CS 772: A1
A Comparative Study of POS Tagging Models

Siddhartha Rajeev, Shanttanu Oberoi

August 30, 2025

Outline

Problem Statement

- Task: Assign a Part-of-Speech (POS) tag to each word in a sentence.
- Input: sequence of words $w_1, w_2, \ldots, w_T$
- Output: sequence of tags $t_1, t_2, \ldots, t_T$
- Evaluation metrics: Precision, Recall, F1-score
- Compare: HMM, Encoder-Decoder (LSTM), and LLM
- Referred to Speech and Language Processing (Jurafsky) for both HMMs and LSTMs

Dataset: Brown Corpus

- Source: NLTK Brown corpus with Universal tagset
- Tags: 12 universal categories
- Preprocessing:
  - Lowercasing
  - Handling out-of-vocabulary tokens with `UNK`
  - Padding sequences with `PAD`
  - Train/Val/Test split: 85/10/5

HMM Theory

- Hidden Markov Model assumes joint distribution:

$$P(t_1^T, w_1^T) = \prod_{i=1}^{T} P(w_i \mid t_i)\, P(t_i \mid t_{i-1})$$

- Transition probabilities: $P(t_i \mid t_{i-1})$
- Emission probabilities: $P(w_i \mid t_i)$
- Inference: most probable tag sequence $\hat{t}_1^T$ via Viterbi.

Viterbi Definitions

- Define $V_i(s)$ = best probability of a tag sequence ending in state s at position i.
- Recurrence relation:

$$V_i(s) = \max_{s'} \left(V_{i-1}(s') \cdot P(s \mid s') \cdot P(w_i \mid s) \right)$$

- Backpointer $\psi_i(s)$ stores the best previous state.

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

Figure: Viterbi definitions illustration.

Viterbi Trellis Diagram

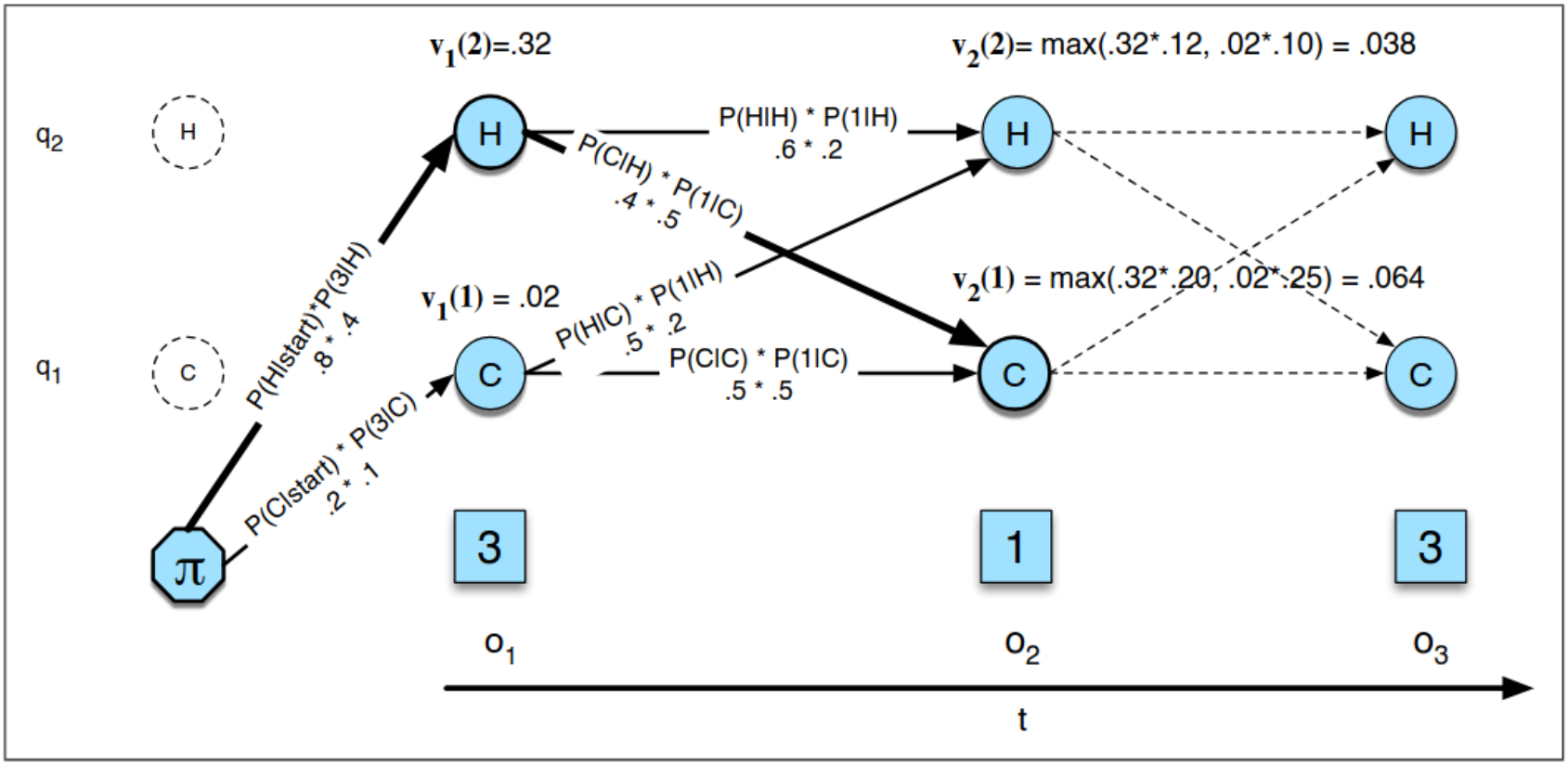


Figure: Trellis structure for dynamic programming .

Viterbi Algorithm

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*, *path-prob*

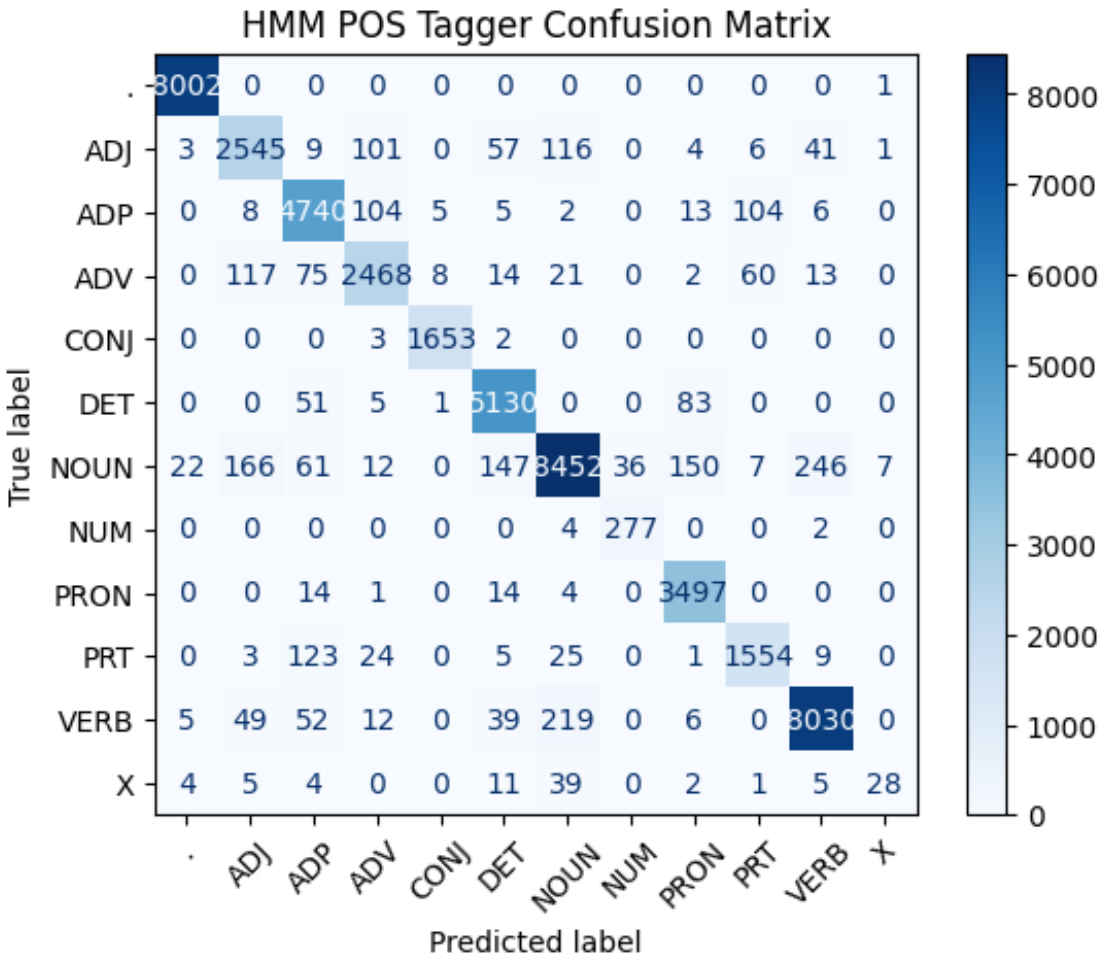
create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
  $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
  $backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step
bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

Figure: Algorithm flowchart / pseudocode box .

- Complexity: $O(T \cdot N^2)$.
- With $N = 12$ tags, essentially linear in sequence length T.

HMM Results

- Classification metrics:
  - Accuracy, Precision, Recall, F1-score (table).
- Error analysis via confusion matrix:



POS Tagging Performance

Tag	Precision	Recall	F1-score	Support
.	0.9963	0.9998	0.9980	8003
ADJ	0.8816	0.8862	0.8839	2883
ADP	0.9260	0.9511	0.9384	4987
ADV	0.9028	0.8891	0.8959	2778
CONJ	0.9934	0.9970	0.9952	1658
DET	0.9526	0.9725	0.9624	5270
NOUN	0.9518	0.9120	0.9315	9306
NUM	0.8878	0.9788	0.9311	283
PRON	0.9348	0.9907	0.9619	3530
PRT	0.8874	0.8853	0.8863	1744
VERB	0.9617	0.9578	0.9597	8412
X	0.7353	0.2525	0.3759	99
Accuracy			0.9485	48953
Macro Avg	0.9176	0.8894	0.8934	48953
Weighted Avg	0.9484	0.9485	0.9480	48953

- Overall accuracy: **94.85%**
- Strong performance on common tags (NOUN, VERB, DET).

Distributional Shift Analysis

- Compare predicted vs true tag distributions.
- Identify over-/under-predicted POS categories.

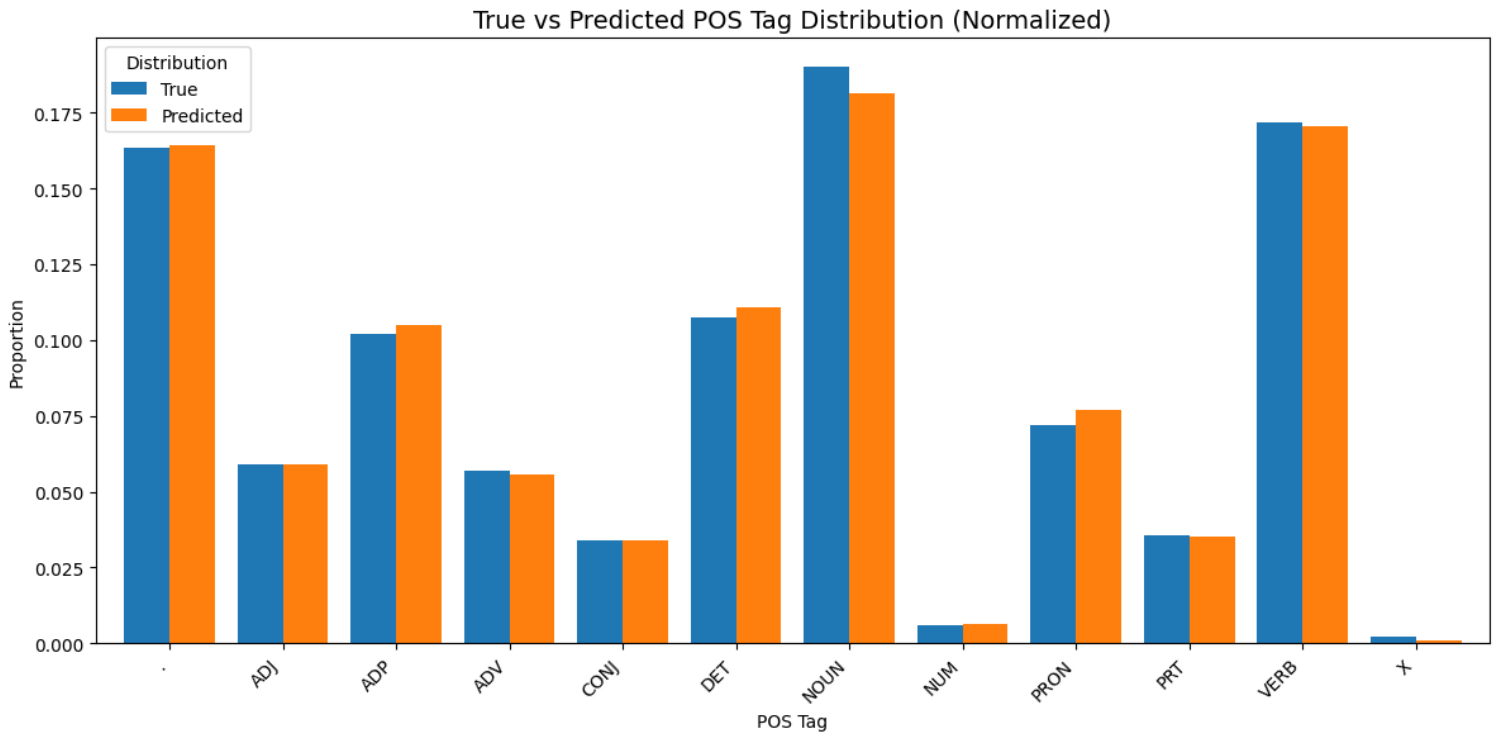


Figure: Distributional shift plot between true vs predicted tags .

LSTM Encoder-Decoder: Theory

- Input embedding: $x_t = E[w_t]$.
- LSTM recurrence:

$$h_t, c_t = \text{LSTM}(x_t, h_{t-1}, c_{t-1})$$

- Final hidden states from encoder $\rightarrow$ initial state for decoder.
- Decoder generates tag sequence autoregressively.

Decoding Algorithm

- Greedy decoding:

$$\hat{t}_t = \arg\max_y P(y \mid h_t)$$

- Teacher forcing during training: decoder input = true previous tag.
- At inference: decoder input = predicted previous tag.

LSTM Encoder-Decoder Diagram

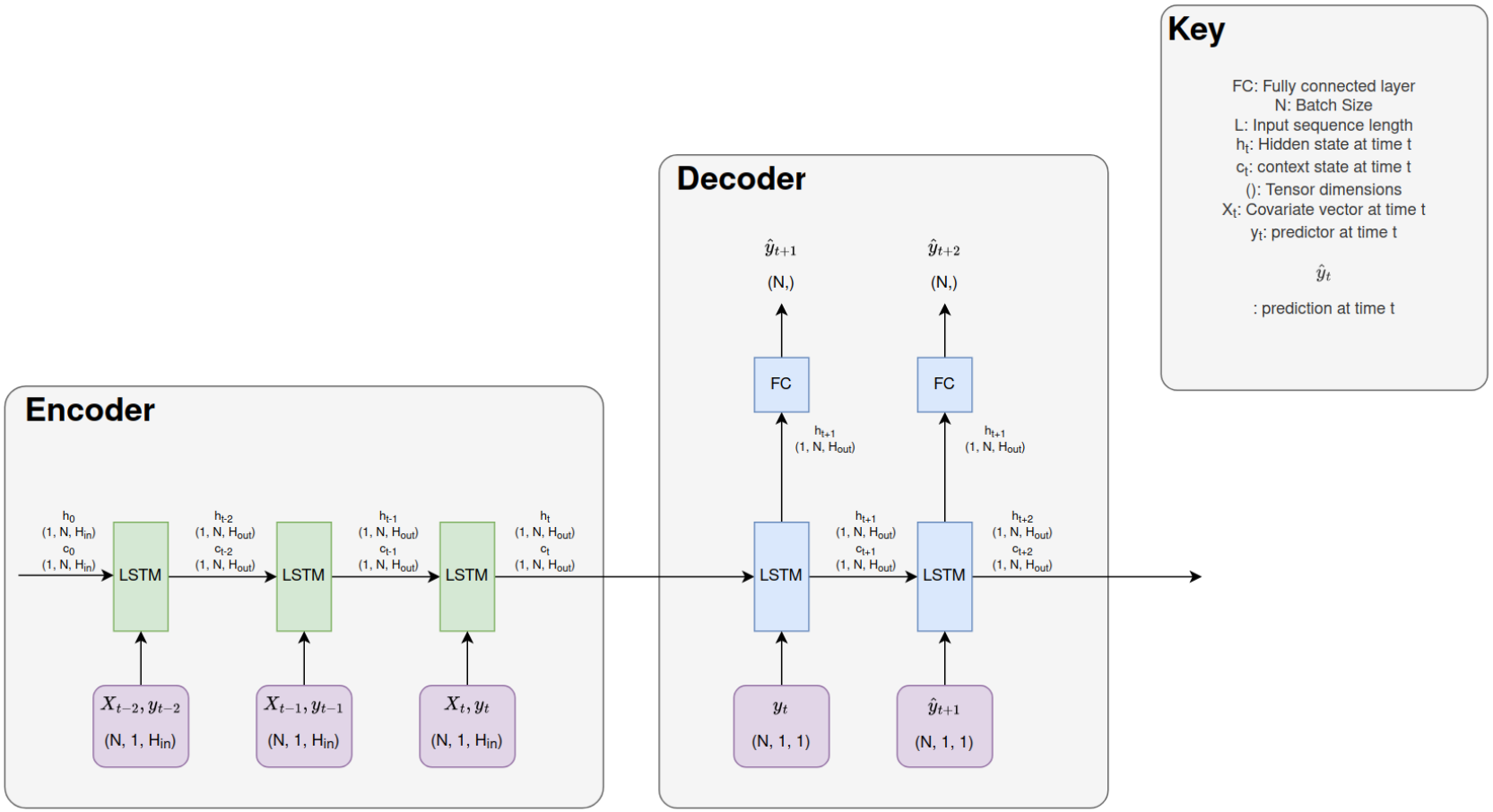
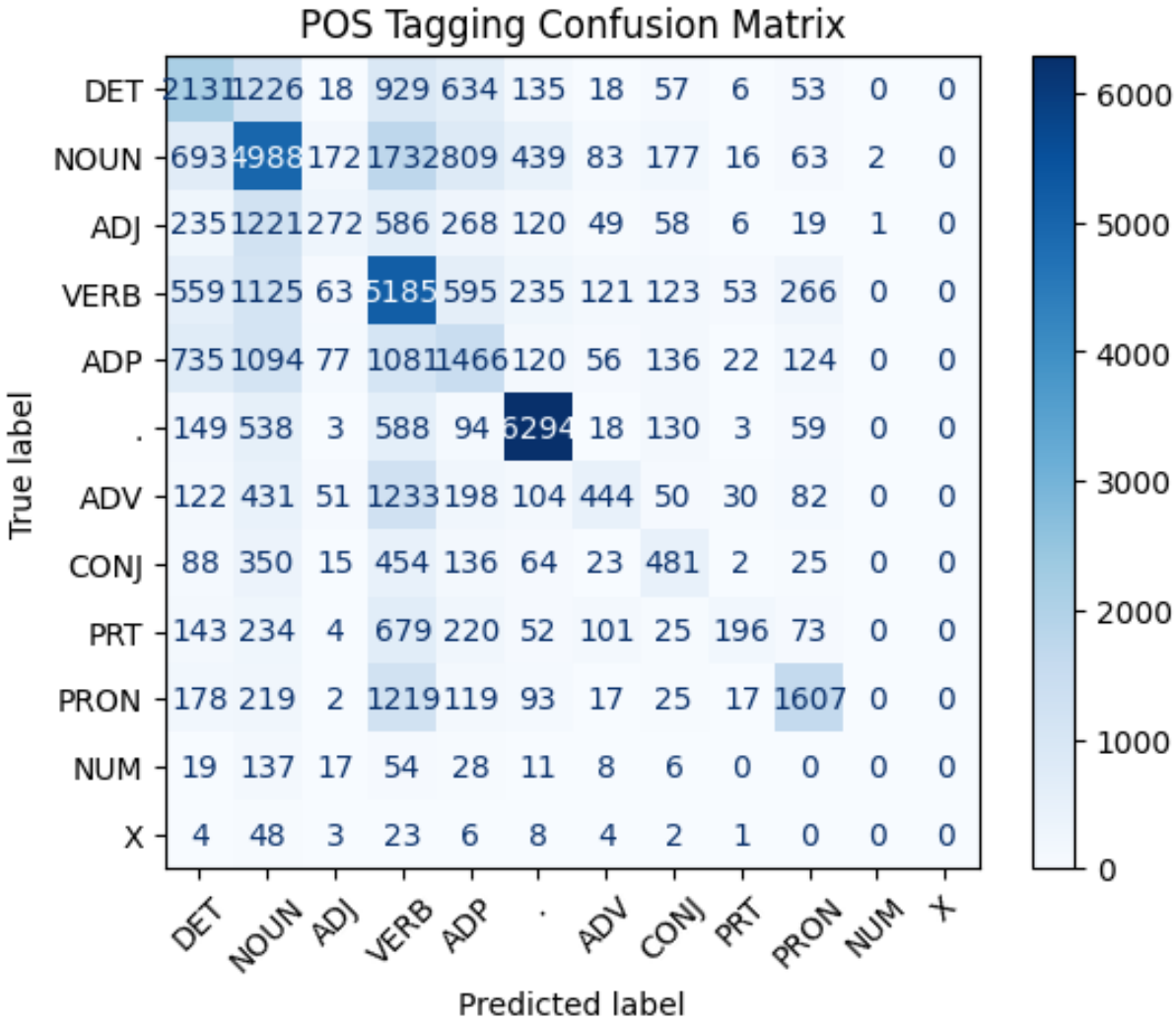


Figure: Schematic of encoder-decoder LSTM for POS tagging .

LSTM Results

- Classification metrics: Accuracy, Precision, Recall, F1-score.
- Error patterns captured in confusion matrix.



POS Tagging Performance (LSTM Model)

- Overall accuracy: **48%**
- Performance varies significantly across POS tags

Tag	Precision	Recall	F1
DET	0.42	0.41	0.42
NOUN	0.43	0.54	0.48
ADJ	0.39	0.10	0.15
VERB	0.38	0.62	0.47
ADP	0.32	0.30	0.31
.	0.82	0.80	0.81
ADV	0.47	0.16	0.24
CONJ	0.38	0.29	0.33
PRT	0.56	0.11	0.19
PRON	0.68	0.46	0.55
NUM	0.00	0.00	0.00
X	0.00	0.00	0.00
Macro Avg	0.40	0.32	0.33
Weighted Avg	0.49	0.48	0.46

Distributional Shift: LSTM

- Compare predicted vs true POS distributions.
- Identifies systematic biases (e.g. overpredicting NOUN, underpredicting ADV).

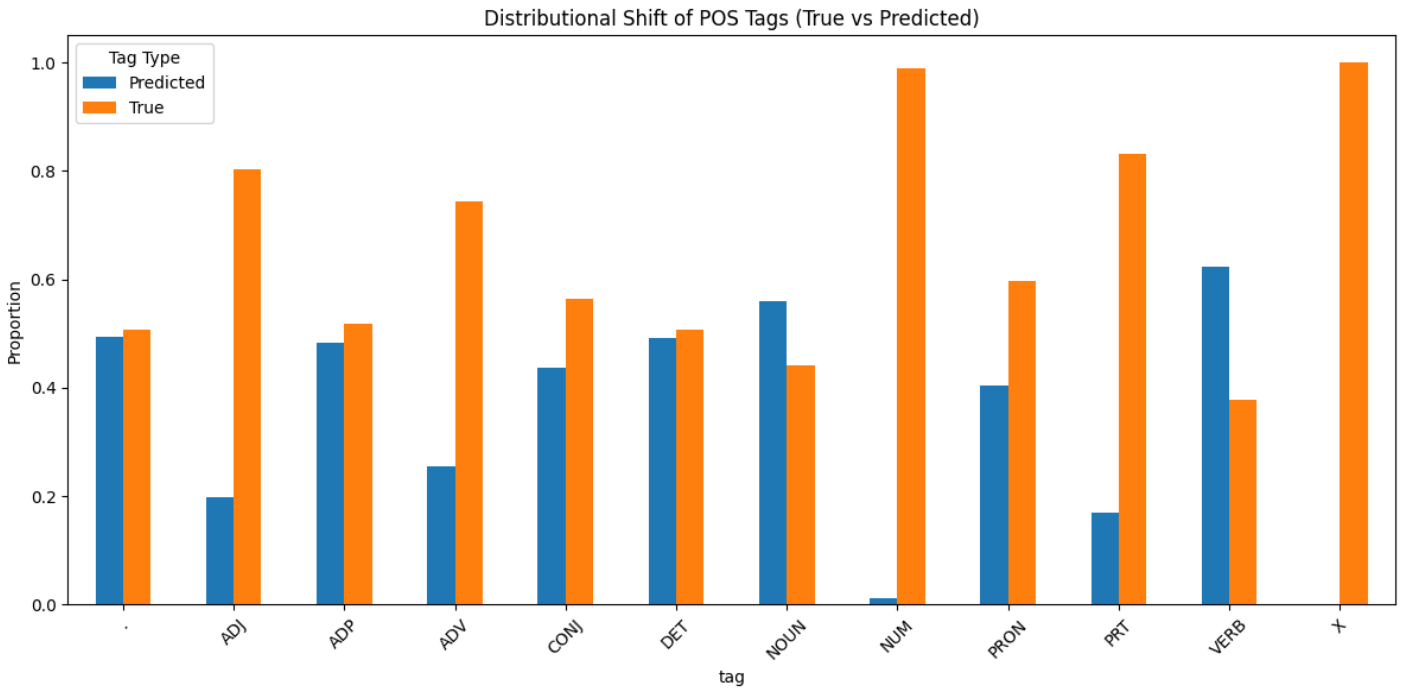


Figure: Distributional shift between predicted vs true tags .

Encoder-Only LSTM: Theory

- Each input word is embedded: $x_t = E[w_t]$.
- LSTM recurrence encodes sequential dependencies:

$$h_t, c_t = \text{LSTM}(x_t, h_{t-1}, c_{t-1})$$

- Tag prediction at each step:

$$\hat{t}_t = \arg\max_y P(y \mid h_t)$$

- Unlike encoder-decoder, no autoregressive decoding — direct classification.

Encoder-Only LSTM Diagram

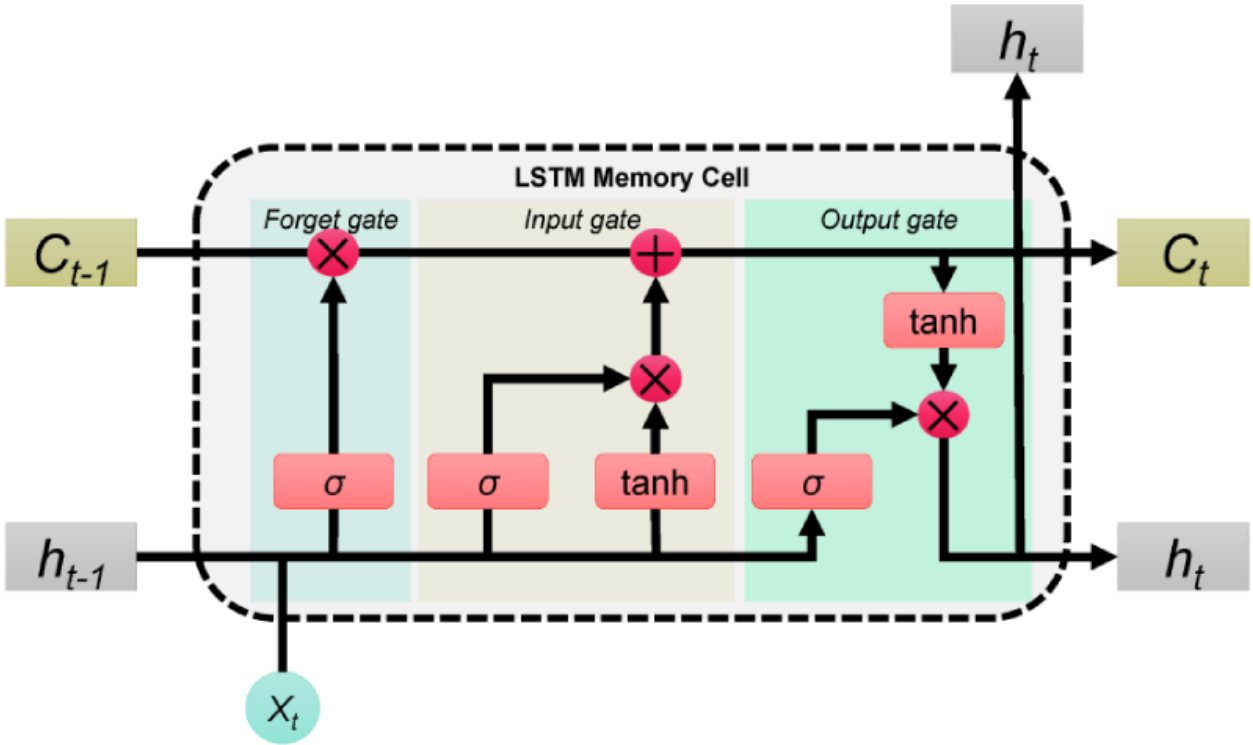
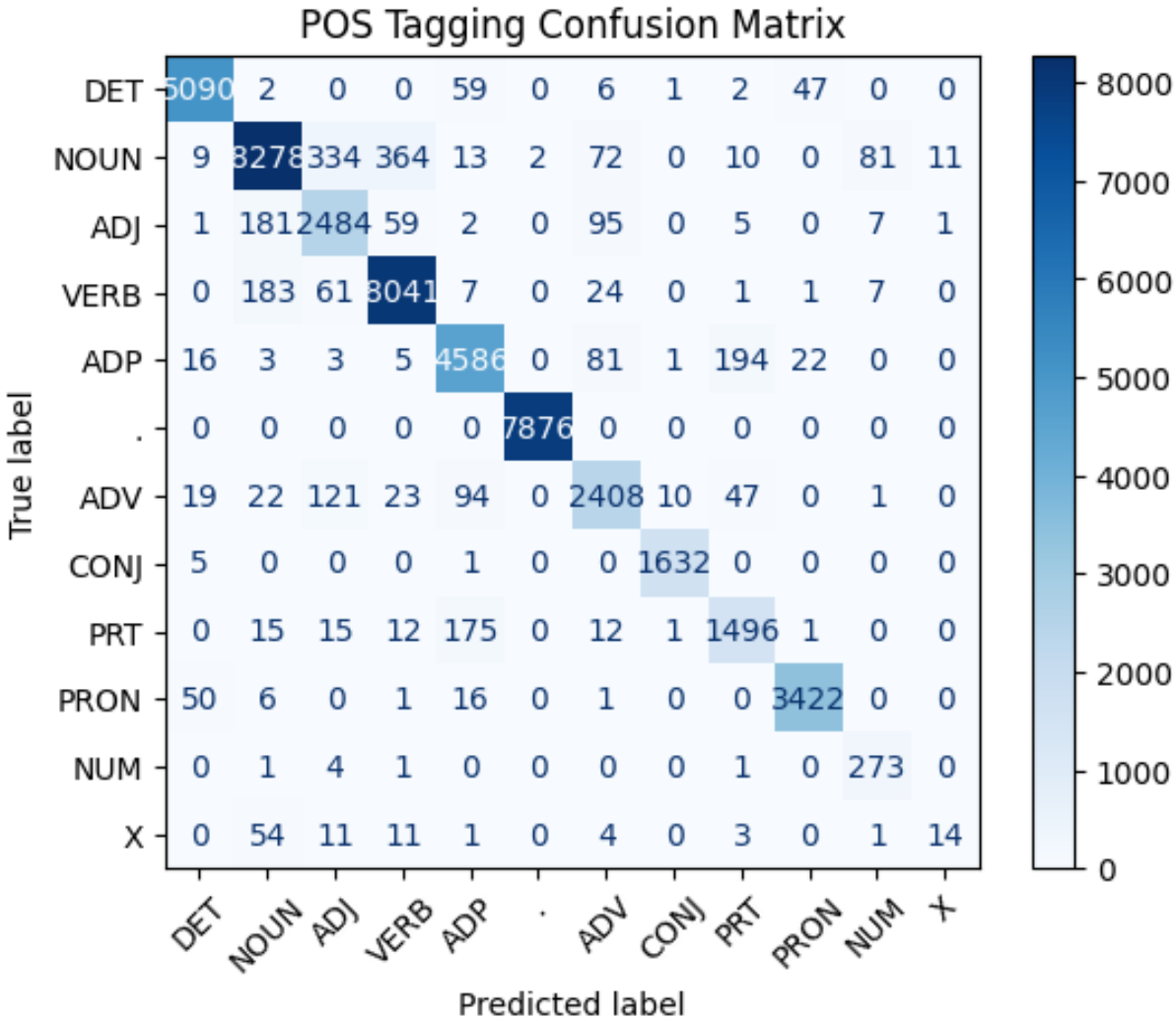


Figure: Schematic of encoder-only LSTM for sequence labeling .

Encoder-Only LSTM Results

- Classification metrics: table .
- Tag-specific confusion matrix analysis.



POS Tagging Performance

- Overall accuracy: **94%**
- Strong performance across most POS tags
- Slight weakness for rare classes (e.g., NUM, X)

Tag	Precision	Recall	F1
DET	0.98	0.98	0.98
NOUN	0.95	0.90	0.92
ADJ	0.82	0.88	0.85
VERB	0.94	0.97	0.95
ADP	0.93	0.93	0.93
.	1.00	1.00	1.00
ADV	0.89	0.88	0.88
CONJ	0.99	1.00	0.99
PRT	0.85	0.87	0.86
PRON	0.98	0.98	0.98
NUM	0.74	0.97	0.84
X	0.54	0.14	0.22
Macro Avg	0.88	0.87	0.87
Weighted Avg	0.94	0.94	0.94

LLM Theory

- Transformer-based models with self-attention.
- Attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$$

- Pretrained on massive corpora $\Rightarrow$ captures syntax/semantics.

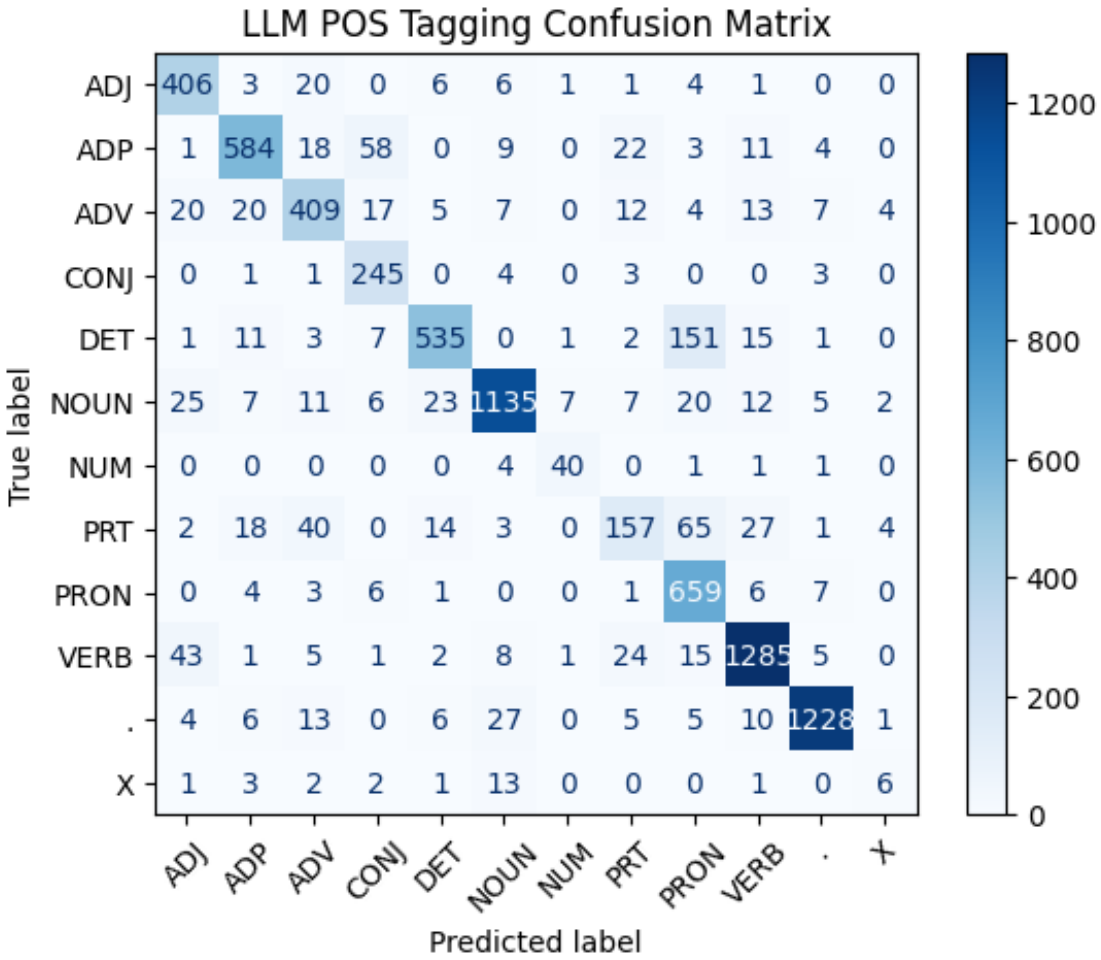
LLM POS Tagging

- Zero-shot system prompting strategy
- Used GPT-5-mini

```
inputs = [" ".join([w for w, _ in sent]) for sent in sentences]
prompt = {
    "role": "user",
    "content": (
        f"You are a POS tagger.\n"
        f"Use only these tags: {', '.join(UNIVERSAL_TAGS)}.\n"
        f"For each input sentence, return a JSON object with key 'results' "
        f"that is a list of items, each item being a list of [word, tag] pairs "
        f"for the corresponding input sentence.\n\n"
        f"Sentences:\n" + "\n".join(f"{i+1}. {s}" for i, s in enumerate(inputs))
    )
}
```

LLM Results

- Overall performance comparable to HMMs
- Captures long-range dependencies but may overgeneralize frequent tags.



POS Tagging Performance (Test Set Evaluation)

- Overall accuracy: **87%**
- Strong performance on frequent tags (NOUN, VERB, DET)
- Lower performance for rare/ambiguous tags (PRT, X)

Tag	Precision	Recall	F1
ADJ	0.81	0.91	0.85
ADP	0.89	0.82	0.85
ADV	0.78	0.79	0.78
CONJ	0.72	0.95	0.82
DET	0.90	0.74	0.81
NOUN	0.93	0.90	0.92
NUM	0.80	0.85	0.82
PRT	0.67	0.47	0.56
PRON	0.71	0.96	0.82
VERB	0.93	0.92	0.93
.	0.97	0.94	0.96
X	0.35	0.21	0.26
Macro Avg	0.79	0.79	0.78
Weighted Avg	0.87	0.87	0.87

Model Comparison

Model	Precision	Recall	F1-score
HMM	0.9473	0.9474	0.9469
LSTM	0.49	0.48	0.46
LLM	0.87	0.87	0.87

Comparative F1-Scores across POS Taggers

POS Tag	HMM	LSTM Enco-Deco	LSTM Encoder	LLM
ADJ	0.88	0.15	0.85	0.85
ADP	0.94	0.31	0.93	0.85
ADV	0.90	0.24	0.88	0.78
CONJ	0.99	0.33	0.99	0.82
DET	0.96	0.42	0.98	0.81
NOUN	0.93	0.48	0.92	0.92
NUM	0.93	0.00	0.84	0.82
PRON	0.96	0.55	0.98	0.82
PRT	0.89	0.19	0.86	0.56
VERB	0.96	0.47	0.95	0.93
.	1.00	0.81	1.00	0.96
X	0.38	0.00	0.22	0.26
Macro Avg	0.89	0.33	0.87	0.78
Weighted Avg	0.95	0.46	0.94	0.87

Analysis of F1-Scores

- **Best Overall:** HMM and LSTM Encoder-only achieve the highest F1 across most POS tags.
- **Weakest:** LSTM Enco-Deco collapses ($<$0.5 F1) on nearly all tags, especially NUM and X.
- **LLM:** Competitive with Encoder LSTM on frequent tags (NOUN, VERB, DET), but underperforms on rare tags (X, PRT, NUM).
- **Frequent vs Rare Tags:** High-frequency classes (. , NOUN, VERB, DET) consistently have strong F1; rare classes (X, NUM, PRT) are difficult across all models.
- **Key Insight:** Structured sequence models (HMM, Encoder-only LSTM) remain state-of-the-art for POS tagging, while LLMs are robust but inconsistent on low-support categories.

Conclusion

- HMM is surprisingly competitive as a baseline.
- LSTM requires more data/pretrained embeddings to shine.
- LLMs leverage pretraining for strong zero-shot tagging.
- Trade-off: Efficiency vs Performance vs Interpretability.