Hindi Pluralization: A Comparison of CRF and LSTM Models

Divya Gupta
Saarland University
digu00002@stud.uni-saarland.de

Abstract

Humans communicate and understand each other through languages. Languages are an integral part of human survival. As computers only understand binary, language models are used to train languages to make computers understand various language patterns. Large Language Models (LLMs) allowing users to interact with machines in their native language have created a new revolution.

This report focuses on training Conditional Random Fields (CRF) and Long Short-Term Memory (LSTM) models to predict plural forms in Hindi. Hindi has a high morphological inflection system, and pluralization is an important part of it. These techniques help us study how Hindi words change when forming plurals, allowing computers to process the language more effectively. Moreover, this report also compares CRF and LSTM, highlighting their strengths and weaknesses in handling word changes in Hindi.

1 Introduction

Pluralization refers to making a word into a form that expresses more than one[1]. Unlike some languages that follow simple rules, Hindi pluralization is more complex and does not always have a fixed pattern.For instance, masculine nouns ending in *-a* often change to *-e*, as in:

लड़का → **लड़के** (boy → boys)

but feminine nouns add *-e or -ya*, such as:

कताब → **कतिाबें** (book → books)

Some words do not change at all, and others follow irregular patterns. In this report, we are comparing two models explained in further subsections.

CRF focuses only on the word endings of singular forms to figure out the change to be made in the plural form. It does not understand the meaning of words. Example: Only ending changes, 'a' is replaced with 'e'

लड़का → **लड़के** (boy → boys)

LSTM works opposite to CRF, as it focuses on the entire word instead of just the ending. It does not follow a fixed rule but learns from patterns. It predicts change based on word structure. Example:

गाय → **गाएँ** (cow → cows)

1.1 Research Questions

- Which model performs better (CRF or LSTM)?
- Strengths and weaknesses of each model?
- Future improvements for both models?

Understanding these questions will help in building better morphological models for Hindi and similar languages.

2 Related Work

Hindi has a rich inflection system where word changes depend on gender, number, and case. To handle this, researchers use two approaches:

- **Rule-based methods:** Use fixed rules to predict word changes.
- **Data-driven methods:** Learn patterns from data to predict word changes.

The paper *"Hindi Morphological Analyzer and Generator"*[2] uses a rule-based approach to analyze inflection in Hindi. A morphological analyzer breaks down words to find their root form and grammatical features (e.g., part of speech, gender, number, person). It works well for common words but struggles with irregular ones. Also, updating and maintaining rules manually requires significant effort.

In contrast, our paper focuses on training language models using data-driven approaches like CRF and neural networks (LSTM). Unlike rule-based models, our models automatically learn patterns from the UniMorph dataset, though they require large amounts of training data. They can also struggle with rare or highly inflected words. This project compares two models for Hindi word changes using the UniMorph dataset, aiming to determine which approach is more effective.

3 Dataset Handling and Preprocessing

In both models, we use the Hindi dataset from UniMorph. However, we process it very differently in both models.

CRF follows a more structured and automated process. It loads datasets using pandas, which removes any duplicates, checks for missing values, and structures the dataset properly. Moreover, to ensure the correctness of the dataset, it runs several sanity checks before training. It further breaks every word into individual characters and applies one-hot encoding to it. It also fine-tunes its settings using grid search, which helps find the best values for key parameters c_1 (L1 regularization) and c_2 (L2 regularization). By testing different combinations, values were selected that balanced model complexity and generalization. The optimal values ensured that the model captured the necessary patterns without overfitting.

LSTM relies on manual adjustments and linguistic transformations. Instead of one-hot encoding, it uses alignment-based transformations to process word variations. Unlike CRF, it does not use grid search for tuning. Instead, it manually adjusts weights based on the dataset's properties. It reads the dataset manually, line by line, without using pandas. Instead of using automated methods, it filters and extracts relevant information using custom rules. This provides more flexibility in handling the data but makes the process more complex and time-consuming.

4 Experiments and Results

4.1 Experiment Setup

The entire experiment was divided into three phases:

- **Preprocessing:** During the initial phase, datasets were divided into smaller sets to feed into the model. This was done to obtain more structured training data.
- **Training:** During training, sanity checks and hyperparameter tuning were implemented to optimize the performance of models and to check for any bugs and errors.
- **Evaluation:** Finally, both models were assessed to determine their accuracy, precision, F1-score, and recall values.

4.2 CRF Model Results

The model was trained on 37,564 labeled instances, equally divided between plural and singular groups to remove any biases. The test set consisted of 16,100 instances, again with equal representation of both classes.

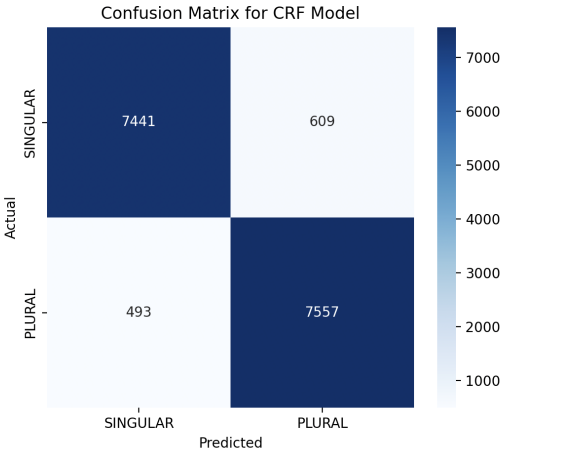

Figure 1: Confusion matrix for CRF model[4]

	Precision	Recall	F1-score
PLURAL	0.925	0.939	0.932
SINGULAR	0.938	0.924	0.931
Overall accuracy of the model:	93.2%		

Table 1: CRF Model Results (Support: 8050 each)

4.2.1 Error Analysis

- In Hindi, some words can be used in both plural and singular forms, depending on the context. CRF fails in this scenario because it only checks the word endings and not the surrounding words and sentence structures.

  Example: CRF predicted both the cases as plural.

  "करती हो"(You do it)

  - **"लू करती हो"**
  - **"मस्ती करती हो"**

  However, in reality, these phrases can be singular (talking to one person) or plural (talking to many people). CRF couldn't determine which one.

- It also fails in complex sentences with multiple meanings because it does not analyze the complete meaning of a sentence and focuses only on word endings.

  Example: CRF predicted it as singular, but it can be both singular and plural.

  "हो जाती होती" (Becomes/Turns into)

– **“अंगारा हो जाती होती”**

This particular phrase involves multiple words, which change the sentence’s meaning completely.

4.3 LSTM Model Results

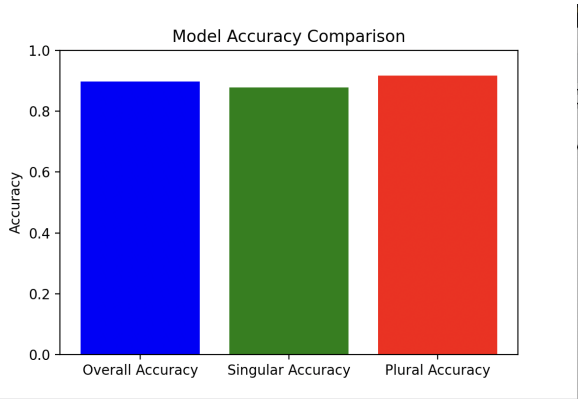

Figure 2: Model accuracy comparison[4]

The LSTM model, trained with a larger dataset of 42,931 instances and tested on 10,733 instances, showed competitive results. Over 20 training epochs, the model’s accuracy improved, stabilizing at around 90.6% in validation performance.

Moreover, plural accuracy was higher than singular accuracy in this model because plural words tend to have more consistent and longer character patterns, making it easier for an LSTM to recognize them. In contrast, singular words are often shorter and more irregular, making their features harder to learn.

	Precision	Recall	F1-score
PLURAL	0.917	0.878	0.897
SINGULAR	0.879	0.917	0.898
Overall Accuracy of the model:			89.7%

Table 2: LSTM Model Results (Support: 5456 for Plural, 5277 for Singular)

4.3.1 Error Analysis

- LSTM learns from words it has encountered before, so if a rare word appears, it tries to predict based on the memory of already trained words. There can be other reasons, such as insufficient training data. Hence, it fails with rare words in the dataset.
- It fails in cases of overfitting, where it remembers the data too well. Instead of understanding general rules, it tends to repeat the same learned patterns.

Example: LSTM false predicted both as plural

– **छोड़ती हो**

– **त्यागती हो**

LSTM may have learned a general rule that **“-ती हो” /“-ती होती”** means plural, even though this is not always true.

5 Comparison and Analysis

While both models achieved high classification accuracy, the CRF model slightly outperformed the LSTM in terms of precision, recall, and F1-score.

The reason LSTM performed slightly worse than CRF is that pluralization mostly depends on short words and endings, and LSTM struggles to work with small data. Moreover, LSTM focuses on memorizing patterns instead of learning general rules, leading to overfitting, while CRF avoids this problem by focusing more on general rules. Additionally, our dataset contained about 42,931 words, which might not be enough for LSTM to fully understand how Hindi words change when pluralized.

Feature	CRF	LSTM
Uses sequence context	✅	✅
Handles complex rules	❌	✅
Works well with small data	✅	❌
Runs faster	✅	❌
Learns new words better	❌	✅

Figure 3: Comparison table of the two models[5]

6 Conclusion

In this report, we compared two models, Conditional Random Fields (CRF) and Long Short-Term Memory (LSTM) networks, for modeling Hindi pluralization using the UniMorph dataset. We covered dataset handling, preprocessing, evaluation, feature extraction, and overall performance.

After the experiment, we can conclude that CRF is a better choice when training on smaller datasets and predictable patterns in Hindi pluralization. However, LSTM can outperform CRF if trained on a much larger dataset, as it has the ability to learn more complex patterns.

7 Future Work

While our study has provided good results, there is still room for improvement and further exploration.

We can enhance and expand the dataset by training the model on not only the UniMorph dataset

but also other sources. Furthermore, we can incorporate data augmentation techniques like synthetic data generation and character substitution to improve results. Additionally, combining CRF and LSTM models could help leverage the strengths of both approaches to further improve training.

8 References

1. Cambridge Dictionary. "Pluralize." https://dictionary.cambridge.org/us/dictionary/english/pluralize

2. Agarwal, A. et al. (2014). "Morphological Analyser for Hindi – A Rule-Based Implementation."

3. OpenAI. "ChatGPT: A Large Language Model." https://openai.com/chatgpt

4. Generated by a Python script from the implemented code.

5. Evaluation metrics computed from CRF and LSTM model experiments.

6. Hochreiter, S., Schmidhuber, J. (1997). "Long Short-Term Memory." *Neural Computation*, 9(8), 1735–1780.

7. Graves, A. (2013). "Generating Sequences With Recurrent Neural Networks." *arXiv preprint arXiv:1308.0850.*

8. Lafferty, J., McCallum, A., Pereira, F. (2001). "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data." *Proc. of ICML*, 2001.

9. Sutton, C., McCallum, A. (2012). "An Introduction to Conditional Random Fields." *Foundations and Trends in Machine Learning*, 4(4), 267–373.

10. UniMorph. "The Universal Morphology Project." https://unimorph.github.io/