*Journal of Computer Science*

Research Paper

# Few-shot Fine-tuning of BERT Multilingual for Hindi Word Sense Disambiguation

**Shailendra Kumar Patel[1], Rakesh Kumar[1], Anuj Kumar Sirohi[2]**

[1]*Department of Computer Science, Assam University, Silchar, India*
[2]*Yardi School of Artificial Intelligence, Indian Institute of Technology, Delhi, India*

*Article history*
Received: 13 February 2025
Revised: 9 May 2025
Accepted: 30 June 2025

Corresponding Author:
Shailendra Kumar Patel
Department of Computer Science,
Assam University, Silchar, India
Email:
shailendra.kr.patel.2019@gmail.com

**Abstract:** Word Sense Disambiguation (WSD) is a fundamental task in Natural Language Processing (NLP), addressing the challenge of identifying correct word meanings in context. This task is particularly complex for morphologically rich and resource-limited languages like Hindi, which exhibit significant lexical ambiguity compounded by limited availability of annotated corpora. To address these challenges, we propose a supervised approach combining the multilingual BERT model (mBERT) with Hindi WordNet as a structured lexical resource. Using few-shot learning, we fine-tune mBERT on a dataset constructed from Hindi WordNet to disambiguate contextually ambiguous words across four parts of speech (POS): nouns, verbs, adjectives, and adverbs. Experiments on standard Hindi WSD benchmarks demonstrate that our method significantly outperforms traditional rule-based and embedding-based approaches, achieving 96.48% accuracy—an approximate 3% improvement over the strongest baseline. These results validate the effectiveness of integrating contextualized embeddings from pre-trained language models with structured lexical databases, highlighting the promise of hybrid techniques for advancing WSD in low-resource languages and providing a framework applicable to other morphologically complex languages with similar resource constraints.

**Keywords:** Word Sense Disambiguation, Hindi Natural Language Processing, Multilingual BERT, Hindi WordNet, Low-Resource Languages, Few-Shot Learning, Contextualized Embeddings

## Introduction

Word Sense Disambiguation (WSD) is a core task in Natural Language Processing (NLP) that involves determining the correct meaning of a word based on its context. Accurate WSD is essential for enhancing the performance of various NLP applications such as machine translation, sentiment analysis, and information retrieval. In machine translation, WSD helps ensure semantic consistency by selecting the most contextually appropriate word meanings. In sentiment analysis, it enables accurate interpretation of ambiguous words, thereby improving sentiment classification (Navigli, 2009).

WSD is particularly challenging in morphologically rich, resource-scarce languages like Hindi due to extensive inflectional variation and complex syntax; the limited availability of sense-annotated datasets further restricts robust modeling (Gujjar et al., 2023).

Spoken by over 600 million people, Hindi ranks as the third most spoken language globally and plays a critical role in India's linguistic and cultural diversity (Eberhard et al., 2021). With the rising digitization of Hindi content, the demand for advanced NLP tools to support tasks such as translation, sentiment analysis,

and information retrieval has become increasingly urgent. Within this context, WSD is essential step toward resolving semantic ambiguity in Hindi texts. The challenges are compounded by homonymy and polysemy, and by multiple script representations, standard Devanagari and widely used Romanized (Latin-script) Hindi, which introduce non-standard spellings and necessitate normalization/back-transliteration before downstream modeling (Bali et al., 2014; Roark et al., 2020; Parikh et al., 2021; Gujjar et al., 2023). Hindi's syntactic intricacy and context-sensitive semantics further underscore the need for advanced disambiguation techniques.

BERT (Bidirectional Encoder Representations from Transformers) has emerged as a powerful pre-trained language model, capable of capturing deep contextual relationships through its bidirectional attention mechanism (Devlin et al., 2019). Its multilingual variant BERT Multilingual (mBERT) is trained on 104 languages using a unified vocabulary and parameter space, enabling effective cross-lingual transfer and making it suitable for low-resource languages (Pires et al., 2019). mBERT provides rich contextual embeddings that significantly advantage WSD tasks, especially in scenarios with limited annotated resources.

Despite notable progress in WSD for high-resource languages, Hindi remains underexplored: rule-based approaches struggle with structural variability, and data-driven models are constrained by the limitation of sense-annotated corpora. (Hadiwinoto et al., 2019; Gujjar et al., 2023; Sharma et al., 2019). Transformer-based models like BERT have achieved strong results in WSD for other languages, their application to Hindi remains limited. The absence of standardized datasets and comparative studies involving multilingual or Hindi specific transformer models further highlights this gap.

In this paper we present a novel approach to Hindi WSD by using the mBERT model to address semantic ambiguity. Our method focuses on disambiguating context-dependent word meanings across four parts of speech: nouns, verbs, adjectives, and adverbs. The approach involves preprocessing and fine-tuning mBERT with Hindi WordNet-based dataset. By using BERT's contextual learning capabilities this research aims to enhance the performance of Hindi WSD, ultimately contributing to improved machine translation, sentiment analysis, and information retrieval. This work addresses a critical gap in computational linguistics and supports the broader development of NLP tools for underrepresented languages.

## Literature Review

This section reviews significant contributions to Hindi Word Sense Disambiguation (WSD) with a particular focus on approaches using the Hindi WordNet dataset.

One of the earliest methods was introduced by Sinha et al. (2004), who implemented an overlap-based technique in which the sense with the intersection between the context and related words was selected. Their custom 'Build Context' module enabled noun disambiguation with reported accuracies ranging from 40% to 70%. The method's dependence on lexical overlap limited its scalability and generalizability.

Mishra & Siddiqui (2012) proposed a hybrid WSD model based on the Yarowsky algorithm, combining unsupervised clustering with manual annotation. By using Hindi WordNet resources and Roget's Thesaurus, the approach reduced the dependency on large annotated corpora. The need for manual intervention restricted its automation and scalability.

Sarika & Sharma (2016) developed a supervised WSD method using cosine similarity by vectorizing both the sense definitions and input context. Their model achieved a precision of 78.99% for nouns, though its evaluation was constrained to a small dataset comprising 90 ambiguous words. In a similar line of work, Sharma & Joshi (2019) used the Lesk algorithm with Hindi WordNet, reaching an accuracy of 71.40%. This approach also suffered from limitations due to its dependency on lexical overlap.

With the advent of distributed word representations, Kumari & Lobiyal (2019) explored the use of Word2Vec (CBOW and Skip-Gram) to compute cosine similarity between context and sense vectors. Despite adopting modern embedding techniques, their approach achieved only 52% accuracy, likely due to challenges in capturing semantic nuances in a low-resource language like Hindi.

Tripathi et al. (2021) enhanced the traditional Lesk algorithm by using glosses, hypernyms, hyponyms, and synonyms to improve the semantic scoring mechanism. Although this enriched semantic representation the method remained fundamentally rule-based, thereby limiting its adaptability.

More recently Jha et al. (2023) introduced a graph-based unsupervised model that exploited the structural

connectivity of Hindi WordNet. This approach showed promise in disambiguating open-class words by using network-based representations. The dependency on heuristic scoring and graph construction may restrict its adaptability across different contexts and domains.

### *BERT Multilingual: Background*

BERT Multilingual (mBERT) is an extension of the original BERT architecture (Bidirectional Encoder Representations from Transformers) developed by Devlin et al. (2019). Designed to support 104 languages, mBERT is a versatile model for multilingual and low-resource language processing. It is built on the Transformer architecture introduced by Vaswani et al. (2017), which uses self-attention mechanisms to capture bidirectional context in contrast to traditional unidirectional models.

The architecture of mBERT consists of 12 transformer encoder layers, each with a hidden size of 768 and 12 self-attention heads, total approximately 110 million parameters. A shared Word-Piece tokenizer with a vocabulary of 110,000 tokens is used across all supported languages, enabling the model to learn language-agnostic representations that generalize across different linguistic families.

Pre-training is performed on a multilingual Wikipedia corpus covering all 104 supported languages ensuring broad linguistic diversity (Wu & Dredze, 2020). The model is trained using two core objectives:

*Masked Language Modeling (MLM):* Random tokens in the input are masked and the model learns to predict them using the surrounding bidirectional context.

*Next Sentence Prediction (NSP):* The model is trained to determine whether a given sentence pair appears sequentially in the original text, improving inter-sentential comprehension (Clark et al., 2020).

These objectives equip mBERT with the capability to handle complex NLP tasks including text classification, question answering, named entity recognition, and WSD.

Despite its robust multilingual capabilities mBERT has limitations. Its shared parameter space across multiple languages can restrict the model's ability to capture fine-grained linguistic features specific to morphologically rich and low-resource languages such as Hindi, Turkish, and Finnish (Pires et al., 2019; Conneau et al., 2020). The pre-training data is skewed toward high-resource languages which may result in performance disparities across the language spectrum.

mBERT remains a foundational model in multilingual NLP, widely used in benchmark suites such as XGLUE and XTREME (Liang et al., 2020).

Fig. 1 presents the mBERT architecture. On the left the 12-layer encoder stack is shown where each successive layer improves contextual embeddings. On the right the multi-head self-attention mechanism is shown wherein multiple attention heads work in parallel to capture different syntactic and semantic dependencies. These outputs are then concatenated and linearly transformed, generating rich, context-aware embeddings, a critical asset for high-precision task for Hindi Word Sense Disambiguation.

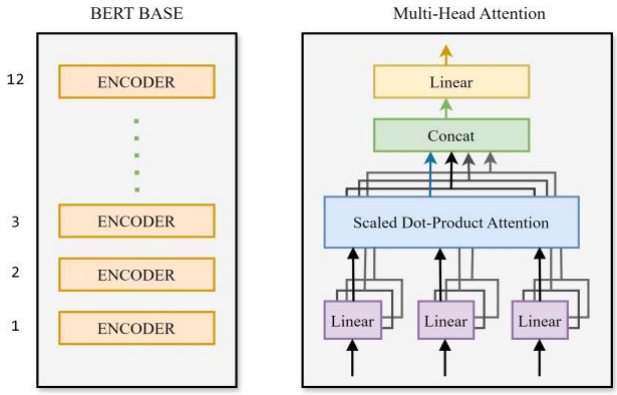


**Fig. 1:** Architecture of the BERT Multilingual (mBERT) Model and Multi-Head Self-Attention Mechanism

### *Few-Shot Fine-Tuning*

Few-shot fine-tuning is performed by adapting the pre-trained mBERT model to the Hindi WSD task using a limited number of labeled examples. During training only the first two and last two transformer layers are updated while the remaining layers are kept frozen to preserve pre-trained knowledge and prevent overfitting. Target words are explicitly highlighted in the input using special markers (<tgt>...</tgt>) to focus the model's attention. A small batch size, a low learning rate of 2e-5, and dynamic learning rate scheduling are used for stable optimization. To address class imbalance random oversampling is applied thereby enhancing few-shot generalization.

### *Few-Shot Learning Parameters*

1. Support Set: The support set consists of labeled sentences where the target word is explicitly marked with special tokens providing minimal but focused context for learning.
2. N-way Classification: The model performs N-way multi-class classification where N represents the number of unique word senses present after applying oversampling to the dataset.

3. K-shot Learning: In this study, explicit K-shot episodes are not manually constructed. Random oversampling is used to balance the distribution of training instances across classes. By ensuring that each class is represented with an approximately equal number of examples, the approach effectively simulates a few-shot learning environment. This strategy provides the model with more uniform exposure to all sense categories, thereby reducing bias toward majority classes and enhancing its generalization ability in low-resource languages.
4. Model Adaptation Strategy: Only the first two and the last two layers of the mBERT model are fine-tuned, while the intermediate layers remain frozen. This selective fine-tuning strategy enables the model to adapt effectively to the task without overfitting.
5. Input Encoding: The input sequence is limited to a maximum of 20 tokens to maintain focus on the target word and reduce the risk of overfitting associated with longer sequences.
6. Batch Size and Optimization: A small batch size of 16 is used. Optimization is performed using the Adam and RMSprop optimizers with a learning rate of 2e-5, providing stable convergence even with limited training data.
7. Training: Training is conducted over 5, 10, 50, and 100 epochs and model is saved based on improvements in validation accuracy to select the best-performing model.
8. Evaluation Metrics: The model's performance is evaluated using accuracy, precision, recall, and F1-score to provide a comprehensive understanding of overall prediction quality.

## Materials and Methods

We implemented a few-shot fine-tuning approach using the BERT Multilingual (mBERT) model for Hindi Word Sense Disambiguation (WSD), following the architecture proposed by Devlin et al. (2019). Our framework consists of a pre-trained mBERT encoder paired with a classification layer. The encoder extracts contextual features of polysemous words from input sentences which are then passed to the classifier for sense prediction. The overall methodology includes four key stages: dataset preparation, model architecture, training strategy, and evaluation. Fig. 2 and Algorithm 1 present the complete workflow for training and evaluating the Hindi WSD model. We begin with the Hindi WordNet dataset and apply a structured preprocessing pipeline. This includes context tokenization, stopword removal, and class balancing using the RandomOverSampler (Lemaître et al., 2017) technique to address label imbalance. The processed data is divided into training, validation, and testing subsets. In the next stage each sentence is tokenized using the BERT tokenizer followed by label mapping and data loading into model-compatible formats. During fine-tuning we adopt a selective training strategy: the intermediate transformer layers are frozen to reduce computational cost while the first two and last two encoder layers along with the task-specific output layer are updated. This approach ensures a balance between computational efficiency and contextual adaptability. Training is conducted through batch-wise optimization and backpropagation, using cross-entropy loss and the Adam and RMSprop optimizers. Method performance is evaluated using standard classification metrics: accuracy, precision, recall, and F1-score.

### *Dataset Preparation*

The input dataset comprises sentences $S = [w_1, w_2, \ldots, w_n]$, where $w_{target}$ is the word requiring sense disambiguation. To highlight the target word within the sentence, special markers $< tgt >$are introduced.

1. Input Sentence Transformation:

   Given a sentence $S$, the modified sentence $S'$ is:

   $$S' = [\, w_1, \ldots, \text{<tgt>}, w_t, \text{</tgt>}, \ldots, w_n \,]$$

   where $w_i$ wi is a word in the sentence.

2. Tokenization:

   The sentence $S'$ is tokenized using the BERT tokenizer. For a sequence length $L$, the tokenizer outputs:

   a. Token IDs $x \in Z^L$:

   $$x = Tokenizer.encode(S')$$

   b. Attention Mask $m \in \{0,1\}^L$:

   $$m_i = \begin{cases} 1, & if\ token\ i\ is\ valid \\ 0, & if\ token\ i\ is\ padding \end{cases}$$

   c. Label Encoding $y \in \{0, 1, .., N-1\}$, where $N$ is the total number of sense classes.

### *Model Architecture*

The proposed Hindi WSD method includes two core components: a BERT encoder, which generates contextualized token representations and a classification head, which maps these representations to predefined word sense categories.

### *BERT Encoder*

The encoder utilizes the mBERT model to capture deep contextual relationships in the input sequence. Each input token is represented by the sum of three embeddings:

$$E_i = E_{T,i} + E_{P,i} + E_{S,i}$$

where $E_T$ , $E_P$ , and $E_S$ denote token, position and segment embeddings. These embeddings are passed through $L$ transformer layers. Each layer consists of:

1. Multi-Head Self-Attention:

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

with

$$Q = H^{(l)}W_Q, \quad K = H^{(l)}W_K, \quad V = H^{(l)}W_V$$

where $H^{(l)}$ is the output of the $l^{th}$ layer, $W_Q$, $W_K$ and $W_V$ are projection matrices, and $d_k = H/h$ is the dimensionality of each attention head for $h$ heads and hidden size $H$.

2. Feed-Forward Network (FFN):

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$$

The contextualized output representation of the special $[CLS]$ token from the final layer denoted as $h_{[CLS]} \in R^H$ is used for classification.

### *Classification Head*

The classification layer maps $[CLS]$ to a vector of logits over $N$ word sense classes:

$$z = Wh_{[CLS]} + b$$

where $W \in R^{N \times H}$ and $b \in R^N$ are the trainable parameters. The class probabilities $\hat{p}$ are computed via softmax:

$$\widehat{p_i} = \frac{\exp(z_i)}{\sum_{j=1}^{N} \exp(z_j)}, \quad i = 1, \ldots, N$$

The predicted class $\hat{y}$ is obtained as:

$$\hat{y} = \arg\max_i \widehat{p_i}$$

### *Training Procedure*

The model is fine-tuned using the cross-entropy loss, which is well-suited for multi-class classification tasks such as WSD. The loss function is defined as:

$$\mathcal{L} = -\frac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{N} y_{i,j} \log(\widehat{p_{i,j}})$$

where $y_{i,j}$ is the true one-hot label, $\hat{p}_{i,j}$ is the predicted probability and $B$ is the batch size.

During training the middle layers of mBERT are frozen to reduce computational overhead while the first two and last two encoder layers along with the output classification layer are updated. This strategy ensures a balance between efficiency and contextual adaptability.

### *Evaluation Metrics*

To evaluate the model's performance on Hindi WSD, we use the following standard classification metrics:

1. Accuracy: Represents the ratio of correctly predicted labels to the total number of predictions. It offers a general overview of the model's correctness across all classes.

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Samples}}$$

2. Precision: Measures the proportion of correctly predicted instances among all instances predicted as belonging to a particular class. This metric is crucial for understanding the model's ability to minimize false positives.

$$\text{Precision} = \frac{\sum_{c=1}^{N} w_c \cdot \text{TP}_c}{\sum_{c=1}^{N}(\text{TP}_c + \text{FP}_c)}$$

3. Recall*:* Indicates the proportion of actual instances of a class that are correctly identified by the model. It reflects the model's ability to capture all relevant instances, particularly important in imbalanced datasets.

$$\text{Recall} = \frac{\sum_{c=1}^{N} w_c \cdot \text{TP}_c}{\sum_{c=1}^{N}(\text{TP}_c + \text{FN}_c)}$$

4. F1-Score: The harmonic mean of precision and recall, providing a single score that balances both metrics. It is especially useful when there is an uneven distribution of classes or when both false positives and false negatives are important.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

where $TP_c$, $FP_c$, and $FN_c$ are the true positives, false positives and false negatives for class c and $w_c$ is the weight for class $c$.

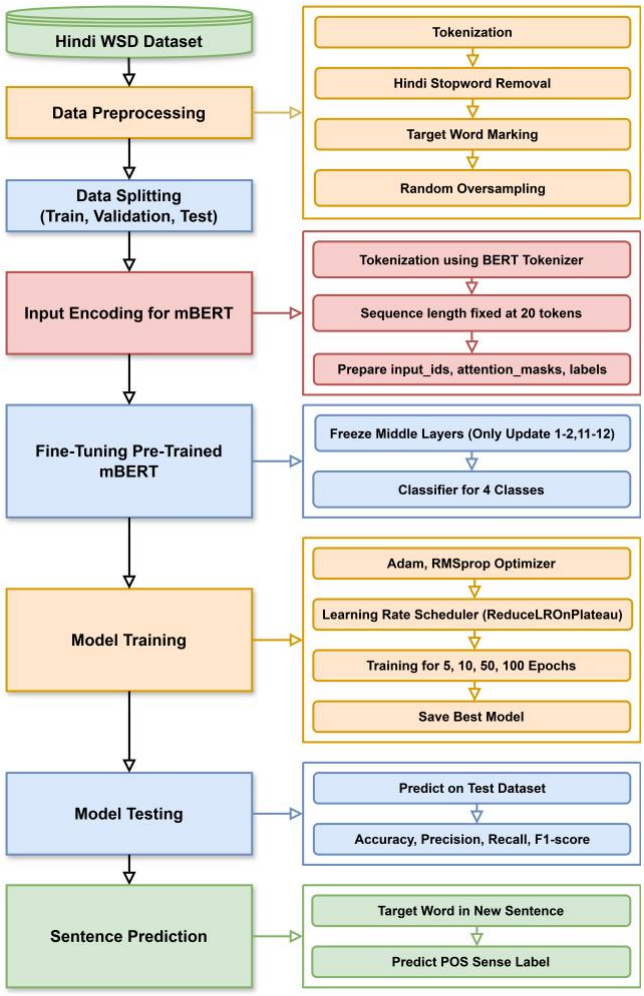


**Fig. 2:** End-to-End Pipeline for Fine-Tuning the mBERT Model for Hindi Word Sense Disambiguation (WSD)

## Experimental Setup and Results

This section outlines the experimental setup and presents empirical results to demonstrate the effectiveness of the proposed approach. Comprehensive experimentation and analysis are conducted to validate the model's performance in Hindi Word Sense Disambiguation (WSD).

### *Dataset*

A well-structured dataset is essential for addressing the challenges of Word Sense Disambiguation. For this research we use the Hindi WordNet developed by IIT Bombay (https://www.cflt.iitb.ac.in/wordnet/webhwn/) which provides as a rich lexical knowledge base for the Hindi language (Bhattacharyya, 2006). Modeled after the English WordNet the Hindi WordNet focuses on open-class categories nouns, verbs, adjectives, and adverbs. Each entry in the dataset includes three components:

1. Word: the target ambiguous word requiring disambiguation.
2. Context: a sentence or example representing the usage of the word.
3. Label: the predicted sense of the word derived from the Hindi WordNet.

**Algorithm 1:** Hindi WSD Using mBERT

**Input:** Dataset $D_{\text{train}}, D_{\text{eval}}, D_{\text{test}}$, pre-trained multilingual BERT, hyperparameters: $lr$, $batch\_size$, $num\_epochs$
**Output:** Trained Model, Accuracy Metrics
**procedure** DATASET PREPROCESSING($D, model\_name$)
  Initialize tokenizer using BERT from $model\_name$
  **for all** data points in $D$ **do**
    Extract word and context
    Replace target word with `<tgt>` markers
    Tokenize sentence with padding/truncation
    Convert labels to integers
    Store tokenized inputs and labels
  **end for**
  Compute unique label indices and count
**end procedure**
**procedure** TRAINING($model, train\_loader, optimizer, criterion, device$)
  Set model to train mode
  **for all** batches in $train\_loader$ **do**
    Load $input\_ids$, $attention\_mask$, and $labels$ to device
    Perform forward pass
    Compute loss
    Backpropagate and update parameters
    Track loss and accuracy
  **end for**
**end procedure**
**procedure** EVALUATION($model, eval\_loader, criterion, device$)
  Set model to evaluation mode
  **for all** batches in $eval\_loader$ **do**
    Load inputs and labels
    Perform forward pass without gradient
    Compute loss and predictions
    Track evaluation metrics
  **end for**
**end procedure**
**procedure** RUN TRAINING($num\_epochs, model, scheduler, train\_loader, eval\_loader$)
  **for** $epoch = 1$ to $num\_epochs$ **do**
    Train model and evaluate on validation data
    Update learning rate using scheduler
    Save model if validation accuracy improves
    Print epoch statistics
  **end for**
**end procedure**
**procedure** TEST MODEL($test\_loader, model$)
  Set model to evaluation mode
  Predict and evaluate metrics: accuracy, precision, recall, and F1-score
  Output classification metrics
**end procedure**
**procedure** PREDICTION($sentence, model, word, tokenizer$)
  Preprocess sentence: Mark target word
  Tokenize and encode sentence
  Perform forward pass and predict label
  Map predicted label to word sense
  Return prediction
**end procedure**
**Main Steps:**
Preprocess datasets $D_{\text{train}}, D_{\text{eval}}, D_{\text{test}}$
Initialize model, optimizer, loss function, and scheduler
Train and evaluate model over $num\_epochs$
Test model on $D_{\text{test}}$ and evaluate metrics
Predict word sense for new sentences using trained model

Before resampling the dataset shows significant class imbalance as shown in Table 1 and Fig. 3. Nouns dominate with 30352 instances while adverbs are severely underrepresented with only 571 samples. This imbalance poses a challenge for model generalization, especially for minority classes.

To minimize this, we apply RandomOverSampler (Lemaître et al., 2017) resampling technique that synthetically balances the dataset by duplicating minority class samples. This ensures that the model receives sufficient training examples across all parts of speech, promoting equitable learning and reducing bias toward majority classes.

**Table 1:** Class Distribution in the Imbalanced Hindi WSD Dataset

| POS | Label | Number of Samples |
|---|---|---|
| Noun | 0 | 30352 |
| Adjective | 1 | 6579 |
| Verb | 2 | 3894 |
| Adverb | 3 | 571 |

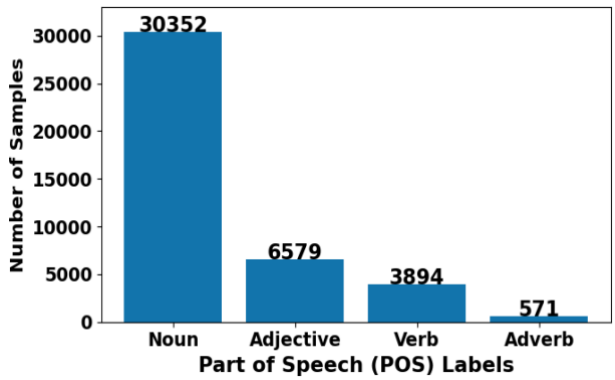


**Fig. 3:** Part-of-Speech (POS) Distribution in the Imbalanced Hindi WSD Dataset

## *Dataset Preprocessing and Oversampling*

Dataset preprocessing is an important step in preparing data for machine learning tasks. It transforms raw, unstructured data into a clean, structured, and analyzable format, thereby improving both data quality and model performance. In this research key preprocessing steps include tokenization, stopword removal, and class balancing to address the issue of skewed class distributions.

Fig. 4 outlines the initial stages of the preprocessing pipeline. The process begins with loading the Hindi WordNet dataset followed by context tokenization and the removal of non-informative stopwords. To resolve class imbalance oversampling techniques are applied, resulting in a more balanced and representative dataset that is suitable for robust model training. A step-by-step description of this process is provided in Algorithm (2), covering context processing, stopword filtering, feature-label separation, and the application of the RandomOverSampler (Lemaître et al., 2017) technique.

### *Load Hindi WordNet Dataset*

The preprocessing pipeline begins by importing the Hindi WordNet dataset into a structured format. The data is loaded into a Pandas DataFrame, a flexible tabular structure that facilitates efficient data manipulation, transformation and exploration throughout the preprocessing stages.

### *Tokenization*

Tokenization involves breaking down the context sentences into smaller units called tokens which can be words, subwords. This transformation converts raw text into a structured form suitable for downstream NLP tasks. Tokenization also works as a foundational step for additional processing tasks such as stopword removal and feature extraction, enabling the model to predict and learn from linguistic patterns more effectively.

### *Stopword Removal*

To further improve the tokenized text, stopwords commonly occurring words with small semantic value are removed. We use the stopwordsiso library to obtain a Hindi specific stopword list as detailed in Table (2). Eliminating these non-essential words reduces data dimensionality and noise, allowing the model to focus on more informative linguistic features. This enhances the overall efficiency and accuracy of the learning process, particularly for the Hindi WSD task.

### *Oversampling to Handle Imbalance*

Due to the significant class imbalance in the original dataset particularly the underrepresentation of certain parts of speech (POS) such as adverbs, we apply RandomOverSampler (Lemaître et al., 2017) to balance the distribution of samples across all classes. This technique duplicates samples from minority classes to ensure equitable representation, which is essential for fair and unbiased training. The result is a balanced dataset that supports improved generalization and consistent model performance across all word categories.

**Table 2:** List of Hindi Stopwords Used in the Hindi WSD Task

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| अंदर | अत | अदि | अप | अपना | अपनि | अपनी | अपने |
| अभि | अभी | आदि | आप | इहिं | इहें | इहों | इतयादि |
| इत्यादि | इन | इनका | इन्हीं | इन्हें | इन्हों | इस | इसका |
| इसकि | इसकी | इसके | इसमें | इसि | इसी | इसे | उहिं |
| उहें | उहों | उन | उनका | उनकि | उनकी | उनके | उनको |
| उन्हीं | उन्हें | उन्हों | उस | उसके | उसि | उसी | उसे |
| एक | एवं | एस | एसे | ऐसे | ओर | और | कइ |
| कई | कर | करता | करते | करना | करने | करें | कहते |
| कहा | का | काफि | काफ़ी | कि | किंहें | किंहों | कितना |
| किन्हें | किन्हों | किया | किर | किस | किसि | किसी | किसे |
| की | कुछ | कुल | के | को | कोइ | कोई | कोन |
| कोनसा | कौन | कौनसा | गया | घर | जब | जहाँ | जहां |
| जा | जिंहें | जिंहों | जितना | जिधर | जिन | जिन्हें | जिन्हों |
| जिस | जिसे | जीधर | जेसा | जेसे | जैसा | जैसे | जो |
| तक | तब | तरह | तिंहें | तिंहों | तिन | तिन्हें | तिन्हों |
| तिस | तिसे | तो | था | थि | थी | थे | दबारा |
| दवारा | दिया | दुसरा | दुसरे | दूसरे | दो | द्वारा | न |
| नहिं | नहीं | ना | निचे | निहायत | नीचे | ने | पर |
| पहले | पुरा | पूरा | पे | फिर | बनि | बनी | बहि |
| बही | बहुत | बाद | बाला | बिलकुल | भि | भितर | भी |
| भीतर | मगर | मानो | मे | में | यदि | यह | यहाँ |
| यहां | यहि | यही | या | यिह | ये | रवासा | रहा |
| रहे | ऱ्वासा | रखें | लिए | लिये | लेकिन | व | वगेरह |
| वरग | वर्ग | वह | वहाँ | वहां | वहिं | वहीं | वाले |
| वुह | वे | वग़ैरह | संग | सकता | सकते | सबसे | सभि |
| सभी | साथ | साबुत | साभ | सारा | से | सो | हि |
| ही | हुअ | हुआ | हुइ | हुई | हुए | हे | हें |
| है | हैं | हो | होता | होति | होती | होते | होना |
| होने | | | | | | | |

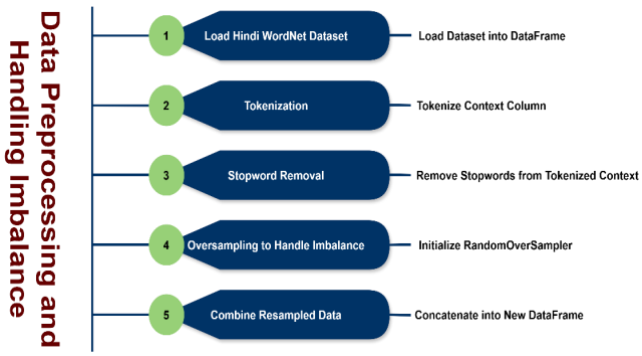


**Fig. 4:** Preprocessing Pipeline for the Hindi WordNet Dataset and Handling Class Imbalance

**Algorithm 2:** Data Preprocessing and Class Imbalance Handling for Hindi WSD

**Input:** Dataset $D$ with columns `context` and `label`.
**Output:** Preprocessed and Balanced Dataset $D_{\text{res}}$.
**procedure** PREPROCESSANDRESAMPLE($D$)
  **Load the Dataset:**
  $D \leftarrow$ read_csv("HindiWordNetDataset.csv")
  **Tokenize Context:**
  **for** each row $i$ in $D$ **do**
    $D[i, \text{context}] \leftarrow \text{word\_tokenize}(D[i, \text{context}])$
  **end for**
  **Remove Stopwords:**
  $S \leftarrow$ load_stopwords("hi")
  **for** each row $i$ in $D$ **do**
    $D[i, \text{context}] \leftarrow \{w \mid w \in D[i, \text{context}] \wedge w \notin S\}$
  **end for**
  **Separate Features and Labels:**
  $X \leftarrow D[\text{columns} \neq \text{label}]$
  $y \leftarrow D[\text{label}]$
  **Handle Class Imbalance using Oversampling:**
  $\text{ROS} \leftarrow \text{RandomOverSampler}(\text{random\_state} = 42)$
  $X_{\text{res}}, y_{\text{res}} \leftarrow \text{ROS.fit\_resample}(X, y)$
  **Recombine Features and Labels:**
  $D_{\text{res}} \leftarrow \text{concat}(X_{\text{res}}, y_{\text{res}})$
  **Return:** $D_{\text{res}}$
**end procedure**

To ensure a robust and unbiased evaluation of the proposed Hindi WSD method the dataset is divided into training, validation, and test sets using stratified random sampling based on class labels. This technique preserves the original distribution of word senses across all subsets, thereby minimizing the risk of performance skew due to class imbalance. The test set remains entirely unseen during both training and validation phases, ensuring that the model's evaluation is conducted on genuinely novel data. This strict separation helps prevent data leakage and provides a reliable measure of the model's ability to generalize to unseen instances. Performance is evaluated using standard classification metrics accuracy, precision, recall, and F1-score calculated on the test set. These metrics collectively offer a comprehensive view of the model's effectiveness across both majority and minority classes. While stratified sampling significantly improves class distribution consistency, the potential overlap of semantically similar contexts between subsets may still introduce minor evaluation biases. This splitting strategy remains a widely accepted best practice for fair and consistent model evaluation in low-resource NLP tasks.

**Table 3:** Class Distribution in the Balanced Hindi WSD Dataset

| POS | Label | Training Set | Validation Set | Test Set |
|---|---|---|---|---|
| Noun | 0 | 19425 | 4857 | 6071 |
| Adjective | 1 | 19425 | 4857 | 6071 |
| Verb | 2 | 19425 | 4857 | 6071 |
| Adverb | 3 | 19425 | 4857 | 6071 |

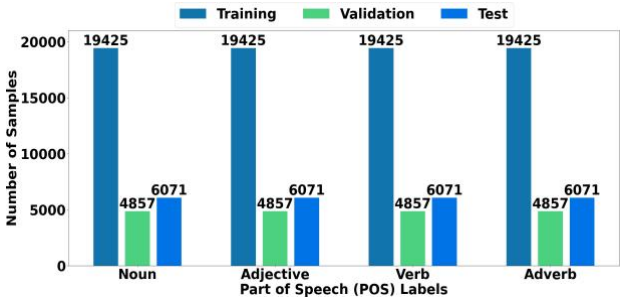


**Fig. 5:** POS-wise Distribution of Samples in the Balanced Hindi WSD Dataset Across Training, Validation, and Test Splits

To ensure fair evaluation and effective model training the dataset is meticulously balanced across all classes, providing equal representation for each Part-of-Speech (POS) category. The class labels are encoded as follows: Noun (0), Adjective (1), Verb (2), and Adverb (3). After applying oversampling techniques, the training set consists of 19425 samples per class, the validation set contains 4857 samples per class, and the test set includes 6071 samples per class. This uniform distribution is summarized in Table (3), which outlines the dataset statistics following the balancing process. Figure (5) provides a visual representation of the consistent class distribution across all data splits. By ensuring equal exposure to each POS category this balanced partitioning reduces bias toward any single class and supports more robust and reliable performance evaluation of the Hindi WSD model.

## Experimental Settings

In this research the mBERT model is used for fine-tuning on the Hindi Word Sense Disambiguation (WSD) task. The model architecture comprises 12 transformer encoder layers, each containing 12 self-attention heads and a hidden size of 768. To prevent overfitting during training, a dropout rate of 0.1 is applied within the classifier layer. The model is trained with a maximum sequence length of 20, a batch size of 16, and an initial learning rate of 2e-5. To evaluate the model's performance under varying training conditions, experiments are conducted over 5, 10, 50, and 100 epochs using two different optimization algorithms Adam and RMSprop. A learning rate scheduler specifically ReduceLROnPlateau is used to dynamically adjust the learning rate in response to changes in validation loss thereby enhancing training stability and convergence. The classification task is structured around four output classes corresponding to the part-of-speech based sense categories defined in the WSD dataset. All method development and training processes are implemented using the PyTorch framework. Experiments are executed on an NVIDIA Tesla P100 GPU with 16 GB of RAM, providing the computational efficiency necessary for high-performance training. Table 4 summarizes the complete set of hyperparameters and configurations used during fine-tuning.

**Table 4:** Experimental Setup for Hindi WSD Using the mBERT Model

| Component | Hyperparameter Setting |
|---|---|
| Pre-trained Model | bert-base-multilingual-cased |
| Tokenizer Settings | max_length = 20, padding = max_length, truncation = True, add_special_tokens = True |
| Model Fine-tuning Strategy | Freeze all layers except first 2 and last 2 |
| Batch Size | 16 |
| Optimizers | Adam, RMSprop |
| Learning Rate | 2e-5 |
| Loss Function | CrossEntropyLoss |
| Learning Rate Scheduler | ReduceLROnPlateau (mode = min, factor = 0.1, patience = 2) |
| Number of Epochs | 5, 10, 50, 100 |
| Dataset Splitting | Stratified split (random_state = 42) |

## Results and Discussion

The proposed approach is evaluated using the Hindi WordNet dataset and benchmarked against several existing methods for Hindi WSD. As presented in Table (5), the fine-tuned BERT Multilingual model demonstrates superior performance across all major evaluation metrics including accuracy, precision, recall, and F1-score. The method achieves an accuracy of 96.48%, substantially outperforming prior knowledge-based, supervised, semi-supervised, and unsupervised approaches. This notable improvement underscores the model's capability to capture fine-grained contextual information and disambiguate word senses effectively, even in a morphologically rich and resource limited language like Hindi. The results reaffirm the effectiveness of transformer-based architectures particularly those using contextual embeddings from pre-trained multilingual models in advancing WSD tasks for low-resource languages.

**Table 5:** Comparison of Accuracy Between the Proposed mBERT Approach and Existing Hindi WSD Methods

| Hindi WSD Methods | Approach | Accuracy (%) |
|---|---|---|
| WSD Algorithm (Sinha et al., 2004) | Knowledge-Based | 40-70 |
| Yarowsky Algorithm (Mishra & Siddiqui, 2012) | Semi-Supervised | 61.70 |
| Cosine Similarity (Sarika & Sharma, 2016) | Supervised Method | 78.99 |
| Lesk Algorithm (Sharma & Joshi, 2019) | Knowledge-Based | 71.40 |
| CBOW and Skip-Gram Techniques (Kumari & Lobiyal, 2019) | Knowledge-Based | 52 |
| Lesk Algorithm (Tripathi et al., 2021) | Supervised Method | 32 |
| Graph-Based Algorithm (Jha et al., 2023) | Unsupervised Method | 63.39 |
| Hindi WSD with IndicBERT | Supervised Method | 93.45 |
| **Proposed: Hindi WSD with BERT Multilingual** | **Supervised Method** | **96.48** |

To evaluate the fine-tuning effectiveness of the mBERT model we conducted experiments using two widely adopted optimizers Adam and RMSprop. The model is trained across four training durations 5, 10, 50, and 100 epochs under identical conditions to ensure a fair comparison. Both optimizers achieved similar precision at 50 epochs (96.54% for RMSprop and 96.61% for Adam), Adam consistently outperformed RMSprop across all evaluation metrics including accuracy, recall, and F1-score.

Adam demonstrated a consistent performance improvement with increasing epochs, underscoring its capacity for sustained learning and convergence. In contrast RMSprop exhibited early performance saturation with minimal gains beyond 10 epochs thereby limiting its suitability for longer training cycles. The 50th epoch is identified as the optimal training point for Adam, achieving the best overall performance in terms of predictive accuracy and class-wise balance. Comprehensive metric comparisons are presented in Table (7) (Accuracy and F1 Score) and Table (8) (Precision and Recall). A visual comparison across all evaluation metrics is provided in Figure (6), which includes subfigures for (a) Accuracy, (b) F1 Score, (c) Precision, and (d) Recall.

The training and validation loss trends presented in Figure (7), indicate stable convergence behavior. The training loss shows a sharp decline during the early epochs, while the validation loss plateaus around epoch 10, suggesting effective learning with minimal signs of overfitting. Similarly, Figure (8) demonstrates a rapid increase in training accuracy, reaching near-perfect levels by epoch 10, while validation accuracy stabilizes above 96%, reflecting strong generalization capability.

The model's best overall performance is summarized in Table (9), where it achieved 96.48% accuracy, 96.61% precision, 96.48% recall, and a 96.42% F1 score. These outcomes confirm the model's robust and balanced effectiveness in resolving word sense ambiguity in Hindi. Visual confirmation of these metrics is provided in Figure (9), reinforcing the model's reliability across evaluation criteria.

In optimizer analysis we also performed a comparative evaluation of Transformer-based models applied to Hindi WSD. Table (6) presents a performance comparison between the IndicBERT and mBERT models, measured using Accuracy, F1 Score, Precision, Recall, and average training time per epoch. The mBERT model clearly outperforms IndicBERT in all categories, achieving 96.48% accuracy, 96.42% F1 Score, 96.41% precision, and 96.35% recall. In comparison, IndicBERT achieves lower scores of 93.45% accuracy, 93.32% F1 Score, 93.48% precision, and 93.45% recall. mBERT also exhibits superior training efficiency, requiring only 4.2 minutes per epoch, compared to 8.5 minutes for IndicBERT. These results highlight mBERT as not only a more accurate but also a more computationally efficient model for the Hindi WSD task.

**Table 6:** Comparison Against Other Transformer-Based Models

| | Accuracy | F1 Score | Precision | Recall | Training Time Per Epoch (minutes) |
|---|---|---|---|---|---|
| IndicBERT | 93.45 | 93.32 | 93.48 | 93.45 | 8.5 |
| **BERT Multilingual** | **96.48** | **96.42** | **96.41** | **96.35** | **4.2** |

**Table 7:** Accuracy and F1 Score of the mBERT Model for Hindi WSD

| | Training Algorithm | | | |
|---|---|---|---|---|
| | Adam | | RMSprop | |
| Epochs | Accuracy (%) | F1 Score (%) | Accuracy (%) | F1 Score (%) |
| 5 | 95.28 | 95.19 | 94.77 | 94.65 |
| 10 | 96.28 | 96.22 | 96.40 | 96.34 |
| **50** | **96.48** | **96.42** | 96.41 | 96.35 |
| 100 | 96.47 | 96.41 | 96.35 | 96.29 |

**Table 8:** Precision and Recall of the mBERT Model for Hindi WSD

| | Training Algorithm | | | |
|---|---|---|---|---|
| | Adam | | RMSprop | |
| Epochs | Precision (%) | Recall (%) | Precision (%) | Recall (%) |
| 5 | 95.44 | 95.28 | 94.99 | 94.77 |
| 10 | 96.36 | 96.28 | 96.47 | 96.40 |
| **50** | **96.61** | **96.48** | 96.54 | 96.41 |
| 100 | 96.60 | 96.47 | 96.47 | 96.35 |

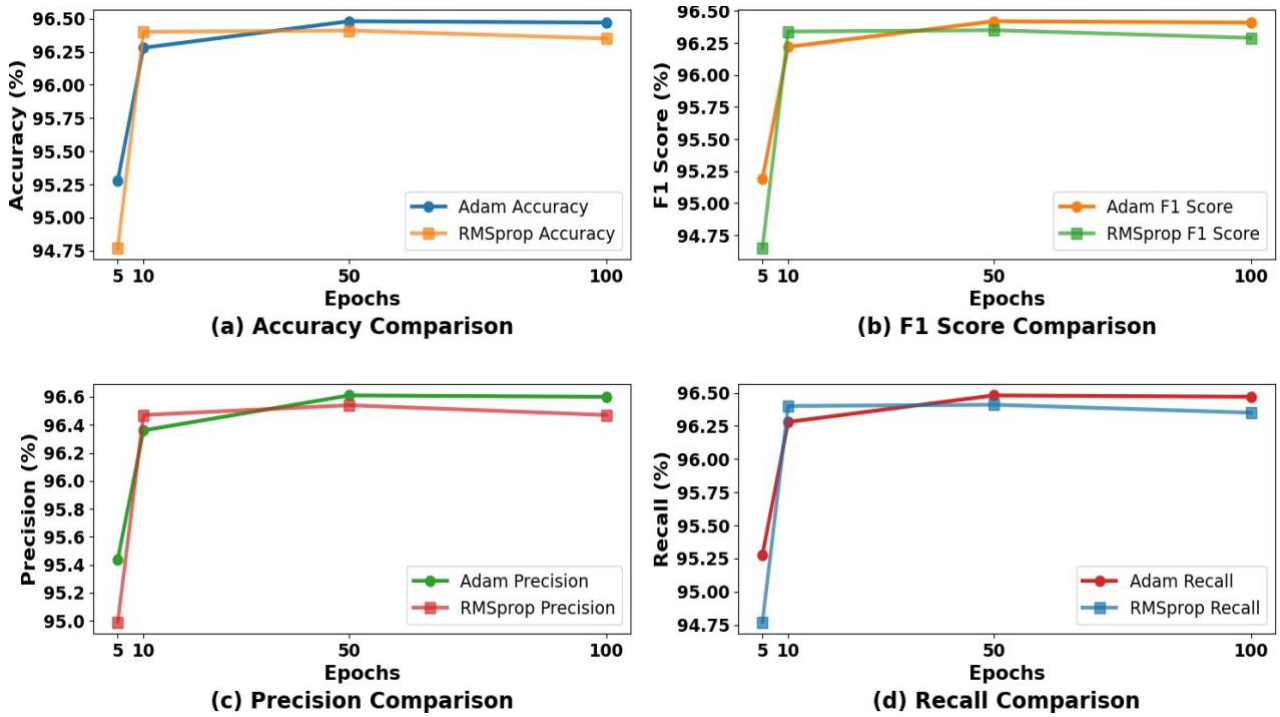


**Fig. 6:** Comparative Performance of the mBERT Model Across Accuracy, F1 Score, Precision, and Recall

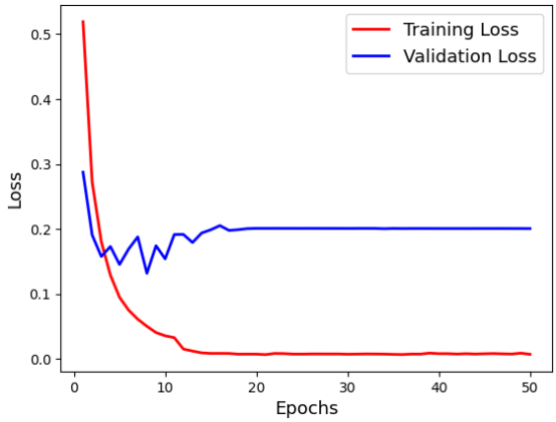


**Fig. 7:** Training and Validation Loss Comparison for the mBERT Model Across Epochs

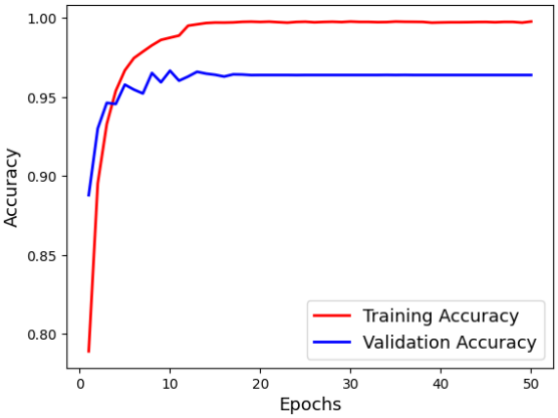


**Fig. 8:** Training and Validation Accuracy Comparison for the mBERT Model Across Epoch

**Table 9:** Best Performance Metrics of the mBERT Model for Hindi WSD

| Evaluation Metric | Score (%) |
|---|---|
| Accuracy | 96.48 |
| Precision | 96.61 |
| Recall | 96.48 |
| F1 Score | 96.42 |

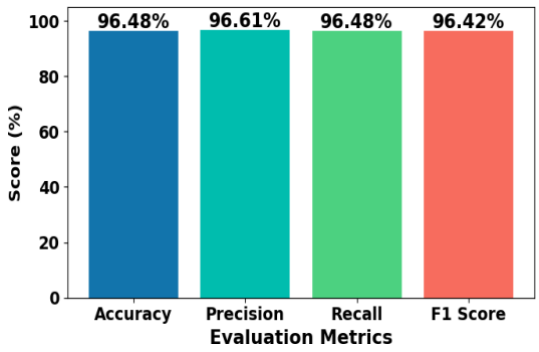


**Fig. 9:** Best Performance Metrics of the mBERT Model for Hindi WSD

To further evaluate the effectiveness of the mBERT model for Hindi a series of ambiguous input words are evaluated across different contextual sentences using a balanced dataset. The results summarized in Table (10), demonstrate the model's strong ability to correctly identify part-of-speech (POS) categories based on contextual information.

Several examples present the model's contextual sensitivity and classification accuracy. The word 'अजात' is accurately classified as an adjective in 'लावारिस अजात होते हैं' and as a noun in 'यहाँ अजातों को आश्रय मिलता है'. Similarly, 'रूढ़' is labeled as an adjective in 'इस शब्द का रूढ़ अर्थ क्या है' and as a noun in 'पंकज शब्द कमल के लिए रूढ़ हो गया है'. The model also correctly disambiguates 'मनमानी', recognizing it as a noun in 'तुम्हारी मनमानी यहाँ नहीं चलेगी' and as an adjective in 'एक बड़े अधिकारी ने अपने विभाग में कुछ मनमानी नियुक्तियाँ कर दी'**.** The word 'अन्य' is consistently identified as an adjective in contexts such as 'बढ़ती हुई जनसंख्या की समस्या के साथ अन्य समस्याएँ भी खड़ी हो गई हैं'.

The model effectively differentiates meanings of 'खाना', tagging it as a noun in 'उसने शतरंज के मोहरे को अगले खाने में रखा' and as a verb in examples like 'शेर मांस खा रहा है'**,** 'दीमक लकड़ी को खा जाती है', and 'बचपन में मैंने बहुत गाली-मार खाई है'**.** Similarly, 'अनारी' is classified correctly as an adjective in 'उस पर अनारी साड़ी बहुत सुन्दर लग रही है' and as a noun in 'माँ ने दुकान से एक किलो अनारी मँगाई है'. The word 'ग्रामीण' is appropriately labeled as an adjective in 'ग्रामीण जन शहरी निवासियों की अपेक्षा कम शिक्षित होते हैं' and as a noun in 'ग्रामीणों ने संतों का बहुत स्वागत किया'.

Words such as 'नासमझ', 'अंध', and 'प्रवीण' are also accurately disambiguated based on context. The model correctly assigns 'बुद्धिमान' as a noun in 'बुद्धिमानों की संगति में रहते-रहते तुम भी बुद्धिमान हो जाओगे' and as an adjective in 'बुद्धिमान व्यक्ति व्यर्थ की बहस में नहीं पड़ते हैं'. The word 'पक्का' is consistently predicted as an adjective in examples involving both literal and metaphorical certainty. Similarly, 'अत्यंत' is correctly identified as an adverb across various contexts, reflecting the model's ability to handle nuanced grammatical usage.

While the model performs robustly across most examples, a notable exception occurs with the word 'उदात्त'. Although it is correctly tagged as an adjective in 'उन्हें उदात्त स्वर ही सुनाई पड़ता है', it is misclassified in 'पंडितजी उदात्त से वेद पढ़ रहे हैं', where it arguably uses as an adverb. This isolated misclassification highlights a potential area for further refinement in capturing subtle semantic shifts.

These results underscore the model's strong contextual understanding and its effectiveness in accurately disambiguating word senses in Hindi. The use of a balanced dataset further supports its consistent performance, reinforcing the model's reliability for linguistically complex tasks such as Hindi WSD.

**Table 10:** Test Results of Hindi WSD Using the mBERT Model

| Word | Sentence | Predicted Label | Predicted POS |
|---|---|---|---|
| अजात | लावारिस अजात होते हैं | 1 | Adjective |
| अजात | यहाँ अजातों को आश्रय मिलता है | 0 | Noun |
| रुचि | इस समय का रुचि अंध वया है | 1 | Adjective |
| रुचि | पंचम वेद वेदम् के लिए रुचि हो गया है | 0 | Noun |
| मनमानी | तुम्हारी मनमानी यहाँ नहीं चलेगी | 0 | Noun |
| मनमानी | एक बड़े अधिकारी ने अपने विभाग में कुछ मनमानी नियुक्तियाँ कर दी | 1 | Adjective |
| अन्य | बढ़ती हुई जनसंख्या की समस्या के साथ अन्य समस्याएँ भी खड़ी हो गई हैं | 1 | Adjective |
| खोजना | उसने शेरों के मोहरे को आगे खोजने में रखा | 0 | Noun |
| खोना | शेर मानव बन खो गया है | 2 | Verb |
| खोना | उसकी यादों को खो जाती है | 2 | Verb |
| खोना | बचपन में मैंने बहुत गलती-गुस्ताखी खोई है | 2 | Verb |
| अन्य | अन्य कारणों से बहुत दुख उठाना पड़ा है | 1 | Adjective |
| अमरीकी | मेरी मुँह दुकान से एक मिला अमरीकी माल है | 0 | Noun |
| अमरीकी | अमरीकी राष्ट्रपति ट्रंप विनाशकारी निर्णयों की सूची में शरीक हैं | 1 | Adjective |
| ग्रामीण | ग्रामीणों ने सस्ती व बेहतर सुविधाओं की मांग की | 0 | Noun |
| ग्रामीण | ग्रामीण संतों का बहुत सत्कार करते हैं | 1 | Adjective |
| अमर | यह अमर वृक्ष है | 1 | Adjective |
| अमर | अजय अमर की जीवनी को सुन रहा है | 0 | Noun |
| अमर | हिंदी की कविता अमर है | 1 | Adjective |
| प्रतीक | प्रतीक ने प्रतीक किया | 0 | Noun |
| प्रतीक | प्रतीकात्मक अनुप्राण ने तेल में महकी की परखकर उसकी आँख पर निशाना लगाया | 1 | Adjective |
| बुद्धिमान | बुद्धिमान व्यक्ति अपनी बुद्धि का उपयोग करता है | 0 | Noun |
| बुद्धिमान | बुद्धिमान व्यक्ति की सलाह से हमें बड़ी मदद मिली | 1 | Adjective |
| बुद्धिमान | बुद्धिमान की बात सही साबित हुई | 1 | Adjective |
| प्रकार | वह विभिन्न प्रकार के व्यवसाय करता है | 1 | Adjective |
| प्रकार | इस प्रकार से ही हिंदी को बहुत अधिक मान्यता मिलती है | 3 | adverb |
| प्रकार | उसने उसी प्रकार से प्रतिक्रिया दी | 3 | adverb |
| उदार | उनका स्वभाव बड़ा ही उदार है | 1 | Adjective |
| उदार | पंडितों ने उदार से वेद पढ़ रहे हैं | 1 | Adjective |

*Error Analysis*

To comprehensively evaluate the limitations of the fine-tuned mBERT model for Hindi Word Sense Disambiguation (WSD) both quantitative and qualitative analyses are conducted. The confusion matrix shown in Figure (10), offers a detailed view of the model's classification behavior across the four principal grammatical categories: Noun, Adjective, Verb, and Adverb. Complementing this quantitative perspective, a case-level analysis are performed to examine the model's handling of contextually ambiguous words, particularly those exhibiting overlapping syntactic or semantic roles.

The confusion matrix reveals that the model achieves high overall accuracy; however, certain misclassification patterns are noteworthy. Within the noun category, a considerable number of instances are misclassified as adjectives (426 cases) or verbs (257 cases). This indicates the model's difficulty in resolving nominalized forms and differentiating constructs that share similar morphological features, a common characteristic in Hindi. 97 adjectives are incorrectly labeled as nouns an error likely attributable to the flexible word order in Hindi and Many adjectives can use as nouns in certain syntactic contexts. In contrast, the model exhibits strong performance in distinguishing verbs and adverbs. Only 11 verb

instances are misclassified, typically as nouns, indicating relatively minor confusion. The adverb category shows no misclassifications, suggesting that adverbs possess more distinct lexical boundaries and contextual clarity, making them easier for the model to identify accurately within sentence structures.

These findings highlight that the mBERT model performs robustly overall, challenges remain in differentiating between closely related grammatical forms particularly nouns and adjectives due to the inherent linguistic ambiguity and morphological richness of the Hindi language. Addressing these issues may require integrating additional syntactic features or using POS specific training enhancements in future work.

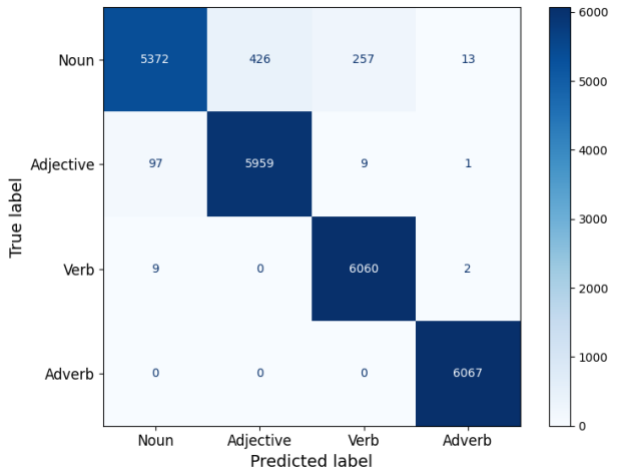


**Fig. 10:** Confusion Matrix Showing the Performance of the mBERT Model on Hindi WSD

To better understand the sources of misclassification, a qualitative review is conducted on nine representative error cases involving contextually ambiguous words, tokens that can belong to multiple POS depending on their usage. These examples are presented in Table (11) and highlight scenarios where the model's interpretation deviates from the correct grammatical categorization due to overlapping syntactic or semantic roles.

Several of the misclassified instances involve words that have both nominal and adjectival use. The words 'जरा', 'बेहतर', and 'सुदर्शन' can used as either as nouns or adjectives, depending on sentence structure and contextual information. In these cases, the model often defaults to the more frequent or statistically dominant meaning, leading to incorrect predictions. Similarly, the word 'फेंकना' typically a verb can be used nominally in certain correct contexts. However, the model tends to favor its more common verbal usage, resulting in classification errors.

Additional challenges arise with words such as 'तेलुगु' and 'तुर्की', which may use as adjectives (describing language or origin) or as nouns (referring to ethnic groups or languages themselves). The model's confusion in these cases reflects the semantic duality and context-dependent nature of such words, which complicate disambiguation.

These examples collectively underscore the limitations of even advanced transformer-based models when dealing with linguistic polysemy and morphologically flexible languages like Hindi. Future improvements may involve incorporating syntactic information, contextual embeddings from surrounding sentences, or using external linguistic knowledge bases to enhance disambiguation accuracy in such cases.

**Table 11:** Misclassifications of Ambiguous Words

| Sentence | Target Word | True Label | Predicted Label |
|---|---|---|---|
| मुझे जरा विश्वास | जरा | Noun | Adjective |
| पता कहाँ रमेश चार सौ रुपए फेंक दिए | फेंकना | Noun | Verb |
| चौकीदार छात्रावास बंद मुख्य द्वार खोला | बंद | Noun | Adjective |
| लगातार बेआराम काम रहता | बेआराम | Noun | Verb |
| पंडालों सुदर्शन प्रतिमाएँ भव्य थीं | सुदर्शन | Noun | Adjective |
| मैं शर्तिया कहता हूँ जल्द ठीक जाएँगे | शर्तिया | Adjective | Noun |
| कल सुबह आइएगा उत्तर बेहतर | बेहतर | Adjective | Noun |
| तुम तेलुगु लोगों संस्था सदस्य क्यों बने | तेलुगु | Adjective | Verb |
| तुर्की टोपी तुम्हें कहाँ मिली | तुर्की | Adjective | Noun |

## Conclusion

This paper presents a Hindi Word Sense Disambiguation (WSD) approach that effectively integrates the lexical richness of Hindi WordNet with the contextual strengths of a fine-tuned BERT Multilingual (mBERT) model. A novel contribution of our work lies in the POS-based dataset balancing strategy, which ensures a more representative distribution across four key POS verbs, nouns, adjectives, and adverbs enhancing both learning stability and disambiguation accuracy. To balance performance and efficiency, we used a selective fine-tuning approach, updating only the first two and last two transformer layers while freezing the rest. We experimented with both Adam and RMSprop optimizers to evaluate training dynamics comprehensively. Our model achieves a 96.48% accuracy, representing a 3% improvement over prior methods, making it one of the most effective approaches reported for Hindi WSD to

date. To better understand which elements of our training pipeline contributed most significantly to performance, we conducted an ablation study, examining the impact of POS-based balancing, optimizer choice, and layer freezing on final accuracy. The results show that POS-based balancing and selective layer fine-tuning are key drivers of improvement, validating our design choices. We ensured the originality of our methodology and presentation by carefully rephrasing background material particularly regarding BERT's architecture and pre-training objectives and referencing all foundational sources. There are several promising directions for future research. One involves domain adaptation, tailoring the model to context-sensitive applications such as healthcare, legal texts, and social media, where word senses can diverge from general usage. Another avenue is addressing code-mixed Hindi-English language, increasingly common in digital discourse, by incorporating multilingual embeddings or joint learning frameworks. Advanced fine-tuning methods, including adapter layers, LoRA, or prompt tuning, could further boost performance while minimizing computational costs. Expanding the dataset to include regional dialects, colloquialisms, and user-generated content will enhance the model's generalization capabilities, ultimately contributing to the development of robust NLP tools for underrepresented languages like Hindi.

## Acknowledgement

The authors express their heartfelt gratitude to the reviewers of this study for their insightful and helpful comments; to the editors of this study for their prompt efforts in managing the manuscript that have considerably helped to enhance the original submission.

## Funding Information

The authors have not received any financial support or funding for this research.

## Author's Contributions

**Shailendra Kumar Patel:** Conceptualization, Data curation, Methodology, Software, Investigation, Formal analysis, Visualization, Writing - Original Draft Preparation, and Writing - Review & Editing.
**Rakesh Kumar:** Supervision, Project administration, and Writing - Review & Editing.
**Anuj Kumar Sirohi:** Conceptualization, Formal analysis, and Writing - Review & Editing.

## Ethics

This article is written adhering to all the ethical standards that are necessary.

## References

Bali, K., Sharma, J., Choudhury, M., & Vyas, Y. (2014). "I am borrowing ya mixing?" An analysis of English-Hindi code-mixing in Facebook. In M. Diab, J. Hirschberg, P. Fung, & T. Solorio (Eds.), Proceedings of the First Workshop on Computational Approaches to Code Switching (pp. 116-126). Association for Computational Linguistics. https://doi.org/10.3115/v1/W14-3914

Parikh, D., & Solorio, T. (2021). Normalization and back-transliteration for code-switched data. In T. Solorio, S. Chen, A. W. Black, M. Diab, S. Sitaram, V. Soto, E. Yilmaz, & A. Srinivasan (Eds.), Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching (pp. 119-124). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.calcs-1.15

Roark, B., Wolf-Sonkin, L., Kirov, C., Mielke, S. J., Johny, C., Demirşahin, I., & Hall, K. (2020). Processing South Asian languages written in the Latin script: The Dakshina dataset. In N. Calzolari et al. (Eds.), Proceedings of the Twelfth Language Resources and Evaluation Conference (pp. 2413-2423). European Language Resources Association. https://aclanthology.org/2020.lrec-1.294/

Gujjar, V., Mago, N., Kumari, R., Patel, S., Chintalapudi, N., & Battineni, G. (2023). A literature survey on word sense disambiguation for the hindi language. Information, 14(9), 495. https://doi.org/10.3390/info14090495

Bhattacharyya, D. P. (2006). Hindi WordNet Data and Associated Software License Agreement. *Indian Institute of Technology, Mumbai, CSE dept., Tchnical Report*.

Clark, K., Luong, M. T., Le, Q. V., & Manning, C. D. (2020). Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*. https://doi.org/10.48550/arXiv.2003.10555

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 8440-8451). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.747

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

*Technologies* (Vol. 1, Long and Short Papers, pp. 4171-4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Eberhard, D. M., Simons, G. F., & Fennig, C. D. (2021). Ethnologue: Languages of the World, 24th End. SIL International.

Liang, Y., Duan, N., Gong, Y., Wu, N., Guo, F., Qi, W., Gong, M., Shou, L., Jiang, D., Cao, G., Fan, X., Zhang, R., Agrawal, R., Cui, E., Wei, S., Bharti, T., Qiao, Y., Chen, J.-H., Wu, W., Liu, S., Yang, F., Campos, D., Majumder, R., & Zhou, M. (2020). XGLUE: A new benchmark dataset for cross-lingual pre-training, understanding and generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 6008-6018). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.482

Jha, P., Agarwal, S., Abbas, A., & Siddiqui, T. J. (2023). A novel unsupervısed graph-based algorithm for Hindi word sense disambiguation. *SN Computer Science*, *4*(5), 675. https://doi.org/10.1007/s42979-023-02116-1

Kumari, A., & Lobiyal, D. K. (2019). Word2vec's distributed word representation for hindi word sense disambiguation. In *International Conference on Distributed Computing and Internet Technology* (pp. 325-335). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-36987-3_21

Mishra, N., & Siddiqui, T. J. (2012). An investigation to semi supervised approach for HINDI word sense disambiguation. *Trends in Innovative Computing 2012-Intelligent Systems Design*, *12*, 126-130.

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of machine learning research*, *18*(17), 1-5.

Navigli, R. (2009). Word sense disambiguation: A survey. *ACM computing surveys (CSUR)*, *41*(2), 1-69. https://doi.org/10.1145/1459352.1459355

Hadiwinoto, C., Ng, H. T., & Gan, W. C. (2019). Improved word sense disambiguation using pre-trained contextualized word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 5297-5306). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1530

Sharma, P., & Joshi, N. (2019). Knowledge-Based Method for Word Sense Disambiguation by Using Hindi WordNet. *Engineering, Technology & Applied Science Research*, *9*(2), 3985-3989. https://doi.org/10.48084/etasr.2596

Pires, T., Schlinger, E., & Garrette, D. (2019). How multilingual is BERT Multilingual?. *arXiv preprint arXiv:1906.01502*. https://doi.org/10.18653/v1/P19-1493

Sarika, & Sharma, D. K. (2016). Hindi word sense disambiguation using cosine similarity. In *Proceedings of International Conference on ICT for Sustainable Development: ICT4SD 2015 Volume 2* (pp. 801-808). Singapore: Springer Singapore. https://doi.org/10.1007/978-981-10-0135-2_76

Sinha, M., Reddy, M. K., Bhattacharyya, P., Pandey, P., & Kashyap, L. (2004). Hindi word sense disambiguation. *International Symposium on Machine Translation, Natural Language Processing and Translation Support Systems*, Delhi, India.

Tripathi, P., Mukherjee, P., Hendre, M., Godse, M., & Chakraborty, B. (2021). Word sense disambiguation in hindi language using score based modified lesk algorithm. *Int. J. Com. Dig. Sys*, *10*(1). https://doi.org/10.12785/ijcds/100185

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*. https://doi.org/10.48550/arXiv.1706.03762

Wu, S., & Dredze, M. (2020). Are all languages created equal in BERT Multilingual? In *Proceedings of the 5th Workshop on Representation Learning for NLP* (pp. 120-130). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.repl4nlp-1.16