Comparing approaches for Romanian Part of Speech Tagging

Liviu-Ștefan Neacșu-Miclea, ICA 246/2

June 2025

1 Problem Statement

Part-of-Speech (POS) tagging is a fundamental task in Natural Language Processing (NLP), where each word in a sentence is labeled with its grammatical role, such as noun, verb, or adjective. These tags are crucial for syntactic parsing, translation, and other higher-level language tasks.

Formally, given a sentence of words $W = (w_1, w_2, \ldots, w_n)$, the goal is to predict the most likely tag sequence $T = (t_1, t_2, \ldots, t_n)$, with each t_i drawn from a predefined tag set $\mathcal{T}$. In this word, $T = \{ADJ, ADP, ADV, AUX, CCONJ, DET, INTJ, NOUN, NUM, PART, PRON, PROPN, PUNCT, SCONJ, VERB, X\}$, where X is the unknown/unclassifiable tag.

Challenges in POS tagging include:

- **Ambiguity:** Words like *"saw"* can be tagged differently based on context.
- **Unknown words:** The model must handle out-of-vocabulary (OOV) tokens robustly.
- **Morphology:** Word forms vary widely in inflected languages.

This work focuses on building a probabilistic POS tagger using Hidden Markov Models (HMMs), comparing both character-level and word-level representations, with an emphasis on supervised learning using word IDs.

2 Proposed Solution

2.1 Theoretical Aspects

We explore three different models for POS tagging:

- **LSTM**: A Long Short-Term Memory network models the sequence of words, capturing long-range dependencies. Its inputs are sequences of character-encoded words and outputs tag probabilities for each word.
- **CNN-1D**: A 1D Convolutional Neural Network processes each word as a sequence of characters and extracts morphological patterns. The convolutional output is used to classify the word's POS tag.
- **HMM**: A Hidden Markov Model is trained in a supervised fashion with discrete word IDs as observations and POS tags as hidden states. It models transition and emission probabilities directly from the training data.

These models are compared in terms of accuracy and robustness to unknown or ambiguous words.

2.2 Dataset Used in Application

We use the UD Romanian RRT dataset from the Universal Dependencies project. This corpus contains Romanian sentences annotated with morphological and syntactic information, including high standard Part-of-Speech (POS) tags following the Universal POS tag set.

The dataset was preprocessed by:

- Cleaning and normalizing words to retain only ASCII characters and Romanian diacritics.
- Padding sentences and words to fixed lengths for input compatibility with neural models. An additional pad tag was included in the character ids.
- Representing words at the character level for neural models (LSTM, CNN-1D) and as integer IDs for the HMM.
- Encoding target labels as one-hot vectors for supervised training in the case of neural network models.

The final dataset consists of 8,043 training and 729 test sentences, with fixed-length representations of 163 words per sentence and 55 characters per word used for efficient batch processing.

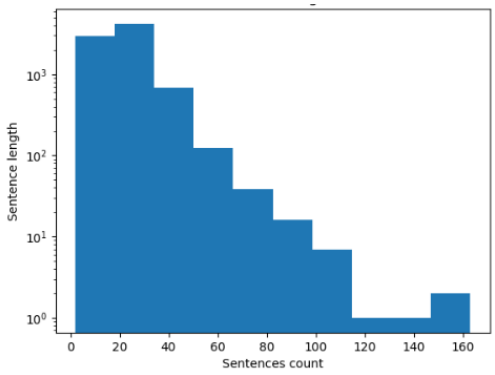

Figure 1: Sentence length distribution

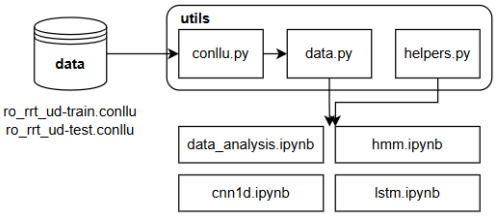

Figure 2: Application diagram

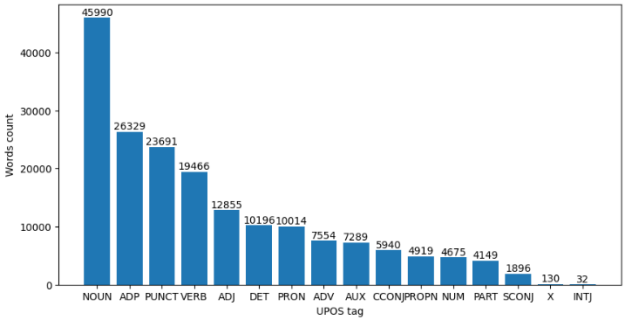

Figure 3: Distribution of PoS tags across the train dataset

Figure 4: Word cloud dataset, after filtering common occurrences like punctuation or conjunctions

3 Application

The practical application consists of Jupyter notebooks running in a Python virtual environment. A diagram of the epplication can be consulted in Figure 2

3.1 Implementation Details

3.1.1 Libraries and Functions

The implementation makes use of both deep learning and probabilistic modeling libraries in Python. The main components are:

- **TensorFlow/Keras**: Used to define and train the deep learning models with ‘Bidirectional(LSTM)‘ and ‘Conv1D‘ layers followed by a ‘TimeDistributed(Dense)‘ output layer. Optimized using the Adam optimizer with categorical cross-entropy loss.

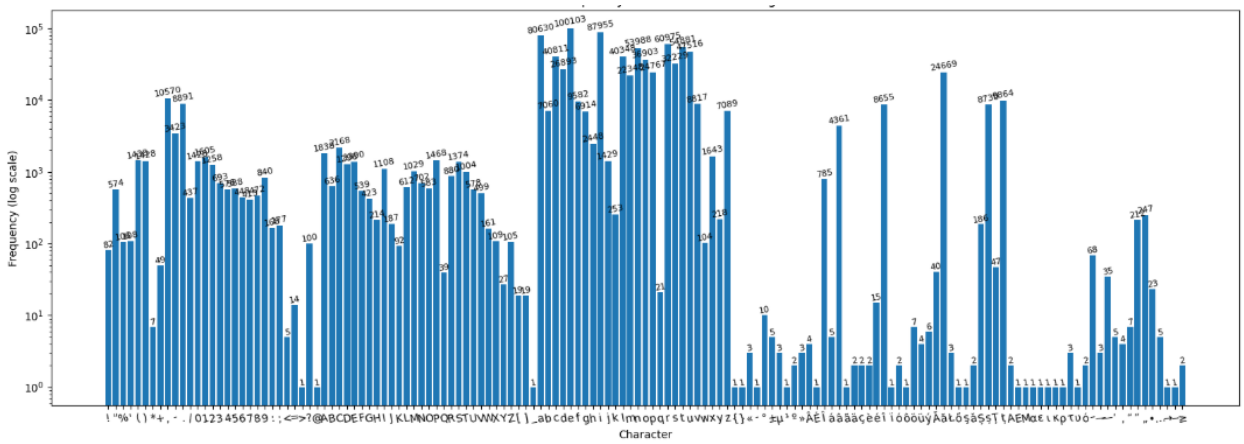

Figure 5: Character frequency in train dataset

- **hmmlearn**: Used to implement the Hidden Markov Model (HMM) with categorical emissions. Transition, emission, and start probabilities were manually initialized using maximum likelihood estimation with Laplace smoothing.
- **scikit-learn**: Used for classification metrics and calculation of confusion matrix.
- **matplotlib/seaborn**: Used for visualizing training curves, confusion matrices, and data distributions.

Data loading and processing are performed by a custom 'Dataset' class. It handles encoding of inputs at both character and word levels, normalizes words, filters valid Romanian characters, and pads sequences to fixed lengths. The class supports multiple encoding modes:

- `chars` mode: Words are represented as sequences of character IDs, enabling character-level modeling for CNNs and LSTMs.
- `word_id` mode: Words are represented as integer indices, used as emissions in the HMM.

3.1.2 Original Contribution

The main contribution is a comparative analysis of the three distinct modeling approaches for PoS tagging on a Romanian language dataset:

- A CNN1D model over character sequences, capturing sub-word patterns.
- A BiLSTM model over character sequences, capturing long-range dependencies.
- A supervised HMM model trained from annotated data, with explicitly estimated transition and emission probabilities.

The 'Dataset' class is implemented from scratch, featuring dynamic vocabulary building, normalization for Romanian-specific characters, categorical encoding, and padding for deep learning compatibility, and can be extended with word vector embeddings. All models were trained and evaluated on the Universal Dependencies Romanian RRT dataset using consistent preprocessing. Additionally, confusion matrices and classification metrics were used to compare model performance, offering insights into their strengths and limitations.

4 Experiments and Results

4.1 Training Setup

For the neural models (Conv1D and Bi-LSTM), we used TensorFlow 2.19 with Keras. All models were trained on a character-level representation of input tokens. The Dataset class was used to extract valid

characters and prepare padded inputs, ensuring consistent input dimensions across samples. For supervised models, the output labels were encoded as one-hot vectors using a predefined Universal POS tag set.

Each model was trained using the Adam optimizer with a learning rate of 0.001. Categorical cross-entropy was used as the loss function, and accuracy was used as the evaluation metric during training. A batch size of 64 and a fixed number of 20 epochs were used for all neural architectures. Data was split into training and validation sets (90%-10%) from the original training dataset (ro_rrt-ud-train.conllu). The model such trained was used for evaluation on the test set (ro_rrt-ud-test.conllu).

The Conv1D model used three stacked 1D convolutional layers with increasing filter sizes (64, 128, 256) and kernel sizes (11, 7, 3), respectively. The Bi-LSTM model used a single bidirectional LSTM layer with 128 hidden units per direction. For both models, the final output layer was a time-distributed softmax classifier over the 17 POS tag categories.

For the probabilistic baseline, a Hidden Markov Model (HMM) was implemented using hmmlearn. Emission, transition, and start probabilities were computed directly from the labeled training sequences using add-1 smoothing. Viterbi decoding was used to generate predictions on the test set.

4.2 Results

We evaluated the models on the full test set (16,324 tokens) using standard metrics: precision, recall, and F1-score, computed per class as well as macro and weighted averages. The results are summarized in Table 1.

Table 1: Per-label performance metrics (Precision, Recall, F1-score) for each model on the test set.

Label	**Conv1D**			**Bi-LSTM**			**HMM**		
	P	R	F1	P	R	F1	P	R	F1
ADJ	0.48	0.14	0.21	0.52	0.37	0.43	0.42	0.53	0.47
ADP	0.55	0.78	0.64	0.68	0.82	0.74	0.62	0.85	0.72
ADV	0.34	0.06	0.11	0.38	0.21	0.27	0.45	0.21	0.29
AUX	0.26	0.11	0.15	0.39	0.45	0.42	0.28	0.16	0.21
CCONJ	0.88	0.70	0.78	0.77	0.79	0.78	0.90	0.75	0.82
DET	0.21	0.13	0.16	0.36	0.16	0.22	0.24	0.15	0.19
INTJ	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NOUN	0.45	0.86	0.59	0.57	0.78	0.66	0.59	0.72	0.65
NUM	0.41	0.09	0.15	0.58	0.15	0.24	0.30	0.13	0.19
PART	0.62	0.27	0.38	0.64	0.48	0.55	0.64	0.47	0.54
PRON	0.37	0.09	0.15	0.43	0.35	0.39	0.34	0.28	0.31
PROPN	0.00	0.00	0.00	0.28	0.04	0.06	0.59	0.07	0.12
PUNCT	0.78	0.95	0.86	0.83	0.96	0.89	0.79	0.95	0.86
SCONJ	0.33	0.10	0.16	0.49	0.37	0.42	0.60	0.20	0.30
VERB	0.49	0.17	0.25	0.57	0.47	0.52	0.59	0.42	0.49
X	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Macro Avg	0.36	0.26	0.27	0.44	0.38	0.39	0.46	0.37	0.38
Weighted Avg	0.48	0.52	0.45	0.58	0.61	0.57	0.56	0.58	0.55

The Bi-LSTM model outperformed both the Conv1D CNN and the HMM in terms of all key metrics, especially in F1-score and weighted precision. Notably, the HMM achieved comparable results to the Bi-LSTM on several high-frequency tags like ADP, NOUN, and PUNCT, but it struggled significantly with low-frequency or ambiguous tags (e.g., PROPN, INTJ).

In contrast, the Conv1D model demonstrated fast convergence and strong performance on structurally simple tags (like PUNCT and CCONJ), but it underperformed on morphosyntactically complex categories (e.g., VERB, AUX, and PRON), likely due to its lack of sequence memory.

Overall, the Bi-LSTM architecture provides the most balanced trade-off between precision and recall, particularly excelling in capturing longer dependencies critical for accurate POS tagging.