **previous word's tag**
2. **Features** of the current word

Important: MEMM Uses Features (This is what makes it different from HMM)

Example Features Used by MEMM:

Word Feature	Why?
Capital letter?	Names often start with capital letter
Ends with "-s"?	Verbs often end in "s" (eats, runs)
Previous word was noun?	Then next likely is a verb
Is the word common noun in dictionary?	May be noun

Apply MEMM

- **Step 1: Tag first word → "Riya"**

Features:

- Begins with capital → **Probably a Proper Noun**
- Position: First word in sentence → Usually noun/pronoun

```
Riya → Noun (NNP)
```

Apply MEMM

Step 2: Tag next word → "eats"

Features:

- Ends with **-s** → Often a **Verb** in present tense
- **Previous tag was Noun** → Likely **Verb** (Subject → Action)

```
eats → Verb (VBZ)
```

Apply MEMM

- Step 3: Tag last word → "mangoes“

Features:

- Plural form **-es** → Often **Noun**
- Follows Verb → Object is likely **Noun**

```
mangoes → Noun (NNS)
```

Final Output by MEMM

Word	POS Tag (Predicted)
Riya	NNP (Proper Noun)
eats	VBZ (Verb)
mangoes	NNS (Plural Noun)

Conditional Random Field (CRF) in NLP

1. CRF is a **sequence labelling model** used in NLP.
2. It assigns a **label** to each word in a sentence, but **considers the context of the whole sentence**, not just the previous word.

Where CRF is Used?

Task	Example
POS Tagging	cat (NOUN), runs (VERB)
Named Entity Recognition (NER)	Delhi (LOCATION)
Chunking	"the red dress" (NOUN PHRASE)
Information Extraction	extracting dates, names

What is Chunking

- Chunking is the process of **grouping words into meaningful phrases** (also called **shallow parsing**).
- Instead of analyzing the **full sentence structure**, chunking only identifies **phrases** such as:
- **Noun Phrase (NP)**
- **Verb Phrase (VP)**
- **Prepositional Phrase (PP)**

Goal

- Combine related words into phrases.

JJ is the **POS (Part-of-Speech) tag** for an **Adjective** in NLP.

POS Tag Meaning

Tag	Meaning	Example
JJ	Adjective (basic form)	red, small, beautiful
JJR	Comparative adjective	smaller, bigger, faster
JJS	Superlative adjective	smallest, biggest, fastest

Example in Sentence:

Sentence:
"The red dress"

Word	POS Tag
the	DT (Determiner)
red	**JJ** (Adjective)
dress	NN (Noun)

→ So in chunking:

```scss
[ the/DT red/JJ dress/NN ] → Noun Phrase (NP)
```

2. Why CRF is Needed?

Earlier models → **HMM and MEMM**

But they have problems:

Model	Issue
HMM	Can't use rich features
MEMM	Suffers from *Label Bias* (chooses local best, not global best)

Label Bias

Label Bias: Model tends to choose labels with fewer outgoing transitions due to local normalization.

CRF Solution: Uses **global normalization** so predictions depend on the full sequence context, not local states.

MEMM vs. CRF.

Example sentence

```csharp
He is eating food
```

We want to assign POS tags:

- **N** = Noun
- **V** = Verb

Suppose we are tagging the word **"eating"**.

Now assume in our model:

Current Label	Possible Next Labels	Count
N	{N, V}	2
V	{N, V, V} (Verb can repeat or go noun)	3

So **N has 2 outgoing choices, V has 3 outgoing choices.**

Because of local normalization (as in MEMM):

- If probabilities are distributed equally:
  - From **N**: each path gets probability = 1/2
  - From **V**: each path gets probability = 1/3

So the model sees **N as stronger** (because probability mass is concentrated on fewer paths).

Even **if the sentence structure indicates "eating" should be Verb**, the model gets biased:

```mathematica
MEMM result:
He (N) is (N) eating (**N**) food (N)
```

It tags "**eating**" as **Noun** incorrectly,

because "N" has fewer paths and seems more confident.

This is **label bias**:

The model **prefers labels with fewer transitions**, not based on correctness.

Core Idea of CRF

- CRF predicts labels for all words by considering the entire sentence at once.
- Not word-by-word,
  Not one previous state,
  But **full sequence context**.

Example Sentence

"Riya lives in Pune."

We want to **label each word** in the sentence for **Named Entity Recognition (NER).**

Word	What we want to label (Output)
Riya	PERSON
lives	VERB
in	PREPOSITION
Pune	LOCATION

Step 1: Observations (Words)

```
Riya   lives   in   Pune
```

Step 2: Extract Features for Each Word

CRF uses **features**, not probabilities like HMM.

Common Features:

Feature Type	Example for our sentence	Why it helps
Capitalization	"Riya", "Pune" start with capital letters	Proper nouns / names
Word Shape	"Riya" (Aa), "in" (aa), "Pune" (Aa)	Helps detect names
Suffix	"-s" in lives	Verb marker
Context Words	Word before and after	Context improves accuracy

Feature Table

Word	Capital?	Suffix	Previous Word	Next Word
Riya	Yes	None	START	lives
lives	No	-s → Verb pattern	Riya	in
in	No	None	lives	Pune
Pune	Yes	None	in	END

Step 3: Use Features + Neighbour Labels to Decide Final Tags

CRF labels the entire sentence **considering context**:

Rule Patterns Learned by CRF During Training

- Names usually start with **Capital letter**
- A **Verb** often comes after a **Person**
- Location often comes **after** the word "in"

Apply Rules to Our Sentence

Word	Clues	Best Label
Riya	Starts with capital, appears before verb	PERSON
lives	Ends with -s, follows a noun	VERB
in	Common preposition	PREP
Pune	Capitalized, follows "in"	LOCATION

Final CRF Output

```
Riya → PERSON

lives → VERB

in → PREP

Pune → LOCATION
```

Why CRF is Better?

Model	Decides Label Based On	Problem	CRF Advantage
HMM	Probabilities only	Cannot use rich features	CRF uses many features
MEMM	Previous local decision	Label Bias Problem	CRF uses global optimization
CRF	Whole sentence at once	—	Best overall sequence ✅

Support Vector Machines (SVM) in NLP

- What is SVM?

SVM is a classification algorithm that separates data into **categories** by drawing the **best possible boundary** (called a **hyperplane**) between them.

"Drawing the cleanest separating line between two groups."

Simple Real-Life Analogy

Imagine you have:

- **Mangoes** on one side of a table
- **Apples** on the other
- You want to place **a scale (a line)** between them so that:
- No mango crosses to apple side
- No apple crosses to mango side
- **The line that separates them best = SVM hyperplane.**

Why SVM Works Well in NLP?

Because in NLP, text is converted into **vectors** (numbers), e.g.,

- Bag-of-Words
- TF-IDF
- Word Embedding
- Once words/sentences become numbers → **SVM can classify them efficiently.**

Where SVM is Used in NLP?

NLP Task	Example
Sentiment Analysis	Positive vs Negative reviews
Spam Detection	Spam vs Not Spam
Document Classification	Sports vs Politics vs Health
Fake News Detection	Real vs Fake text
Author Style Identification	Writer A vs Writer B

SVM Example in NLP

- Sentence Sentiment Classification:
  We want to detect **Positive** or **Negative** review.

```
"I love this movie!" → Positive

"This movie is terrible." → Negative
```

Step 1: Convert Text into Numbers (TF & IDF)

We use **TF-IDF**, which stands for:

Term	Meaning	Why?
TF (Term Frequency)	How many times a word appears in a document	Measures importance in that document
IDF (Inverse Document Frequency)	How rare a word is across all documents	Reduces importance of common words like "the", "is", "and"

Formula

$$\text{TF-IDF} = \text{TF} \times \text{IDF}$$

- **TF = Word importance inside the document**
- **IDF = Word importance across the dataset**

Common words → low IDF

Unique keywords → high IDF

Step 2: Example Sentence Classification (Sentiment Analysis)

Sentences:

1. **"I love this movie"** → Positive
2. **"This movie is terrible"** → Negative

Vocabulary List (Words that matter):

love, movie, terrible

TF Calculation (Count occurrences)

Sentence	love	movie	terrible
I love this movie	1	1	0
This movie is terrible	0	1	1

IDF Calculation

- **love** appears in **1** sentence → rare → **high IDF**
- **movie** appears in **both sentences** → common → **low IDF**
- **terrible** appears in **1** sentence → rare → **high IDF**

So movie will have small weight, love and terrible will have higher weight.

Step 3: SVM Classification

SVM now tries to **separate** these vectors:

```
markdown

Positive reviews (+)      |      Negative reviews (-)
-----------------------------------------------------
                  Best separating boundary (Hyperplane)
```

SVM chooses the line / plane that *maximizes the margin* (distance) between classes.

So even if a new review appears:

New Input:

"I love it"

TF-IDF → (high, 0, 0)

This vector lies on **positive side** → SVM predicts:

Task	Process
Convert words → numbers	Use TF-IDF
Classify sentiment	Use SVM to draw best separating boundary