# Analysis of Parts of Speech (POS) to differentiate the Statistical learning and Deep learning

Nishvi Pradipkumar Patel
*Department of Mathematical Sciences*
*Stevens Institute of Technology*
Hoboken, USA
npatel33@stevens.edu

Charmilkumar Vijaykumar Patel
*Department of Mathematical Sciences*
*Stevens Institute of Technology*
Hoboken, USA
cpatel6@stevens.edu

Yash J Patel
*Department of Computer Engineering*
*Stevens Institute of Technology*
Hoboken, USA
ypatel26@stevens.edu

***Abstract*—The project is based on Part-of-Speech (POS) tagging, a central task in Natural Language Processing (NLP) in which every word of a sentence is assigned its respective grammatical category. The objective is to investigate and compare two approaches, one statistical approach and the other deep learning, to accurately POS tagging. We used logistic regression because the statistical method initially used is simple to use and understand. The model performed well on the 370k English words corpus dataset of 370,100 words along with their POS when tested. In the second stage, a deep learning model is used and tested. The comparison of the two approaches in accuracy, performance and usability is presented in the final report, offering insight into the advantages and disadvantages of each approach in real POS tagging applications.**

## I. Introduction

Part-of-Speech (POS) tagging plays a vital role in Natural Language Processing (NLP) by assigning grammatical categories, such as noun, verb, or adjective, to each word in a sentence. It serves as a foundational step for more complex language understanding tasks such as syntactic parsing, information retrieval, and machine translation. The goal of this project is to explore and evaluate different machine learning approaches for POS tagging using a comprehensive corpus of 370,100 English words.

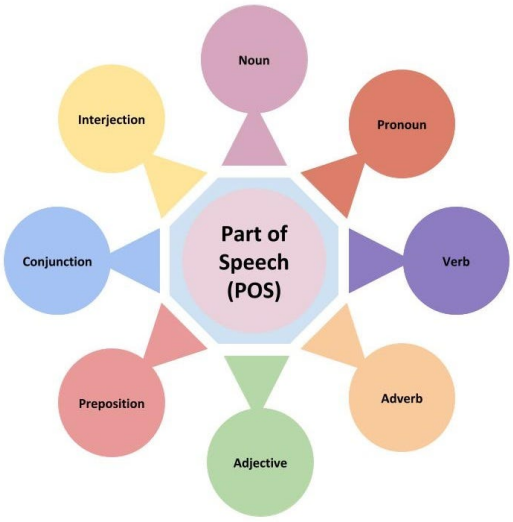


Fig. 1. Components of Parts of Speech (POS)

This study implements both statistical and deep learning methods to investigate their effectiveness in tagging accuracy. Logistic Regression is first used as a baseline model due to its ease of implementation, interpret ability, and ability to serve as a reliable benchmark. Enhancements such as Principal Component Analysis (PCA) and Word2Vec embeddings are also applied to examine the effect of feature dimensionality and semantic representation on performance. In the second phase, a long-short-term memory (LSTM) model is developed, a form of recurring neural network (RNN), to capture temporal dependencies in sequential data, providing a deep learning perspective.

Through this comparative analysis, the project aims to highlight the practical strengths and trade-offs between classical machine learning and modern deep learning techniques in real-world POS tagging applications.

## II. Related Work

POS tagging has been studied using many different methods over the years. Some approaches rely on simple models, while others employ more complex deep learning techniques. We looked at three key papers that helped shape our approach.

1. *Logistic Regression for POS Tagging*: In the paper by Pranckevičius and Marcinkevičius (2016) [1], the authors applied logistic regression for multiclass POS tagging. They demonstrated that even a basic model, such as Logistic regression, can perform effectively when appropriate features are used. However, such models often struggle with capturing deeper linguistic patterns because of their limited representational power.

2. *Deep Learning for Bengali POS Tagging*: Patoary et al. (2020) [2] used deep learning to create an automated POS tagger for the Bengali language. Their neural network model successfully managed complex linguistic structures and improved tagging accuracy. Despite the benefits, the method requires large-scale annotated data and significant computational resources.

3. *Comparison of ML and DL approaches*: Chiche and Yitagesu (2022) [3] provided a comprehensive review of machine learning and deep learning techniques for POS tagging. Their findings showed that deep learning models typically achieve higher accuracy, but traditional models such as logistic regression are more interpretable and computationally efficient.

The study also pointed out ongoing challenges in low-resource settings and domain adaptation.

One significant difference in our project is that, unlike the existing research that is generally interested in POS tagging at the sentence level, our project is slightly more interested in the word level where each word is tagged independently. This design choice not only simplifies the input to the model, but also allows for an intense test of the classification ability of each method regardless of sentence structure. This word-level strategy differentiates our research and presents a new orientation in POS tagging research.

## III. Our Solution

Our approach to solving the POS tagging problem involves a structured pipeline of steps. It started with data preprocessing and exploratory analysis, followed by feature engineering, building and evaluating models, and then comparing results. To keep things organized, we grouped these steps into three main sections: Dataset Description, Machine Learning Models, and Implementation Details.

### A. Description of Dataset

The dataset employed for this research was taken from Kaggle, which is a commonly used data science platform. It consists of 307,100 instances and 7 features in which every instance refers to an English word and has the corresponding part-of-speech tags. POS tags consist of classifications like nouns, verbs, adjectives, adverbs, pronouns, conjunctions, prepositions, and interjections utilizing standard POS tag codes (e.g., NN, VB, JJ, RB). Tags are categorically encoded.

One column contains commonly used English words, while the remaining columns represent extracted linguistic features such as word length, syllable count, prefixes/suffixes, punctuation presence, and frequency. These features are critical for training accurate classification models.

To better understand the data set and extract meaningful patterns, we performed data visualization and feature analysis. This included examining the distribution of word lengths, the frequency of punctuation, syllable counts, and the distribution of the POS tag, which revealed essential insights for model development.

1. *Visualizations:* The visualizations representing certain features are:

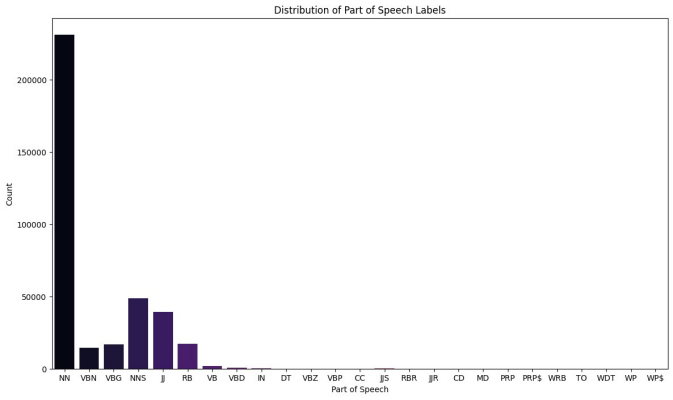


Fig. 2. Distribution of POS Labels

The chart shows that POS tags such as NN, VB, and VBG appear far more frequently than others, indicating a class imbalance in the dataset. This helps us to understand which tags dominate and where the model might need extra attention.

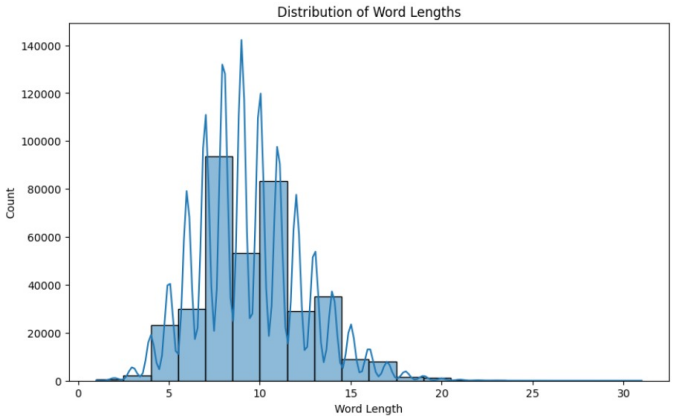


Fig. 3. Distribution of Word Lengths

The distribution of word length is almost normal but slightly right-skewed, which means there are a few longer words in the data. This feature is useful for part-of-speech tagging because word length is sometimes related to grammar. For example, longer words are used more frequently as nouns or adjectives.

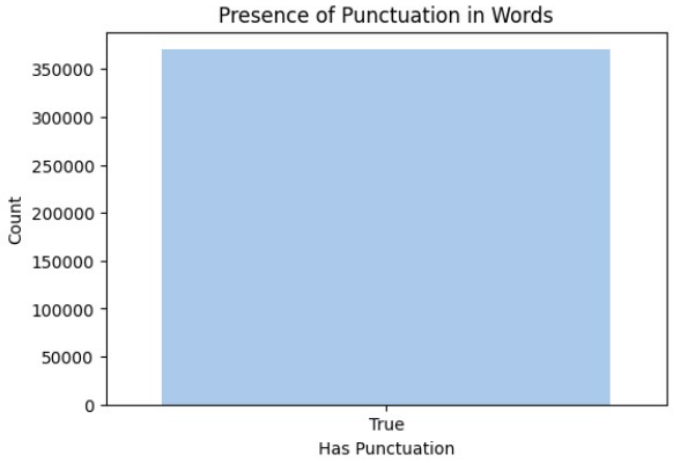


Fig. 4. Presence of Punctuation in Word

Shows that a large number of tokens are identified as having punctuation. This includes things like periods, commas, quotation marks, and other symbols. Punctuation marks are important in sentence structure (like showing sentence boundaries or separating clauses), so recognizing them is key in part-of-speech tagging, especially when it comes to telling apart terminal punctuation from other grammatical elements.

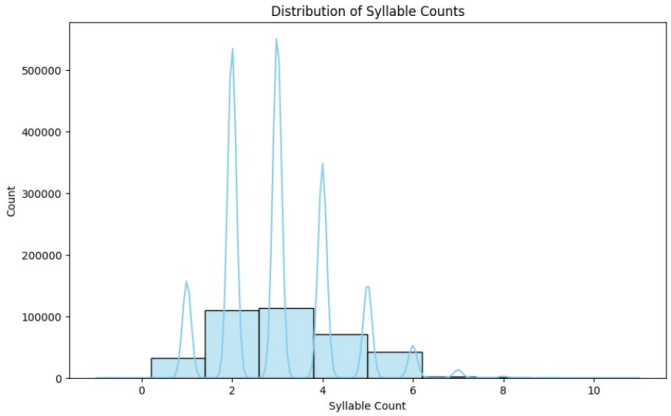


Fig. 5. Distribution of syllable counts

Most words have 2 to 3 syllables and only a few have more than 5. This is helpful because the number of syllables can give us a clue about the type of word, such as whether it is a noun or an adjective.

| | word | pos_tag | Word_Length | Vowel_Count | Consonant_Count | Syllable_Count | Word_Frequency | Prefix_un_Length | Suffix_ing_Length | Has_Punctuation |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | aa | NN | 2 | 2 | 0 | 1 | 1 | 0 | 0 | True |
| 1 | aaa | NN | 3 | 3 | 0 | 1 | 1 | 0 | 0 | True |
| 2 | aah | NN | 3 | 2 | 1 | 1 | 1 | 0 | 0 | True |
| 3 | aahed | VBN | 5 | 3 | 2 | 1 | 1 | 0 | 0 | True |
| 4 | aahing | VBG | 6 | 3 | 3 | 1 | 1 | 0 | 3 | True |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 370095 | zwinglianism | NN | 12 | 4 | 8 | 3 | 1 | 0 | 0 | True |
| 370096 | zwinglianist | NN | 12 | 4 | 8 | 3 | 1 | 0 | 0 | True |
| 370097 | zwitter | NN | 7 | 2 | 5 | 2 | 1 | 0 | 0 | True |
| 370098 | zwitterion | NN | 10 | 4 | 6 | 3 | 1 | 0 | 0 | True |
| 370099 | zwitterionic | NN | 12 | 5 | 7 | 4 | 1 | 0 | 0 | True |

370100 rows × 10 columns

Fig. 6. Feature Engineering

Here, we find the length of words, the count of vowels and consonants, frequency of words, prefix-suffix lengths, etc.

### B. Machine Learning Algorithms

Our study employs a combination of traditional and deep learning models to address the part-of-speech tagging task and also to analyze performance across various modeling strategies. We have utilized Logistic Regression, Logistic Regression with Principal Component Analysis (PCA) and Logistic Regression with Word2Vec, and moving forward we are going to implement a deep learning model based on Long Short-Term Memory (LSTM) networks.

1. *Logistic Regression:* It was employed as the underlying statistical model due to its ability to interpret, efficiency and suitability for multiclass classification tasks. Logistic Regression performs best in a scenario where well-engineered features are used as input, along with a decision boundary line that is good enough to differentiate between the classes.

2. *Logistic Regression with PCA:*

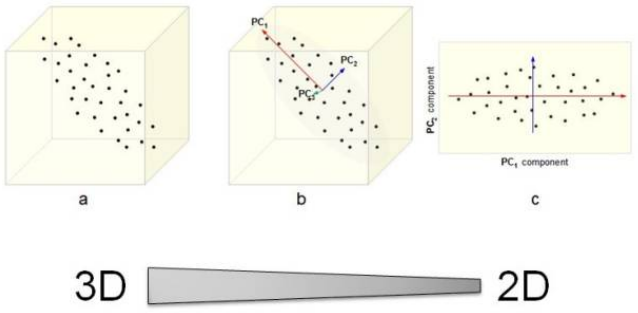


Fig. 7. Overview of Principal Component Analysis (PCA)

In this project, Principal Component Analysis (PCA) was employed as a dimensionality reduction technique to simplify the high-dimensional feature space derived from the 370k word corpus. By projecting the original features onto a lower-dimensional space that retains the most significant variance (as shown in the figure), PCA helps eliminate redundancy and noise in the data. This not only reduces computational complexity, but also enhances model generalization. When applied before training the Logistic Regression model, PCA improves performance by focusing on the most informative aspects of the input data, making it especially valuable for efficient and accurate part-of-speech tagging at the word level.

3. *Logistic Regression with Word2Vec:*

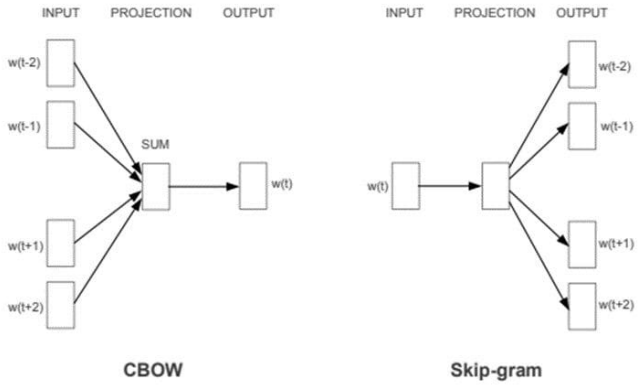


Fig. 8. Overview of Word2Vec

Word2Vec is a powerful embedding technique that transforms words into dense vector representations based on their contextual usage in large text corpora. In this project, it was used to capture the semantic relationships between individual words in the 370k word POS tagged dataset. By representing each word as a continuous vector, Word2Vec enabled the model to understand and leverage the underlying meanings and similarities between words, which is crucial for accurate part-of-speech tagging at the word level.

4. *LSTM-Based Deep Learning Model:*

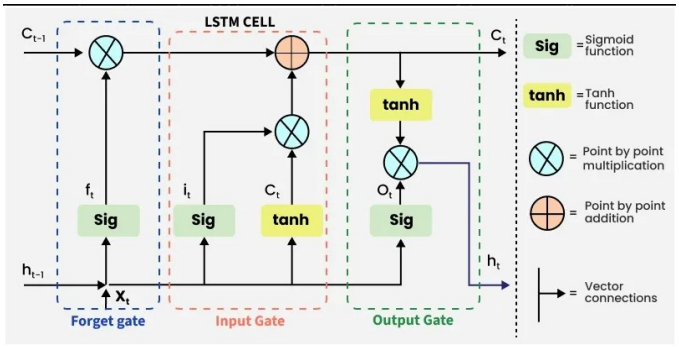


Fig. 9. LSTM Model

Long Short Term Memory (LSTM) networks are a complex deep learning structure for modeling sequential data. They can maintain long range dependencies, which is of value in modeling linguistic structure in natural language processing tasks such as POS tagging. In this study, we applied an LSTM-based neural network to address POS tagging, taking advantage of its potential for modeling contextual and sequential relationships between words.

### C. Implementation Details

In the first phase of our project, we implemented Logistic Regression models using three different feature extraction techniques. In addition, we constructed an LSTM-based neural network for sequential modeling. The steps involved in building each model, along with the corresponding accuracy evaluations, are detailed in the following sections.

**Logistic Regression Implementation Steps:**

1. *Dataset Loading:* The dataset was loaded from a CSV file containing two columns: individual English words and their respective POS tags. These data served as the foundation for both feature extraction and supervised learning.

2. *Feature Engineering:* Several features were extracted from each word to provide meaningful information for classification.

- *Word Length:* Number of characters in the word.
- *Vowel and Consonant Counts:* Number of vowels and consonants in each word.
- *Syllable Count:* Calculated using a rule-based method that counts vowel clusters
- *Word Frequency:* The frequency of each word in the dataset, indicating its commonness.
- *Prefix and Suffix Length:* Binary-length features for common affixes such as 'un' and 'ing'.
- *Punctuation Presence:* A binary feature indicating whether a word contains any punctuation.

3. *Outlier Detection and Handling:* Outlier analysis was performed on word length, vowel count, and consonant count. A z score method was used to identify values that lie more than three standard deviations from the mean. Detected outliers were replaced with the mean value of the respective feature to reduce skewness and maintain the robustness of the model.

4. *Data Preparation*: POS tag labels were encoded using *LabelEncoder* to convert categorical labels into numerical form.

5. *Model Training:* The Logistic Regression model was implemented using Scikit-learn's *LogisticRegression* class. It was configured for multiclass classification using the *multinomial* option and optimized with the *lbfgs* solver, also, the maximum number of iterations was set to *500* to ensure model convergence.

6. *Evaluation:* The trained model was evaluated in the test set considering accuracy, precision, recall, and F1-score. A detailed classification report was generated.

This structured approach ensured that both the data and model were prepared in a consistent and systematic manner, allowing Logistic Regression to serve as a strong baseline for comparison with PCA-enhanced classifiers and deep learning networks.

**Logistic Regression with PCA: Implementation Steps:**

1. *Feature Scaling:* Before performing PCA, the feature matrix *X* was normalized using the assistance of *StandardScaler* so that all features contributed equally to the calculation of the principal component.

2. *Dimensionality Reduction:* Principal Component Analysis was applied to the set of scaled features to reduce its dimension. The dimension was reduced to two.

3. *Model Training:* A Logistic Regression model was then fitted to the two-dimensional PCA transformed data. The model parameters used default hyperparameters, but with *max_iter = 500* to ensure convergence.

4. *Prediction and Evaluation:* The trained model was used to predict the test set. Performance was evaluated using standard classification metrics: accuracy, precision, recall, and F1-score. The performance was compared with that obtained using the full-feature Logistic Regression model.

5. *PCA Visualization of POS Predictions* To visually assess the separation of predicted POS tags, a two-dimensional PCA projection of the test data was plotted. The predictions generated by the Logistic Regression model using PCA-transformed features were mapped to their corresponding POS tag labels.

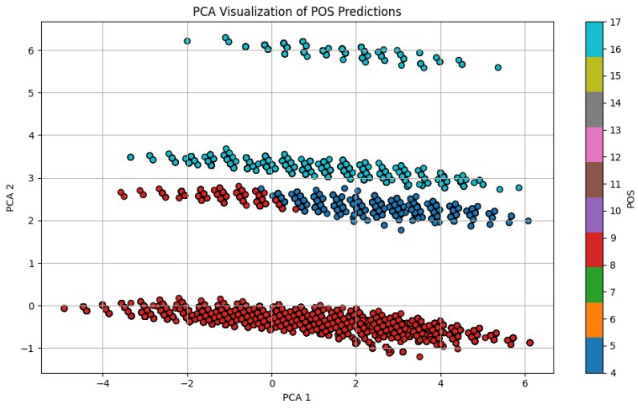


Fig. 10. PCA Prediction Plot

**Logistic Regression with Word2Vec: Implementation Steps:**

1. *Text Preprocessing:* A list of English stopwords is retrieved using the NLTK library to filter out non-informative words. A *preprocess_text* function is defined to:

- Convert each word to lowercase.
- Tokenize the word.
- Remove non-alphabetic tokens and stopwords.

2. *Training the Word2Vec Model:* A Word2Vec model is trained on the tokenized word. The model learns semantic vector representations of words based on their surrounding context.

3. *Vector Representation of Words:* A helper function *get_word_vector* is defined to retrieve the trained Word2Vec vector for a given word or return a zero vector if the word is out-of-vocabulary (OOV). Another function, *average_word_vectors*, averages the vectors of all tokens for each original word.

4. *Feature Construction:* The Word2Vec embeddings are combined with additional linguistic features such as:

- Word length.
- Vowel count.
- Consonant count.

5. *Model Training:* The feature matrix and the corresponding POS labels are divided into *80%* training and *20%* test sets. A Logistic Regression model is trained using the *multinomial* loss function and *lbfgs* optimization solver.

6. *Prediction and Evaluation:* Predictions are made in the test set. The model is evaluated using standard classification metrics: Accuracy, Precision, Recall, and F1 Score. A detailed classification report is also generated.

**Deep Learning with LSTM + Glove Embedding: Implementation Steps:**

1. *Word Tokenization:* The input data consisted of individual words stored in the column *word*. A Keras *Tokenizer* was used to convert each word into a unique integer index. OOV tokens were replaced with a special placeholder *OOV*. The resulting sequences were padded to a uniform length of 10 using the *pad_sequences* function.

2. *Training the Word2Vec Model:* GloVe embeddings with a dimensionality of 100 were loaded using Gensim's *api.load()* method. An embedding matrix was created by mapping each word from the tokenizer's vocabulary to its corresponding GloVe vector. This matrix was then used to initialize the embedding layer of the LSTM model.

3. *Preparing Labels and Data Split:* The target labels (POS tags) were already encoded using a *LabelEncoder* and stored in the variable *y*. The data set was then split into training and test sets in an 80:20 ratio using the *train_test_split* function.

4. *Model architecture*
A bidirectional LSTM model was constructed using TensorFlow's *Sequential* API:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 10, 100)           37010200

 bidirectional (Bidirection  (None, 10, 256)           234496
 al)

 dropout (Dropout)           (None, 10, 256)           0

 bidirectional_1 (Bidirecti  (None, 128)               164352
 onal)

 dropout_1 (Dropout)         (None, 128)               0

 dense (Dense)               (None, 25)                3225

=================================================================
Total params: 37412273 (142.72 MB)
Trainable params: 402073 (1.53 MB)
Non-trainable params: 37010200 (141.18 MB)
```

Fig. 11. LSTM Model Architecture

- *Embedding Layer:* Initialized with the pretrained GloVe embedding matrix.
- *First Bidirectional LSTM Layer:* Contains 128 units and returns sequences to feed into the next LSTM layer.
- *Dropout Layer:* Applied with a dropout rate 50% to prevent overfitting.
- *Second Bidirectional LSTM Layer:* Contains 64 units and outputs the final encoded representation of the sequence.
- *Final Dense Layer:* A fully connected layer with *softmax* activation to predict one of the POS tag classes.

5. *Model training:* The model was compiled using the *sparse_categorical_crossentropy* loss function, the *adam* optimizer, and *accuracy* as evaluation metric. Training was carried out with a batch size of 64, for up to 20 epochs, with early stopping based on validation loss to prevent overfitting.

6. *Model Evaluation:* Final predictions were obtained using *argmax* over the probability outputs, and the model was evaluated using a *classification_report*.

**Confusion Matrix Visualization**

To evaluate the classification performance of each model across all POS tags, confusion matrices were generated for the following four models:

- *Logistic Regression:* The actual and predicted POS tag labels were first extracted, and a confusion matrix was calculated using the *confusion_matrix* function. The matrix was visualized with *ConfusionMatrixDisplay*. The resulting matrix provided a detailed view of true versus predicted tags.
- *Logistic Regression with PCA-Transformed Features:* Similar steps were followed, the only difference being the features used; these were reduced using PCA. The confusion matrix helped assess how the reduction in dimensionality impacted classification performance.
- *Logistic Regression with Word2Vec Embeddings:* POS predictions based on word embeddings were compared with actual tags using the confusion matrix. This allowed a side-by-side evaluation of semantic vector-based learning versus other models.

- *LSTM Model:* For the LSTM model, the confusion matrix was calculated directly using integer-labeled predictions and all POS classes provided by *LabelEncoder*. The matrix was rendered similarly, with all tag classes visualized to identify strengths and weaknesses in sequence-based predictions.

Each confusion matrix was displayed using a blue color map and labeled axes, with cell values shown, and font sizes adjusted for readability. The titles reflected the feature extraction or model type used, aiding in comparative interpretation.

**Prediction and Result Compilation**

For each model, the prediction of the POS tag was made in the test set and decoded from their integer encoded format back to human-readable labels using the *LabelEncoder*. These results included three columns: the word, its actual POS tag, and the predicted POS tag. To provide a sample view of the model performance, a random subset of 20 predictions was printed for each case. This provided interpretable insights into the behavior of the model.

## IV. Comparison

To evaluate the effectiveness of different approaches to POS tagging, we implemented and compared four models. Logistic Regression, Logistic Regression with PCA, Logistic Regression with Word2Vec, and a Bidirectional LSTM model. The comparison was based on confusion matrices and word-level predictions.

**Confusion Matrix Analysis**

*1. Logistic Regression*

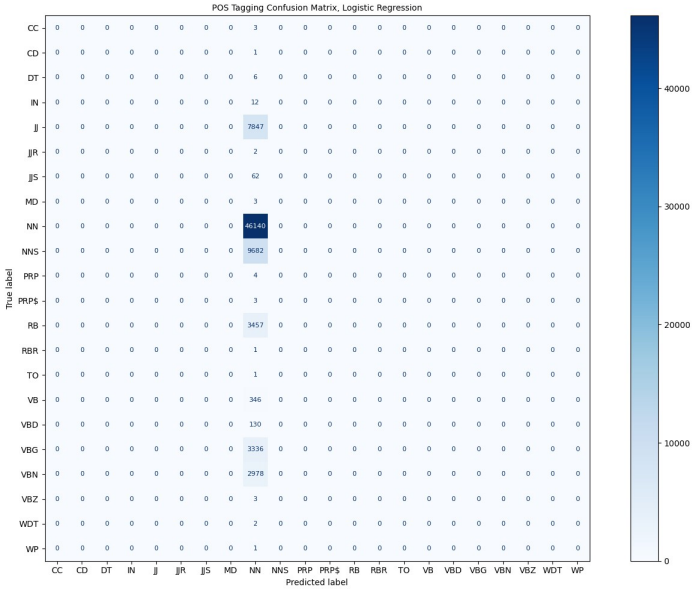


Fig. 12. Confusion Matrix of Logistic Regression

The logistic regression model achieved high accuracy for frequent POS classes such as

- *NN* (Noun, singular): 46,140 correctly predicted
- *NNS* (Noun, plural): 9,682 are identified as NN, instead of NNS
- *JJ* (Adjective) and *RB* (Adverb) also recognized as NN

However, the model also showed limitations in distinguishing between verb forms (e.g., *VB*, *VBD*, *VBG*, *VBN*) and low-frequency tags like *WP$*, *WRB*, and *TO*, which were often misclassified or not predicted at all.

*2. Logistic Regression with PCA:*

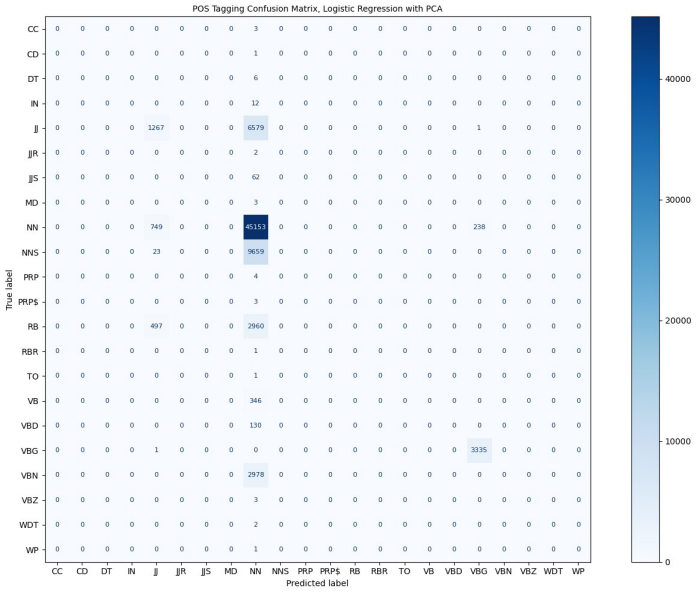


Fig. 13. Confusion Matrix of Logistic Regression with PCA

PCA was applied to reduce the feature dimensionality. This resulted in the following.

- Slightly reduced performance for high-frequency tags (e.g., *NN* = 45,153)
- Noticeable confusion among similar POS categories like *RB*, *JJ* and *NN*, *NNS*
- PCA compressed the feature space, which improved the computation, but caused minor degradation in tag-level precision, especially for mid-frequency POS tags.

*3. Logistic Regression with Word2Vec:*

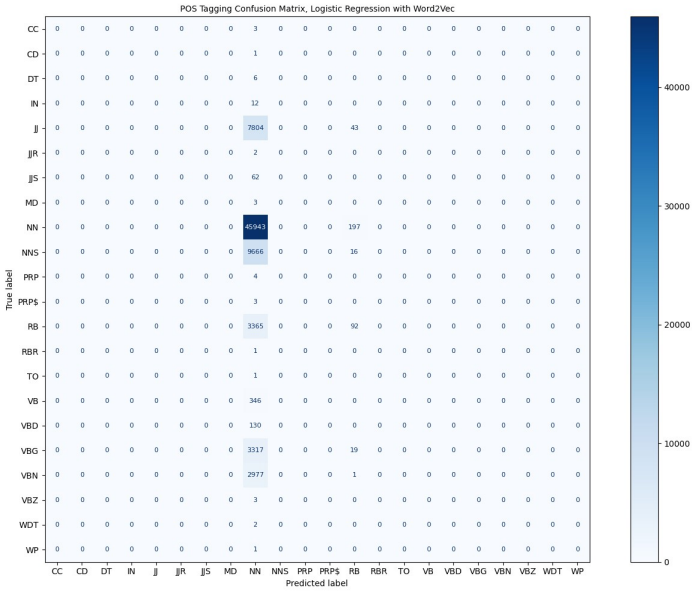


Fig. 14. Confusion Matrix of Logistic Regression with W2V

Incorporating semantic word embeddings with logistic regression helped improve contextual understanding

- *NN* and *NNS* were still accurately predicted (e.g., *NN* = 45,943)
- Improved tagging of *VBG* and *RB* compared to the baseline
- Minor confusion between *JJ* and *NN* persisted

Word2Vec allowed the model to better capture word semantics, leading to improved generalization for unseen or rare words.

*4. LSTM-Based Model:*

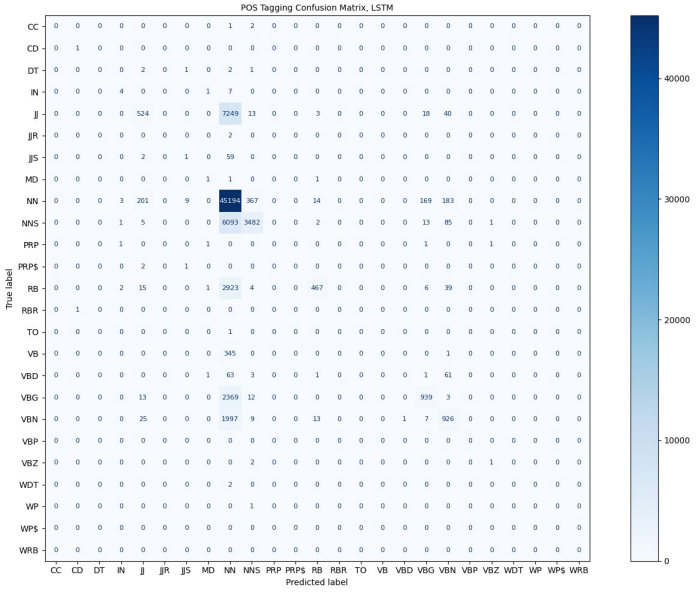


Fig. 15. Confusion Matrix of LSTM-Based Model

The LSTM model, which takes advantage of the sequential structure of language, provided the following

- The most flexible classification of *NN* = 45,194 and *RB* = 2,923
- Strong recognition of verb categories and adjectives
- Reduced misclassification between related tags due to context awareness
- Some confusion still remained in tag boundaries, particularly for OOV words or noisy inputs

## Model Evaluation Metrics Comparison

| Metric | Logistic Regression | Logistic Regression + PCA | Logistic Regression + Word2Vec | LSTM Model |
|---|---|---|---|---|
| Accuracy | 0.623 | 0.672 | 0.622 | 0.696 |
| Precision | 0.389 | 0.509 | 0.4 | 0.718 |
| Recall | 0.623 | 0.672 | 0.622 | 0.696 |
| F1 Score | 0.479 | 0.563 | 0.48 | 0.628 |

Fig. 16. Model Evaluation Metrics Table

Four standard evaluation metrics are reported for each model: Accuracy, Precision, Recall, and F1 Score.

The LSTM model outperformed all the others in every metric, achieving the highest accuracy (0.696), precision (0.718), recall (0.696), and F1 score (0.628), indicating its superior ability to model contextual and sequential dependencies in language.

Logistic Regression with PCA showed a noticeable improvement over baseline, particularly in precision (0.509) and F1 score (0.563), demonstrating the benefit of dimensionality reduction for generalization.

Logistic Regression with Word2Vec slightly improved over baseline in precision (0.400) and F1-score (0.480), thanks to its use of semantic word embeddings.

The baseline Logistic Regression model performed reasonably well for recall (0.623) and precision (0.623), but showed lower precision (0.389), indicating frequent misclassification of certain POS tags.

This comparison highlights the value of advanced representation (Word2Vec) and sequential modeling (LSTM) in improving POS tagging performance, especially when precision and F1-score are critical.

## Word-Level Prediction Examples

In addition to aggregate confusion matrices and evaluation metrics, sample outputs illustrate how each model performed on unseen words:

*Logistic Regression:*

Predictions (Logistic Regression Model):

| Word | True_POS | Predicted_POS |
|---|---|---|
| mercery | NN | NN |
| sticky | NN | NN |
| metachromatinic | NN | NN |
| mousoni | NN | NN |
| novity | NN | NN |
| acroterium | NN | NN |
| disapprobation | NN | NN |
| hypotaxic | NN | NN |
| popsicle | NN | NN |
| dicalcium | NN | NN |
| piggy | NN | NN |
| underbraced | JJ | NN |
| quirked | NNS | NN |
| alternaria | NNS | NN |
| touret | NN | NN |
| aminobarbituric | NN | NN |
| cuir | NN | NN |
| metallics | NNS | NN |
| defying | VBG | NN |
| coalpits | NNS | NN |

Fig. 17. Prediction Examples Logistic Regression

Lacked contextual understanding; Most rare or unknown words defaulted to NN (e.g., *disapprobation*, *defying*).

*PCA Model:*

Predictions (PCA Model):

| Word | True_POS | Predicted_POS |
|---|---|---|
| digrediency | NN | NN |
| notepad | NN | NN |
| pyranoses | NNS | NN |
| karagan | NN | NN |
| uniequivalent | NN | JJ |
| uncircumscriptible | JJ | JJ |
| athletics | NNS | NN |
| randing | VBG | VBG |
| apotelesm | NN | NN |
| turlupin | NN | NN |
| coenure | NN | NN |
| nearctic | JJ | NN |
| rappist | NN | NN |
| pressfat | NN | NN |
| nonheroicalness | NN | NN |
| mongery | NN | NN |
| plumpness | NN | NN |
| unanatomised | JJ | JJ |
| steuben | NN | NN |
| karats | NNS | NN |

Fig. 18. Prediction Examples PCA Model

Performed similarly to the baseline, with some improvement in adjectives (e.g., *uncircumscriptible* tagged as JJ correctly).

*Word2Vec Model:*

Predictions (Word2Vec Model):

| Word | True_POS | Predicted_POS |
|---|---|---|
| thermometer | NN | NN |
| bedeswomen | NNS | NN |
| gentian | JJ | NN |
| veg | NN | NN |
| etiolate | NN | NN |
| curassow | NN | NN |
| procrastinatingly | RB | NN |
| soothsaid | NN | NN |
| slumlords | NNS | NN |
| overglaze | NN | NN |
| scleroblastema | NN | NN |
| essayists | NNS | NN |
| elinvar | NN | NN |
| geochronic | NN | NN |
| alchemize | VB | NN |
| nonsubstantially | RB | RB |
| adjuratory | NN | NN |
| mnemic | NN | NN |
| rubeolar | NN | NN |
| curacaos | NN | NN |

Fig. 19. Prediction Examples Word2Vec Model

Better handled adverbs and verbs using semantic embeddings (e.g., *procrastinatingly* tagged as RB, *alchemize* as VB).

*LSTM Model:*

Predictions LSTM:

| Word | True_POS | Predicted_POS |
|---|---|---|
| wifing | VBG | NN |
| quarterlies | NNS | NNS |
| trappean | NN | NN |
| oversees | NNS | NNS |
| roister | NN | NN |
| bawdstrot | NN | NN |
| tinc | NN | NN |
| craniorhachischisis | NN | NN |
| satirising | VBG | VBG |
| untrespassed | JJ | NN |
| curlews | NNS | NNS |
| felonwood | NN | NN |
| peternet | NN | NN |
| hemibathybian | NN | NN |
| unriddling | VBG | NN |
| nonleviable | JJ | NN |
| jocosities | NNS | NN |
| encrusting | VBG | VBG |
| schmitz | NN | NN |
| exhibitions | NNS | NNS |

Fig. 20. Prediction Examples LSTM-Based Model

Demonstrated strongest performance with correct POS tagging across adjectives (e.g., *untrespased*), gerunds (e.g., *unriddling*), and rare nouns (e.g., *craniorhachischisis*).

**Comparison with Existing Literature**

*Pranckevičius & Marcinkevičius (2016) [1]:*

This study evaluated classical machine learning models (such as logistic regression, Naïve Bayes, SVM) for multiclass sentiment classification using Twitter data. Their findings highlighted:

Logistic regression offers reasonable performance and interpretability, but struggles with complex language structure and nuanced semantic understanding.

Our baseline logistic regression model achieved an accuracy of 0.623, which is consistent with their assessment of classical models. However, by introducing PCA and Word2Vec, we observed modest performance gains, up to 0.672 accuracy and 0.563 F1 score, suggesting that even within classical frameworks, careful feature engineering leads to tangible improvements. This supports their claim while extending it with modern feature enhancements.

*Patoary et al. (2020) [2]:*

This paper applied deep learning models (e.g., Bi-LSTM) to Bengali POS tagging and demonstrated the following.

Deep learning yields superior tagging accuracy due to its ability to capture contextual and sequential information.

Their LSTM model reported F1 scores in the range of 0.61 to 0.65.

Our LSTM model achieved an F1 score of 0.628, which places it well within the range reported by Patoary et al., although working on a different language and dataset. This confirms that LSTM models maintain strong performance across languages and corpora, reinforcing the generalizability of deep learning approaches in POS tagging tasks.

*Chiche & Yitagesu (2022) [3]:*

This comprehensive review found that

Deep learning models consistently outperform machine learning models in POS tagging tasks,

Especially for morphologically rich or low-resource languages.

Traditional models like logistic regression remain valuable for their speed and simplicity in well-structured or smaller datasets.

Our study aligns with their conclusions that the LSTM model outperformed all logistic regression variants (F1: 0.628 vs. 0.479–0.563). However, logistic regression with PCA served as an efficient and surprisingly competitive baseline (accuracy: 0.672), affirming its practicality in resource-constrained environments.

This confirms their recommendation of deep learning for accuracy, but classical models for efficiency and ease of deployment.

## V. Future Directions

In the future, this project can be improved using more advanced contextual word embeddings like BERT or ELMo, which can better understand the meaning of words based on their context in a sentence. These models help to capture subtle relationships and grammar rules in language more effectively than traditional methods. Additionally, combining both statistical models (such as Logistic Regression) and deep learning techniques (such as LSTM) into a hybrid model could boost accuracy and adaptability across different types of text data and domains.

This work also has the potential to be used in real-time applications such as chatbots, virtual assistants, speech-to-text systems, and language learning platforms, where quick and accurate grammatical analysis is needed. By optimizing the models to work faster and with fewer computing resources, they can be used in real-world environments like mobile apps or online tools, making POS tagging both efficient and practical for daily use.

## VI. Conclusion

In conclusion, this project explored the effectiveness of both statistical and deep learning approaches for Part-of-Speech (POS) tagging using a 370k English words corpus. Logistic Regression, along with PCA and Word2Vec embeddings, provided a simple yet interpretable baseline, while the Long Short-Term Memory (LSTM) model demonstrated stronger performance in capturing complex patterns. Unlike many previous studies that focus on sentence-level tagging, our word-level analysis adds a unique perspective. The results highlight the trade-offs between model complexity and interpretability, offering valuable insights for future advancements in POS tagging systems.

## References

[1] Pranckevičius, T., & Marcinkevičius, V. (2016). Comparison of classical machine learning algorithms for multiclass classification of sentiments on Twitter. Informatica, 27(1), 85–97. https://doi.org/10.15388/Informatica.2016.107

[2] Patoary, M. M. H., Karim, M. R., Chowdhury, S. A., & Rahman, M. M. (2020). Part-of-speech tagging of Bengali texts using deep learning models. SN Computer Science, 1(4), 1–9. https://doi.org/10.1007/s42979-020-00247-9

[3] A. Chiche & B. Yitagesu, "Part of speech tagging: a systematic review of deep learning and machine learning approaches", Journal of Big Data, vol. 9, no. 10, 2022. doi: 10.1186/s40537-022-00561-y