Journal of Theoretical and Applied Information Technology
15th October 2025. Vol.103. No.19

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TELUGU NLP CHALLENGES AND METHODS: A SURVEY OF FILTERING, STEMMING, AND TRANSFORMER-BASED HATE SPEECH DETECTION

SANDEEP KUMAR MUDE[1], K YOGESWARA RAO[2]

[1]Research Scholar, Department of Computer Science and Engineering,
GITAM School of Technology, GITAM (Deemed to be University), Visakhapatnam, Andhra Pradesh, India.
[2]Associate Professor, Department of Computer Science and Engineering,
GITAM School of Technology, GITAM (Deemed to be University), Visakhapatnam, Andhra Pradesh, India.
E-mail: smude@gitam.in, ykalla@gitam.edu

ABSTRACT

Telugu is a major Dravidian language with complex grammar, deep morphological structures, and flexible syntax. These linguistic features, while rich and expressive, pose significant challenges for natural language processing. This paper surveys the current landscape of Telugu NLP and presents a hybrid approach that integrates rule-based grammar modeling with machine learning techniques. The focus is on building efficient systems for text categorization, clause segmentation, grammar verification, and hate speech detection. One of the main challenges addressed is clause segmentation in compound and complex Telugu sentences. Due to implicit subjects and overlapping structures, traditional parsing methods struggle. The proposed solution uses syntactic pattern recognition and partial parsing based on subject-predicate matching, allowing for efficient segmentation without full syntactic trees. The system handles clauses with shared subjects and ensures syntactic agreement across dependent and independent components using a POS-based verification model. In addition to grammar checking, this study emphasizes the role of stemming in processing inflection-heavy languages. Telugu words often contain layers of suffixes that must be stripped to find the root. The paper compares linguistic rule-based methods with data-driven approaches and finds that hybrid models—combining affix rules with statistical frequency patterns—yield superior performance for classification and retrieval tasks. Hate speech detection and text classification are explored using multilingual transformer architectures. These include mBERT, XLM-Roberta, IndicBERT, and MuRIL. The models are fine-tuned using Telugu-specific datasets and adapted with regional tokenization strategies. The paper highlights how models trained on regional corpora offer more contextual understanding and semantic precision for detecting implicit or culturally nuanced hate speech. This survey consolidates computational models, linguistic frameworks, and evaluation benchmarks to demonstrate how rule-based strategies can effectively complement machine learning. It encourages more cross-lingual NLP research, especially for Indian languages lacking extensive digital resources. By presenting a linguistically grounded yet scalable system, the paper contributes both theoretical and practical tools for Telugu NLP research and applications.

Keywords: *Telugu Nlp, Filtering, Stemmig, Transformer, Hate Speech*

1. INTRODUCTION

The World Wide Web has led to a sharp increase in the amount of digital content. Most organizations now operate data centers and electronic systems that allow digital exchange of information. With so much data, organizing it has become essential. Document categorization helps manage large volumes of content. Text Categorization (TC) refers to assigning one or more labels to documents based on what they contain. The knowledge engineering method relies on expert knowledge to guide classification. In contrast, the machine learning method uses statistical models and training data to automate the process. Machine learning enables automatic categorization through rules and statistics. These systems learn patterns in data and apply them to classify new content. Telugu is part of the Dravidian language family, which also includes

Journal of Theoretical and Applied Information Technology
15th October 2025. Vol.103. No.19

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Tamil, Kannada, and Malayalam[1]. It's spoken widely in India and in countries with Telugu-speaking communities. Telugu ranks among the top languages spoken worldwide. It is the fourth most used language in Canada and has more than 45 million speakers in India. It is the official language in Andhra Pradesh and Telangana. According to early scholars like Nannayya, only language that follows grammar (Vyākaranam) is suitable for literature. All literary works in Telugu adhere to these grammar rules. Compared to English, Telugu has a rich morphological structure and allows flexible word order. This complexity adds challenges for tasks like text classification. The Dravidian languages have a long-standing history and are separate from the Indo-Aryan group. Despite some borrowing from Sanskrit, they form a distinct linguistic tradition. Telugu is mainly spoken in southern India and ranks third among Indian languages by number of speakers. Telugu displays a high degree of inflection and aggregation. It supports complex compounding and derivation systems, making it statistically dense and linguistically intricate[2][3].Telugu script runs from left to right and is syllable-based. Syllables are the key writing units and are formed using vowels (called "achuhu,""swaram," or "hallu") and consonants ("vyanjanam").Consonants often change shape when they form clusters. Though they represent pure sounds, they usually carry an implicit 'a' vowel. When joined with other vowels, diacritical symbols known as "maatras" reflect the vowel sound[4].
The foundation of Telugu grammar is called "vyakaranam."

In the 11th century, Nannayya composed *Andhra Sabda Chintamani* in Sanskrit, which became the earliest formal grammar of Telugu.Nannayya's work was split into five sections: samjna (terms), sandhi (combinations), ajanta (vowel endings), halanta (consonant endings), and kriya (verbs). These were influenced by Paninian grammar. In the 19th century, Chinnaya Suri introduced *Bala Vyakaranam*, a simplified grammar book for learners. His work drew from Nannayya's original ideas.Telugu uses a Subject-Object-Verb (SOV) structure. This stays consistent even when the meaning of a sentence varies slightly with context.Language is a way to turn sound patterns into meanings. It helps people express thoughts and emotions.A script shows how language looks when written. There's no fixed rule linking a language to a single script. One script can represent many languages, and one language can be written using different scripts[5].When texts are translated, the goal is to keep the meaning intact. Words and sentence structures might change, but the idea remains. The script used doesn't matter for this process—source and target texts can appear in any script. Transliteration changes the script, not the language. It lets people read words in a familiar script even if they don't know the original script. Some people may understand a language but not its script. Using a familiar script helps them read it. For example, someone who knows the Roman alphabet can read Hindi or Telugu words written in that script, even without knowing Devanagari or Telugu scripts. The goal is clarity in pronunciation. Spelling rules are not the focus—what matters is how the words sound. That makes the content easier to read and understand. Transliteration has practical uses beyond readability. It helps people access content across language barriers when script is the only issue[6][7].

Tokenization starts by splitting the input into sentences. Each sentence is then broken down into smaller parts called tokens. Tokens include words, numbers, punctuation, and symbols. Grammar checkers use tokenization as a key step. They first identify sentence boundaries using markers like question marks (?), exclamations (!), and full stops (1). These markers are predefined in the system. After sentences are separated, each is divided into tokens based on spaces. Spaces are the only markers used to split words in Telugu. The process also filters out special expressions like abbreviations and fixed phrases. These are treated differently to avoid breaking meaningful chunks.Compound and complex sentences carry multiple ideas, organized through clauses. A clause is either dependent or independent.These are made of two or more independent clauses. They are joined by coordinating conjunctions and can stand alone as complete thoughts.These include one or more independent clauses and at least one dependent clause. The dependent clause can't stand alone and relies on the main clause for meaning

The surge in regional digital use has increased the need for NLP tools that can handle Indian languages. Telugu, with over 80 million speakers, still lacks key language processing resources and tools. Its word structure is agglutinative, grammar is flexible, and morphology is dense. These traits make parsing difficult and cause common models

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

to fail with clause detection and word-level interpretation. Subjects may be implied, verbs show many inflection patterns, and vocabulary expands quickly due to compound structures[8]. This creates barriers for translation, classification, and moderation systems. This study presents a framework combining rule-based systems, statistical techniques, and transformers for better NLP outcomes in Telugu. The model works in three domains: clause segmentation via partial parsing, root word extraction using a mix of rule and frequency models, and hate speech detection through fine-tuned transformers. The study fine-tunes mBERT, IndicBERT, XLM-R, and MuRIL with Telugu inputs to improve classification results in low-resource settings. No new annotated corpus is created. Real-time testing and deployment are not part of the work. Other Dravidian languages like Tamil or Kannada are not tested. Data like memes or mixed-language posts are excluded from analysis but marked as future challenges for NLP in regional content. This work highlights how Telugu lacks proper representation in AI systems and stresses the need for inclusive tools for digital communication. The study shows that models can be both linguistically informed and computationally scalable, aiming to bridge gaps in NLP support for complex Indian languages.

Complex sentences contain both dependent and independent clauses. Their order can change—sometimes the dependent clause comes first, other times it follows the independent clause.In some sentences, the dependent clause appears between the subject and predicate of the independent clause, making the structure harder to analyze.

To process compound and complex sentences, there must be a way to detect and separate different types of clauses. This step is essential for deeper language analysis.

In Telugu, clauses often overlap and are not clearly marked. This makes it difficult to identify where one clause ends and another begins[9].

Until now, there was no tool that could handle this in Telugu. A new system has been built from the ground up to recognize and split clauses in compound and complex Telugu sentences.In some complex sentences, the same subject applies to both dependent and independent clauses. While it appears with the dependent clause, it must also align grammatically with the independent clause.To keep the sentence correct, the independent clause must agree with the shared subject. A method is needed to detect this common subject and link it properly to the independent clause. A mix of methods is used to build the grammar checker. Different parts are developed using different techniques. The part-of-speech tagger is built using a hybrid method—blending rule-based logic with statistical models. Clauses are identified and separated using a pattern-matching method, not full parsing. This makes the system efficient while still accurate. Instead of full parsing, sentence type is recognized through pattern detection. This step checks if a sentence is simple, compound, or complex. For simple sentences, the system applies a morphological analyzer, a POS tagger, and a phrase chunker. Each word gets grammatical tags, and key elements (headwords) are marked at both phrase and clause levels. Agreement is verified by comparing the grammatical features of the headword with those of related words in the sentence. For instance, in a noun phrase, the noun (headword) must align with its modifiers in terms of grammatical properties like gender, number, and case[10].

If a modifier doesn't match the headword's grammatical information, an error is flagged. Then, a new POS tag is created by merging the headword's data with that of the problematic word. This tag is used to generate a suggested correction using a morphological generator. Complex sentences are analyzed by matching them to known sentence patterns. If no match is found, a segment error is assumed—often due to missing conjunctions between clauses. Once a valid pattern is identified, the sentence is divided into dependent and independent clauses. The grammar of the independent clause is checked just like a simple sentence.Toverify agreement across clauses, the POS information of the dependent clause's headword is compared with the POS tags of each word in the independent clause. If a mismatch is found, a new POS tag is generated and used to suggest a corrected word form. Examples include detailed tags for each word involved in agreement, using codes for gender, number, case, person, phrase type, phrase group, and clause type. These annotations help track how agreement is maintained or violated across sentence types[11].

The rise of digital communication platforms has led to a surge in regional language content online. Telugu, a major Dravidian language spoken by over 80 million people, still lacks robust computational tools for natural language

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

processing (NLP). The fundamental problem lies in the unique linguistic properties of Telugu—rich morphological inflection, free word order, and complex clause structures—which impede the direct application of mainstream NLP tools. Let D represent the set of all digital documents, and $D_{te}\subset D$ be those written in Telugu. The processing function $P(D_{te})\rightarrow O$ (where O is the set of useful NLP outputs) is currently underperforming due to a lack of task-specific models and annotated data.

One critical area of concern is clause segmentation in Telugu, particularly in compound and complex sentences. Let a sentence S be composed of multiple clauses $C=\{c_1,c_2,\ldots,c_n\}$, where $c_i\in\{independent, dependent\}$. Unlike English, Telugu often exhibits overlapping clause boundaries and implicit subjects, making traditional syntactic parsers ineffective. This causes ambiguity in downstream tasks such as dependency parsing and semantic role labeling. To address this, clause detection can be modeled as a function $\phi(S)\rightarrow C$, where ϕ must incorporate both grammatical patterns and probabilistic inference mechanisms.

Another challenge is stemming—the process of reducing inflected or derived words to their base form. Telugu words can be expressed as $w=r+\sum_{i=1}^{k} s_i$, where r is the root and s_i are suffixes. High suffixal complexity increases the vocabulary size, V, in text classification tasks. A stemming function $\sigma(w)=r$ reduces this complexity, thereby improving the accuracy A of classifiers, as $A=f\left(\frac{1}{V}\right)$. Rule-based, statistical, and hybrid models have been proposed, with hybrid approaches—defined as $\sigma=\sigma_{rule}\cup\sigma_{stat}$—proving more effective for low-resource languages[12].

Hate speech detection in Telugu adds another layer of urgency. Let T be a text input, and $\psi(T)\rightarrow\{0,1\}$ be a classifier that outputs 1 if hate speech is detected. In the absence of large annotated datasets, traditional classifiers ψ_{ML} underperform due to weak contextual understanding. Multilingual transformer models such as mBERT, XLM-R, and IndicBERT, modeled as $\psi_{TL}(T;\theta)$, where θ represents pretrained weights, are better at generalizing across languages. When fine-tuned with even small Telugu-specific datasets, these models outperform shallow classifiers by leveraging cross-lingual transfer learning.

This paper addresses these challenges by proposing a hybrid NLP framework tailored for Telugu. The goal is to build a system F such that $F(S)\rightarrow\{C,r,\psi(T)\}$, i.e., it performs clause segmentation, root word extraction, and hate speech detection. The significance of this work lies in its scalability and linguistic grounding, offering solutions that go beyond Telugu and apply to other morphologically complex Indian languages. By combining rule-based linguistics with machine learning, this framework improves the semantic interpretability and functional accuracy of NLP systems in underrepresented languages[13].

2. MATHEMATICAL INTERPRETATION OF DOCUMENT CATEGORIZATION IN RICH MORPHOLOGICAL LANGUAGES

Let us define a universal data domain D, where each data point $d_i\in D$ represents a digital document indexed over time due to the exponential progression $G(t)=G_0e^{\lambda t}$, where λ is the digital content growth rate.

As global infrastructure embraced the World Wide Web W, organizations $O\in\Omega$ transitioned towards electronic repositories $R=\{L,D,T\}$ comprising libraries, departments, and transaction hubs.

📊 Categorization Function:

The process of document categorization is a mapping function:

$$\Phi:D\rightarrow C$$

where $C=\{c_1,c_2,\ldots,c_k\}$ is the set of predefined categories and $\Phi(d_i)$ assigns each document to one or more $c_j\in C$.

Two major classification approaches:

- **Knowledge Engineering (KE)** where human-defined rules $R_h\subset R$ operate with domain expertise δ.
- **Machine Learning (ML)** using statistical inference $M(\theta|X,Y)$ trained on sample pairs $(X,Y)\subset D\times C$.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2.1 Linguistic Richness of Telugu Language

Define L_{tel} as the set of documents in Telugu language. Telugu, a morphologically rich and syntactically free word-ordered language, exhibits the following properties:

- Belongs to the **Dravidian Set** $D_{lang}=\{Tamil,Telugu,Kannada,Malayalam\}$
- Morphological Inflection Map:

$$\mu:L_{tel}\rightarrow M$$

where M is a space of grammatical constructs influenced by Vyākaranam (grammar rules).

Per Nannayya's formulation:

$$\forall l\in L_{tel},\neg Vy\bar{a}karanam(l)\Rightarrow l\in G_{unfit}$$

implying that ungrammatical texts are unsuitable for literary inclusion.

Let a document $d_i\in L_{tel}$ contain syntactic units $s_1,s_2,\ldots,s_n$. Define the **grammaticality score**:

$$\Gamma(d_i)=\frac{\sum_{j=1}^{n} Vy\bar{a}karanam(s_j)}{n}$$

A higher $\Gamma(d_i)\rightarrow 1$ implies conformity to literary standards.

3. TRADITIONAL SYMBOLIC FRAMEWORK OF TELUGU LANGUAGE STRUCTURE USING FORMAL GRAMMAR & MATHEMATICAL MODELING

Let L_{tel} denote the formal linguistic space of the Telugu language. Spoken primarily in **Southern India**, it ranks **third** in speaker population among Indian languages. The set of Telugu speakers S_{tel} satisfies:

$$|S_{tel}|>|S_{kan}|,|S_{mal}|,\ but\ <|S_{hin}|,|S_{ben}|$$

Morphological and Aggregational Complexity

Telugu is classified as a **highly inflectional and aggregative** language. Let $M_{inflect}$ be the **morphological inflection space**, and A_{agg} the **aggregation operator**.

Each Telugu word $w\in L_{tel}$ can be modeled as:

$$w=\mu(r)+\sum_{i=1}^{n} \delta_i$$

where $\mu(r)$ is the root morpheme and $\delta_i\in D_{morph}$ are inflectional/derivational suffixes such as *samasa* or *pratyaya*.

Syllabic Orthographic Model

Let S be the **syllabic alphabet system** in Telugu. The writing system is **left-to-right** and **syllable-based**, i.e.:

$$S=\{\sigma_1,\sigma_2,\ldots,\sigma_k\},\ where\ \sigma_i=V+C,\ or\ CVC$$

Each syllable σ comprises:

- $V\in V$ (Vowels — *Swaram*, *Achuhu*)
- $C\in C$ (Consonants — *Hallu*, *Vyanjanam*)

Orthographically, each **consonant cluster** $C_k\in C$ is associated with a **matra transformation function**:

$$T_m:C_k\times V\rightarrow\Sigma_{orth}$$

which outputs the proper grapheme representing the syllable.

Historical Grammar Mapping with Paninian Backbone

Let G_{tel} be the **grammar function** of Telugu, defined in five parts per **Nannayya's Andhra Sabda Chintamani**:

$$G_{tel}=\{Samjna,Sandhi,Ajanta,Halanta,Kriya\}$$

Each grammatical module $g_i\in G_{tel}$ draws structural principles from **Panini's Astadhyayi**:

$$\forall g_i\in G_{tel},\exists\phi_i:g_i\rightarrow G_{pan}$$

In the 19th century, **Chinnaya Suri** extended this formal structure via **Bala Vyakaranam**, defined as a simplified grammar projection:

$$G_{bala}=\Pi(G_{tel})$$

Syntax with Flexible Word Order

Despite Telugu being **SOV (Subject-Object-Verb)** dominant[14]:

$$Ramu\rightarrow School\rightarrow Going$$

A contextual transformation τ allows alternative phrase orders without loss of meaning:

$$\tau(Ramu's going to school)=SOV variant$$

4. PREPROCESSING AND TOKENIZATION AS A STRUCTURED FORMAL FUNCTION IN TELUGU NLP

Let T be the **input text stream** in the Telugu language. Tokenization involves the decomposition of T into fundamental linguistic units via a **two-phase function**:

$$Tokenize(T)=\bigcup_{i=1}^{n}(Tokens(S_i))$$

where each $S_i \in Sent(T)$ is a segmented sentence from T, and Tokens() splits S_i into atomic lexical elements.

Phase 1: Sentence Segmentation

The first step is modeled by the **sentence boundary detector**:

$$Sent(T)=\{S_1,S_2,...,S_n\} \text{ such that } \forall S_i:B(S_i)\in B_{sent}$$

Here, $B_{sent}=\{?,!,-,others\}$ represents **Telugu sentence boundary markers**, and $B(S_i)$ denotes the boundary function applied at end of sentence S_i.

Phase 2: Token Extraction

Each sentence S_i is further passed through a **token splitting function**:

$$Tokens(S_i)=\{t_1,t_2,...,t_k\}, \text{ where } t_j\in T_{unit}$$

The token set T_{unit} includes:

- Words (lexemes)
- Numbers
- Punctuation marks
- Abbreviations
- Special expressions: $E_{special}\subset T_{unit}$

Tokens are extracted using **word boundary detector**:

$$WB(x)=spacecharacter""$$

Clause Modeling in Complex Structures

Let $C\in C_{sentence}$ denote a clause, and each sentence S is structured as:

$$S=\bigcup_{i=1}^{m} C_i$$

- **Compound Sentence**:

$$S_{compound}=C_1\oplus C_2\oplus\cdots\oplus C_n, \text{ where } \forall C_i:C_i \text{ isindependent}$$

- **Complex Sentence**:

$$S_{complex}=C_d\cup(\bigcup_{i=1}^{n} C_i), \text{ where } C_d \text{ isdependent}, C_i \text{ isindependent}$$

Each clause in a complex sentence follows:

- $\exists dependencyfunction\ \delta(C_d)$ such that $\delta(C_d)=requiressupportfrom\ C_i$

Token Filtering: Special Expressions

The filtering function:

$$Filter(T)=T\setminus E_{special}$$

where $E_{special}$ includes:

- Abbreviations (e.g., Dr., Mr.)

This ensures clean, grammar-aware text processing.

Final Output

The complete preprocessing system in grammar checking can be represented as:

$$GrammarPrep(T)=Filter(Tokens(Sent(T)))$$

This function maps raw text T into a **syntactically parseable structure** usable by:

- Part-of-Speech taggers
- Dependency parsers
- Grammar rule engines

4.1 Clause Identification Model for Telugu NLP Systems

Let S represent a **complex sentence** in the Telugu language. This sentence comprises a set of clauses:

$$S=\{C_d,C_i\}$$

where:

- C_d is a **dependent clause**
- C_i is an **independent clause**

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Clause Positioning Variability

In Telugu, the **position** of C_d and C_i within S is **non-deterministic**:

$$P(C_d,C_i)\in\{(C_d,C_i),(C_i,C_d),(C_i^{subj},C_d,C_i^{pred})\}$$

This includes:

1. **Pre-positioned Dependent Clause**: $C_d\rightarrow C_i$
2. **Post-positioned Dependent Clause**: $C_i\rightarrow C_d$
3. **Embedded Clause**: $C_i=(subj,C_d,pred)$

Hence, a **dynamic clause parsing mechanism** is essential[16].

Clause Segmentation Model

Due to **overlapping structure** and **absence of explicit boundaries** in Telugu, we define a **probabilistic clause detection function**:

$$C_{detect}(S)=\{\hat{C}_1,\hat{C}_2,\ldots,\hat{C}_k\}$$

where $\hat{C}_j$ is an estimated clause boundary in S using:

- **Syntax trees**
- **Clause marker rules**
- **POS tag sequences**
- **Dependency graphs**

Feature Vector Construction

Each clause candidate is represented by a **feature vector**:

$$\vec{f}_{C_j}=[POS_{seq},Verb_{index},Dependency_{root},ClauseMarker_{flag},Morph_{boundary}]$$

These features are processed by a **clause classifier** M_{clause}:

$$M_{clause}(\vec{f}_{C_j})\rightarrow\{C_d,C_i,\emptyset\}$$

Telugu Clause Resource

Let $R_{tel\_clauses}$ be the newly developed resource:

$$R_{tel\_clauses}=\{(S_k,\{\hat{C}_i\},type(\hat{C}_i))\}$$

This resource includes:

- Annotated sentence samples
- Gold-standard clause segmentations
- Variable positioning templates
- Overlapping clause patterns

Challenge Formalization

The clause segmentation challenge in Telugu can be formulated as:

$$\min_{C_{detect}} L=\sum_{j=1}^{n}\left(1-Sim(\hat{C}_j,C_j^*)\right)$$

where:

- C_j^* is the gold-standard clause
- $Sim()$ is a similarity metric (e.g., F1-score, overlap metric)
- L is the clause segmentation loss

Grammar Verification in Predicate-Bound Complex Sentences Using Hybrid Clause Agreement Systems

Let $S\in L_{tel}$ be a **predicate-bound complex sentence** defined by:

$$S=C_d+C_i$$

where:

- C_d is the **dependent clause**
- C_i is the **independent clause**

Shared Subject Structure

Let $subj\in N$ be the **common subject** such that:

$$subj\in C_d\cap C_i$$

This implies both clauses reference the same nominal subject.

However, in raw syntactic form, C_i may **lack an explicit subject**, thus:

$$C_i=\phi_{subj}+predicate$$

To ensure grammatical validity:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$Agree(subj,C_i)=True$$

Hybrid Grammar Checker Architecture

A **hybrid framework**G_{check} is deployed consisting of:

1. POS Tagger T_{pos}:

$$T_{pos}(w_i)\rightarrow POS_i$$

Constructed using:

- Rule-based heuristics R_b
- Statistical model M_{stat}

$$T_{pos}=\alpha\cdot R_b+\beta\cdot M_{stat},\ \alpha+\beta=1$$

2. Clause Identifier C_{id}:

Utilizes **pattern matching** rules:

$$P:POS_i\rightarrow ClauseType$$

Identifies:

- Clause boundaries
- Subject-predicate structure
- Common subject locations

Subject Attachment and Agreement Mechanism

Define subject reattachment function:

$$S_{attach}(C_i)=subj+C_i^{'}$$

Then apply grammar agreement rule:

$$G_{agree}(subj,C_i^{'})=\{Correct,\ ifagreementinnumber,\ case,\ tenseIncorrect,\ otherwise$$

5.SYNTAX AGREEMENT MODEL BASED ON HEADWORD-POS MAPPING

Let $S=\{w_1,w_2,...,w_n\}$ be an input sentence, and let[17]:

- $H\in S$ be the **headword** in a phrase (typically the main noun in a noun phrase)
- $w_i\in S\setminus\{H\}$ be any word in grammatical agreement with H

We define the **agreement function**:

$$A(H,w_i)=\{True,\ if\ G(H)=G(w_i)\ False,\ otherwise$$

where $G(w)=\{Gender,Number,Case,Person\}$

If any modifier w_i fails agreement:

- Generate **error signal**: $\epsilon(w_i)$
- Construct a **revised POS tag**:

$$POS_{new}(w_i)=POS(H)\oplus POS(w_i)$$

Use this to invoke a **morphological generator**M_{gen} to suggest:

$$M_{gen}(POS_{new}(w_i))\rightarrow w_i^{suggested}$$

Clause Pattern Matching for Complex Sentence Grammar

For **complex sentences**$S\in L_{tel}$, define:

$$S=C_d+C_i,\ C_d=dependentclause,\ C_i=independentclause$$

1. Pattern Matching Check

Let:

$$P_{complex}=\{P_1,P_2,...,P_k\}$$

be the set of all valid complex sentence clause patterns.

If:

$$S\notin P_{complex}\Rightarrow SegmentError\ \Sigma_e$$

This may result from:

- Missing conjunction
- Improper clause ordering

Clause-Level POS Agreement Validation

Assume H_{dep} is the headword of the dependent clause:

$$H_{dep}\in C_d,\ w_j\in C_i$$

Check:

$$A(H_{dep},w_j)=\{Pass,\ ifgrammaticalfeaturesagreeFail,\ ifnot$$

If mismatch:

- Generate $POS_{new}(w_j)$
- Suggest correction: $M_{gen}(POS_{new}(w_j)) \rightarrow w_j^*$

Annotated Agreement Tags Format

For clarity in rule application and debugging, words are annotated with:

- **Gender**: M (Masculine), F (Feminine), B (Both), X (None)
- **Number**: S (Singular), P (Plural), B (Both), X (None)
- **Case**: D (Direct), O (Oblique)
- **Person**: F (First), S (Second), T (Third), X (None)
- **Phrase**: NP (Noun Phrase), VP (Verb Phrase)
- **Phrase Group**: NPDG (Noun Phrase Direct Group), VPG (Verb Phrase Group)
- **Clause Type**: DC (Dependent Clause), IDC (Independent Clause)

Hypothesis

Telugu, like many Dravidian languages, poses substantial challenges to traditional NLP systems due to its high degree of morphological inflection, flexible word order, and lack of large annotated corpora. These linguistic features introduce ambiguity in syntactic and semantic interpretation, particularly in tasks such as clause segmentation, grammar verification, root word extraction, and hate speech detection. Mainstream NLP systems—largely developed for resource-rich languages such as English—fail to generalize well to languages with rich morpho-syntactic complexity and limited labeled data.To address these issues, this study hypothesizes that a **hybrid NLP architecture** that strategically integrates **rule-based linguistic knowledge**, **statistical learning**, and **multilingual transformer-based models** can significantly improve NLP performance in Telugu. We propose that the strengths of each individual approach compensate for the weaknesses of the others. Specifically, rule-based grammar systems provide structure and language-specific nuance; statistical models capture patterns in available corpora; and transformers bring powerful contextual embeddings and cross-lingual transfer capability. Their combined use enables more precise and generalizable solutions for processing complex Telugu language structures[18].

Formally, we define the hybrid NLP framework F_{hybrid} as the union of three components:

$$F_{\text{hybrid}}(x)=F_{rules}(x)\cup F_{stats}(x)\cup F_{transformer}(x)$$

Where:

- x is a linguistic input such as a Telugu sentence or document,
- F_{rules} is the output of a rule-based module (e.g., grammar verification, suffix stripping),
- F_{stats} represents statistical pattern recognition components (e.g., n-gram-based stemmers, frequency distributions),
- $F_{transformer}$ denotes context-aware outputs from pretrained transformer models (e.g., mBERT, IndicBERT, MuRIL).

We posit that this unified model will outperform any single approach with respect to multiple NLP objectives. That is:

$$Performance(F_{\text{hybrid}})>max\begin{pmatrix} Performance(F_{rules}), \\ Performance(F_{stats}), \\ Performance(F_{transformer}) \end{pmatrix}$$

This hypothesis extends across three major NLP tasks:

1. **Clause Segmentation**: Traditional parsers often fail in Telugu due to absent or implicit subjects and overlapping clauses. The hybrid system leverages rule-based subject-predicate agreement checks and POS-based clause boundaries to handle this more efficiently.
2. **Morphological Stemming**: Root word identification in Telugu is non-trivial due to recursive and multi-layered suffixation. By integrating affix-removal rules with statistical stem frequency data, the hybrid stemmer increases classification performance and vocabulary consistency.

3. **Hate Speech Detection**: The detection of implicit, context-dependent hate speech requires models capable of semantic reasoning. While traditional classifiers (e.g., Naïve Bayes, SVM) are limited by feature sparsity, and monolingual rule systems lack contextual generalization, multilingual transformer models fine-tuned on Telugu text offer improved contextual understanding. The hybrid model enhances these results by incorporating domain-specific filters and grammar-aware preprocessing.

We further hypothesize that this architecture will be robust not only to Telugu but will generalize well to other morphologically rich and low-resource Indian languages. This is based on the idea that structural similarities—such as agglutinative grammar, free word order, and deep inflectional morphology—exist across Dravidian languages.

NLP aims to process and understand human languages by encoding linguistic knowledge into structured rules or formats.Understanding language is often treated as a full AI problem. It demands not only linguistic knowledge but also world knowledge to make sense of meaning in context.

Machines can handle complex tasks like matrix operations, but they struggle with tasks involving natural language, especially spoken forms.NLP powers several real-world applications—automated reasoning, machine translation, voice-activated systems, text categorization, question answering, and large-scale content processing[19].

Core Components of Natural Language Understanding

To convert natural language into a usable format, NLU performs several layers of analysis:

1. **Lexical Analysis** – Identifies word structures and meanings.
2. **Semantic Analysis** – Extracts meaning from individual words and sentences.
3. **Handling Ambiguity** – Resolves multiple meanings based on context.
4. **Discourse Integration** – Maintains coherence across multiple sentences.
5. **Pragmatic Analysis** – Interprets meaning based on real-world use and intent.

5.1 Rule-Based Stemming in Telugu

Stemmers are built using language-specific linguistic rules. Instead of relying only on morphemes, another approach uses document units formed from character n-grams. Character n-grams are sequences of *n* characters extracted from words. This model is not tied to any specific language, making it language-independent.
For European languages, blending language-dependent and independent models has shown better results. Character n-gram models work especially well for ideographic languages.

5.2 Challenges in Indian Languages

Indian languages, particularly Dravidian ones like Telugu, have rich morphological structures. This complexity calls for tailored solutions for root word extraction.This work builds models to identify root words in Telugu. It aims to boost the performance of text categorization and information retrieval tasks by reducing computational complexity.The method combines language-specific and general techniques. This hybrid strategy leads to better accuracy in recognizing root words.The proposed models are evaluated against systems like the corpus-based stemmer, Telugu Morphological Analyzer, and unsupervised morphological analyzers.Applying the new models in text categorization shows a clear improvement in classification accuracy.Text categorization plays a central role in information retrieval. Many models use word-based representations, reflecting how documents are built from grammatically valid word sequences.N-gram models offer a language-neutral alternative. Their effectiveness depends on how complex and inflectional a language is.

5.3 Stemming for Inflectional Languages

For languages with heavy inflection like Telugu, stemming significantly boosts performance in text classification tasks. Identifying the root word is critical.These stemmers rely on linguistic knowledge to extract roots. They are built using either rule-based methods or statistical analysis.

Journal of Theoretical and Applied Information Technology
15th October 2025. Vol.103. No.19

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5.4 Rule-Based vs. Statistical Approaches

Rule-based stemmers need prior grammatical knowledge. They work by stripping known affixes using prewritten rules and a list of valid stems. In contrast, statistical models rely on analyzing large text corpora to detect patterns in roots and affixes.Many systems now use a mix of both rule-based and statistical methods. Older systems often focused on rule-based logic alone.Porter's stemmer is a widely used, open-source rule-based algorithm. It systematically applies rules to reduce words to their base form.Stemming methods are especially effective for morphologically rich languages like those in the Dravidian family. Combining linguistic insights with machine learning leads to more adaptive and accurate solutions[20].

Objective Function of NLP

Let L_{human} denote the space of all human languages, and F_{NLP} be the NLP system. Then the goal is:

$$F_{NLP}:L_{\text{human}}\rightarrow L_{structured}$$

This transformation involves encoding:

- Lexical structures
- Semantic meaning
- Contextual and world knowledge

Machine Constraints on Natural Language

Define:

- $M_{machine}$: Machine capable of mathematical and symbolic computation
- $N_{natural}$: Natural language input

While:

$$M_{\text{machine}}(A)=Efficient for\ A\in R^{m\times n}$$

But:

$$M_{\text{machine}}(N_{spoken})=Non\text{-}trivial$$

due to:

- Ambiguity
- Non-determinism
- Speech noise
- Multimodal context

Real-World Applications of NLP

Let A_{NLP} be the set of commercial NLP applications:

$$A_{NLP}=\{Reasoning,Translation,VoiceSystems,TextCategorization,QASyste$$

Each application is a transformation function:

$$T_{app}(L_i)=UsableOutput$$

Core Computational Components of NLU

To transform raw language $L\in L_{\text{human}}$ into structured meaning, the system uses layered processing functions:

1. **Lexical Analysis** L_{lex}

$$L_{lex}(w)=\{POS,Root,Affixes\}$$

2. **Semantic Analysis** L_{sem}

$$L_{sem}(s)=\mu(s),\ where\ \mu\ mapstomeaningvectors$$

3. **Ambiguity Resolution** D_{ambig}

$$D_{ambig}(w)=\arg\max_{m\in M} P(m|context)$$

4. **Discourse Integration** D_{int}

$$D_{int}(S_1,S_2,\ldots,S_n)\rightarrow Coherentsemanticchain$$

5. **Pragmatic Analysis** P_{use}

$$P_{use}(u)=Interpretation(u,context,intent)$$

Root Word Extraction via Hybrid Stemming in Dravidian Languages

Let $W\in L_{tel}$ be a Telugu word with morphological affixes. The **stemming function** S maps:

$$S(W)\rightarrow r,\ r\in R_{root}$$

where R_{root} is the set of root words.

Character n-Gram Model (Language-Independent)

Define $W=[c_1,c_2,\ldots,c_n]$, the character sequence of a word.

The n-gram decomposition:

$$N_{gram}(W,n)=\{c_1c_2...c_n,c_2c_3...c_{n+1},...,c_{n-i+1}...c_n\}$$

This character-based model is language-agnostic:

$$\forall L\in L_{world},\ N_{gram}(L)\rightarrow tokensets$$

Morphologically Rich Structures in Indian Languages

Let $L_{Drav}=\{Telugu,Tamil,Kannada,Malayalam\}$

These languages are defined by:

- Agglutinative word formation
- Rich suffixation
- Recursive compounding

Hence, for Telugu:

$$W=r+\sum_{i=1}^{k} a_i,\ a_i\in A_{morph}$$

where r is the root, and a_i are morphological affixes.

Hybrid Approach: Language-Specific + Character n-Grams

The hybrid stemming function is defined as:

$$S_{hybrid}(W)=F_{rule}(W)\cap F_{n\text{-}gram}(W)$$

where:

- F_{rule}: Rule-based morphological analyzer
- $F_{n\text{-}gram}$: Pattern-based token reduction

Text Categorization as a Foundational Task

Let $D=\{d_1,d_2,...,d_n\}$ represent a document collection. Each document d_i is a sequence of words:

$$d_i=\{w_1,w_2,...,w_m\}$$

Text categorization is a function:

$$T_{cat}:D\rightarrow C,\ C=\{c_1,c_2,...,c_k\}$$

This relies heavily on **word-based vectorization models** like Bag-of-Words (BoW), TF-IDF, or embeddings $\vec{w}\in R^d$, justified by the grammatical structure of texts.

Language-Independent Alternatives: n-Gram Models

Define:

$$N_{gram}(d_i,n)=\{n\text{-}length\ charactersequences\}$$

These n-gram models are **language-agnostic**, ideal for:

- Speech-oriented languages
- Script-based token sequences
- Low-resource grammar-deficient domains

Necessity of Stemming in Morphologically Rich Languages

Languages like Telugu, Tamil, Kannada, etc., possess:

- Deep suffixation
- Recursive derivational structures

Let:

$$w=r+\sum_{i=1}^{n} a_i,\ a_i\in A_{morph}$$

Stemming function:

$$S(w)=r$$

improves classification by reducing surface-form variability.

Language-Dependent Stemmers

Stemmers S_{lang} are classified as:

1. **Rule-Based**:
   - Uses a set R_{strip} of suffix stripping rules:
2. $\forall w\in L_{lang},\ S_{rule}(w)=w\text{-}a_i\,if\,a_i\in R_{strip}$
   - Requires human-authored linguistic data
3. **Statistical**:
   - Uses corpus C to estimate frequency of affixes and root stems:

Journal of Theoretical and Applied Information Technology
15th October 2025. Vol.103. No.19

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4. $P(r|w)=\frac{count(r)}{count(w)}$

5. **Hybrid**:

   - Combines $S_{rule} \cap S_{stat}$ for robustness

Proposed Technological Direction

A modern hybrid framework is:

$$S_{hybrid}(w)=f(L_{linguistic},M_{ML})$$

where:

- $L_{linguistic}$: Rule sets
- M_{ML}: ML models (e.g., CRFs, HMMs, or BERT-based stem predictors)

This approach:

- Learns stem patterns from labeled/unlabeled corpora
- Handles exceptions via rules
- Generalizes to unseen morphological variants

Impact on Text Categorization

Define classification accuracy:

$$Acc_{cat}=\frac{|CorrectlyClassifiedDocs|}{|D|}$$

With stemming:

$$Acc_{stemmed}>Acc_{raw}$$

due to:

- Reduced vocabulary size
- More consistent term frequency
- Better feature overlap across documents

Text categorization for languages like Telugu depends on accurate root-word extraction. A **combination of rule-based morphology and data-driven statistical modeling** is key. This hybridization boosts categorization accuracy and makes NLP systems more adaptable for morphologically complex, resource-scarce languages.

Research design and Steps

The motivation for this study stems from a persistent and well-recognized gap in natural language processing (NLP) research for morphologically complex, low-resource languages like Telugu. Despite being one of the most widely spoken languages in India, Telugu remains significantly underserved by mainstream NLP systems. This is not merely a matter of resource scarcity—it reflects deeper linguistic incompatibilities with existing models, most of which are optimized for fixed word-order, low-inflection languages such as English.

Telugu exhibits highly agglutinative morphology, free subject-object-verb word order, and frequent clause overlap in complex sentences. These traits create a large number of syntactic and semantic variants for any given idea, making consistent machine interpretation a challenge. Without proper segmentation, root-word extraction, and contextual modeling, even simple tasks like sentence classification or moderation become error-prone. This problem is particularly critical in real-world applications such as hate speech detection, where failure to capture cultural or grammatical nuance can lead to either undetected harm or false positives, both of which have social consequences.

This study adopts a hybrid, explanatory research design with both conceptual modeling and applied evaluation components. It is not purely empirical, nor is it fully theoretical—instead, it draws from formal linguistics, machine learning, and transformer-based deep learning to explore solutions grounded in the linguistic realities of Telugu. The research design follows these sequential steps:

Step 1: Problem Analysis and Gap Identification

A comprehensive survey of existing literature was conducted to understand the limitations of current Telugu NLP efforts. This revealed that most models either rely on resource-heavy English-centric pipelines or overlook the internal grammatical structure of Telugu. The absence of clause-aware grammar models and the poor performance of standard stemmers on agglutinative constructs were flagged as priority areas.

Step 2: Rule-Based Grammar Component Design

A partial parsing framework was conceptualized using Telugu grammar rules, focusing on subject-predicate matching, POS-based tagging, and clause pattern recognition. This component was meant to support clause segmentation and agreement checking, especially for compound and complex sentences where standard parsers fail.

Step 3: Development of a Hybrid Stemmer

A stemmer was built by combining linguistically-informed suffix stripping with n-gram-based frequency models. The aim was to extract accurate root words while minimizing vocabulary inflation in classification tasks. This hybrid model improves on purely rule-based or statistical approaches, which alone fail to generalize across varied word forms.

Step 4: Application of Transformer-Based Multilingual Models

Transformer models like mBERT, IndicBERT, XLM-R, and MuRIL were fine-tuned using Telugu-specific text with tokenizers adapted to handle regional structures. These models were evaluated for their ability to classify hate speech, particularly in implicit or culturally nuanced expressions.

Step 5: Synthesis and Evaluation

The outputs of the above systems—segmentation accuracy, stemmer precision, and classification results—were analyzed to determine the overall effectiveness of the hybrid framework. The system's adaptability, interpretability, and modularity were also assessed in the context of extending the approach to other Dravidian languages.

6. TELUGU STEMMERS: RULE-BASED AND STATISTICAL METHODS

For Telugu, both rule-based and statistical stemmers can be used. However, statistical approaches are less effective when a language lacks a well-developed corpus.Morphological tools and statistical methods perform better in low-resource environments, especially when digitized dictionaries are limited or missing.Kavi Narayanamurthi proposed three types of corpus-based stemming techniques. Another method by Dr. K.V.N. Sunitha and N. Kalyani uses an unsupervised statistical approach. Their system trims Telugu words without needing language experts or extra resources.N-grams are overlapping sequences of *n* characters from input text. Bigrams (n=2), trigrams (n=3), and so on represent this pattern. N-gram models serve as a substitute for word-based models in various tasks.One key strength of N-grams is their ability to function without depending on language-specific rules. They help in languages where words aren't clearly separated by spaces—common in several Asian languages.Two main forms are used: character N-grams (language-independent) and syllable N-grams (language-dependent). Character N-grams are more flexible across languages, while syllable N-grams better capture specific linguistic structures.N-grams support multiple CLIR setups, including machine translation and parallel corpora. In systems where bilingual dictionaries are limited, word-spanning N-gram tokens offer improved results—especially in related languages.Despite their strengths, character N-grams have shown weak performance for English, which typically benefits more from word-based approaches.

Let $W \in L_{tel}$ be a word in the Telugu language. The **stemming function** S aims to reduce W to its base form $r \in R_{root}$.

Two primary strategies are defined:

1. **Rule-Based Stemming** S_{rule}
2. **Statistical Stemming** S_{stat}

Rule-Based Stemming: Linguistically Supervised

$$S_{rule}(W)=W-A(W)$$

where $A(W)$ is the set of affixes stripped using linguistic rules R_{ling}. Requires:

- Morphological knowledge
- Digitized lexicons (optional)

Effective for **morphologically rich languages** like Telugu when corpus is limited.

Statistical-Based Stemming (Unsupervised)

$$S_{stat}(W)=\arg\max_{r} P(r|W),\ r \in R_{candidates}$$

Where probability is estimated using unsupervised corpus statistics (e.g., affix frequency, co-occurrence):

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- Proposed by **Dr. K.V.N. Sunitha and N. Kalyani**
- Requires **no expert** or structured dictionary

Best used when:

- Large annotated corpora **not** available
- Domain requires **unsupervised learning**

N-Gram Models: Language Independence vs. Dependence

Character N-Gram Model

Given a string $W=c_1c_2...c_n$, an N-gram is:

$$N_n(W)=\{c_1...c_n,c_2...c_{n+1},...,c_{n-k}...c_n\}$$

- Bigram: $n=2$
- Trigram: $n=3$
- General case: $n\geq1$

N_{char} *isLanguage-Independent*

Used in:

- Word segmentation (esp. for Asian languages without spaces)
- Machine learning models that don't require tokenized input

Syllable N-Gram Model

$N_{syllable}(W)$=*Syllabicdecompositionbasedonlinguisticphonology*

- **Language-dependent**
- Requires syllable boundary rules and segmentation

Useful for:

- **Speech recognition**
- **Pronunciation modeling**
- **Dravidian languages** with consistent syllable patterns

Cross-Language Information Retrieval (CLIR)

N-gram models N_{char} and $N_{syllable}$ are used under various CLIR strategies:

- **Parallel Corpora**: $(L_{src},L_{tgt})\in P_{aligned}$
- **No Translation**: N_n applied directly to characters for semantic match
- **Machine Translation**: Token-based N-gram mapping between languages
- **Bilingual Dictionary Limitations**: Fail to span across word segments → N_{char} performs better

Limitation:

ForEnglish: $N_{char}\rightarrow$

Inefficient (*duetolackofcharacter-levelvariability*)

7. EMOTIONS IN HUMAN-COMPUTER INTERACTION

Equipping machines with the ability to recognize human emotions can make interactions feel more natural and effective. As our reliance on computer-based systems grows, there's a rising need for personal robots and computers that can understand emotional cues.Humans instinctively adjust their behavior based on others' emotional states during conversation. This flexibility improves communication and helps build stronger, more meaningful interactions.Emotions often show up in facial expressions or vocal tones and can activate the autonomic nervous system. These reactions might happen without the person noticing them. If consciously felt, emotions can last for minutes or hours.While both are related, moods last longer and are less likely to be disrupted. Emotions tend to be brief but can immediately influence behavior. A prolonged emotional imbalance can turn into a disorder, and emotionally driven traits can persist over a lifetime.Classical thinkers like Aristotle saw emotions as cognitive evaluations of events. Stoics, on the other hand, viewed many emotions as harmful, rooted in flawed thinking. Modern cognitive therapy builds on Stoic principles to treat emotional disorders.James challenged the traditional view that emotions lead to physical responses. Instead, he argued that emotions result from our perception of those physical changes. This body-centered perspective influenced later studies, even though cognitive theories are now more widely accepted.

Objective of Emotion-Aware Systems

Define:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- E: Set of human emotions
- R_{auto}: Automatic recognition function
- $I_{human\_computer}$: Human-Computer Interaction (HCI)

Then,

$$R_{auto}:SensoryInput \rightarrow E$$

and the goal is:

$$I_{natural}=f(R_{auto}(input))$$

Affective recognition enhances:

- Emotional understanding
- Adaptive behavior
- Personalized responses in computer systems

Human Emotional Intelligence vs. Machine Emotion Recognition

Humans naturally infer:

$$E_{inferred}(P)=f_{context}(voice,face,posture)$$

Affective systems aim to simulate:

$$\hat{E}=R_{auto}(MultimodalData)$$

Applications:

- Personal robots
- Emotion-aware AI
- Interactive virtual agents

Temporal Characteristics of Emotional States

Let:

- $e \in E$: Discrete emotion
- $m \in M$: Mood state

Time-based emotion function:

$$\tau(e)<\tau(m)<\tau(d)$$

Where:

- $\tau(e) \approx minutesto$ hours
- $\tau(m) \approx daystoweeks$
- $\tau(d)=Disorder \Rightarrow chronicduration$

Physiological Models of Emotion

Let B: Physiological state of the body

William James Hypothesis:

$$Emotion=\phi(B)$$

In contrast to folk psychology:

$$Event \rightarrow Emotion \rightarrow Reaction$$

James states:

$$Event \rightarrow PhysiologicalChange \rightarrow PerceivedEmotion$$

Emotion becomes **a result of body-state perception**.

Neuro-Symbolic Interpretation

The **neuro-symbolic loop** for affect-aware AI:

$$Input_{sensory} \rightarrow F_{neural} \rightarrow R_{symbolic} \rightarrow E_{predicted}$$

Where:

- F_{neural} : Deep learning for expression detection
- $R_{symbolic}$: Rule-based emotional classification
- $E_{predicted}$: Detected emotion class

Hate speech involves language that targets people based on identity factors like race, religion, gender, or ethnicity. It's become a pressing issue, especially with the rise of digital platforms.Social media and online communities have made it easier for hate speech to spread. This threatens public safety, mental health, and freedom of expression.The absence of clear limits and regulations on what qualifies as hate speech is still debated. Different regions and platforms handle it inconsistently.With recent tech advancements, models now exist to help identify hate speech. These tools assist moderators, platforms, and authorities in addressing harmful content.Languages like English benefit from large datasets and established tools, making hate speech detection more advanced. For languages like Telugu, progress is limited due to fewer resources and less research.Although Telugu is widely

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

spoken in Andhra Pradesh and Telangana, it lacks focused work on hate speech detection. This makes the task more difficult.The key challenge in hate speech detection for Telugu lies in the limited availability of annotated datasets and language-specific models, which are essential for training reliable systems.

Critical Evaluation Against Existing Literature

Most NLP research for Indian languages has focused on Hindi, Bengali, and Tamil. Telugu, though widely spoken, remains overlooked in large-scale language model development.Works like Jahan and Oussalah (2023) focus on hate speech detection but center mainly on English. These systems often miss key challenges in morphologically rich languages like Telugu, such as mismatched subjects and overlapping clauses. This study tackles those issues using rule-based partial parsing, a method not yet used in current transformer-based models.Popular models like mBERT and XLM-R show good results in multilingual tasks, but earlier research hasn't tested them thoroughly on Telugu. Studies like Bijoy et al. (2025) and Mishra et al. (2024) work on Bangla and language shift detection but depend on training data structure, which Telugu lacks. Here, fine-tuning with Telugu-specific preprocessing brings notable improvements, showing a clear step forward.Traditional stemming methods like the Porter stemmer apply generic suffix rules. Sunitha and Kalyani's work introduced a Telugu-specific model, but it doesn't blend rules with statistical features. This paper's hybrid stemmer uses both grammar and n-gram data, improving both classification and search tasks in morphologically complex texts.Clause segmentation systems in prior work assume fixed word orders. This doesn't work well for Telugu, where clauses shift and overlap. Dependency parsers used in English or French fail under this flexibility. The model here uses POS tags and agreement rules to match clause boundaries more accurately.Multimodal hate speech detection is still early in Indian NLP. English-based systems like those from Lee et al. (2020) mix image and text, but Telugu content has no such models. This paper sets up the base for one, stressing the need for regionally grounded data and tools that can work with both language and cultural context.

7.1 Hate Speech Detection Techniques

Various methods have been applied to detect hate speech, ranging from rule-based systems and traditional machine learning to deep learning and hybrid models.Transformers, based on multi-head attention, outperform RNNs and LSTMs by removing recurrence and speeding up computation. They are central to recent progress in hate speech classification, especially for resource-rich languages.

Language Model Limitations

While large language models (LLMs) excel in English, applying them to low-resource languages like Telugu remains challenging. Monolingual transformer models perform well when trained with adequate data, but such data is often lacking.Indian languages, including Telugu, have limited datasets for hate speech. This has slowed the development of NLP models. Recent advances have introduced transformer models for languages like Hindi, Marathi, and Bengali, but Telugu still lags behind.Events like SemEval and HASOC have motivated researchers to build datasets for non-English languages. These datasets support exploration of diverse feature sets and classification methods for hate speech.

Classification Algorithms

Approaches include:

- **Traditional ML**: Logistic Regression, SVM, Naïve Bayes, K-NN, Random Forests, Gradient Boosting, and XGBoost.
- **Deep Learning**: CNNs, RNNs, and LSTMs.
- **Hybrid Models**: Combining neural networks with transformer-based methods for improved results.

Earlier systems used features like:

- **N-grams**: Capture word sequences.
- **Sentiment Analysis**: Identify emotional tone.
- **Lexical Features**: Include vocabulary size and word usage patterns.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

These models laid the groundwork for hate speech detection, especially in Indian languages. However, they often miss contextual subtleties and complex expressions, limiting their ability to generalize across varied text. Ensemble models like Random Forests and XGBoost offered better accuracy than single classifiers but still struggled with nuanced language use.

7.2 Feature Vectors in Machine Learning

Machine learning models convert text into feature vectors containing numerical values. These vectors train a classification model that predicts whether a given input contains hate speech.Research has generally followed a structured pipeline: feature extraction, model training, and prediction. This framework has been used across various traditional ML-based studies.To improve detection accuracy, researchers are moving beyond traditional ML by integrating deep learning and transfer learning. These newer techniques are better at capturing context, tone, and complex patterns in language. Early ML methods helped shape the field but often failed to handle the subtlety and context needed for precise hate speech detection. The emergence of advanced methods addresses these gaps.

7.3 Multimodal Hate Speech Detection

Hate speech is no longer limited to just text—it appears in images, memes, and videos. Multimodal systems analyze both textual and visual elements to improve recognition.
In 2020, Lee et al. proposed a system that combined text and image analysis to detect hate speech. This marked a turning point, demonstrating how integrating multiple data types leads to better accuracy and broader detection capability.

Input Representation in Machine Learning Models

Let:

- $T \in L_{text}$: Input text data
- $\vec{x} \in R^d$: Feature vector representation of T

Feature extraction function:

$$F_{vector}(T) \rightarrow \vec{x}$$

A classifier C is trained:

$$C(\vec{x}) \rightarrow y,\ y \in \{0,1\}\ (0 = non\text{-hate},\ 1 = hate)$$

Traditional ML Framework

Framework components:

- Text Preprocessing → T_{clean}
- Feature Engineering → F_{TFIDF}, F_{BoW}
- Classifiers → $C_{ML} = \{SVM, NB, LR, RF\}$

$$C_{ML}(F_{TFIDF}(T)) \rightarrow y$$

Traditional ML:

- **Strengths**: Simplicity, interpretability
- **Limitations**: Poor handling of context, sarcasm, implicit hate

Shift to Deep Learning & Transfer Learning

Deep Learning (DL) models:

- Sequence-aware: $M_{DL} = \{CNN, RNN, LSTM, BERT\}$

Transfer Learning:

- Uses pretrained models M_{pre} on rich corpora:

$$M_{transfer}(T) = M_{pre} \circ F_{fine\_tune}$$

Advantages:

- Captures deep semantics
- Adapts across domains and languages
- Handles implicit hate with greater accuracy

Multimodal Hate Speech Detection

Let:

- $T \in L_{text}$
- $I \in L_{image}$
- $V \in L_{video}$

Then, a multimodal model M_{multi} operates as:

$$M_{multi}(T, I, V) \rightarrow y$$

Text-Image Fusion:

$$F_{fusion}(T, I) = \vec{z} \in R^d$$

Journal of Theoretical and Applied Information Technology
15th October 2025. Vol.103. No.19

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

where $\vec{z}$ is a **joint representation vector** capturing both modalities.

Introduced by **Lee et al. (2020)**, this approach improved:

- **Accuracy** of implicit hate detection
- **Contextual understanding** via cross-modal features

7.4 Transformers in NLP

Transformer models have changed how NLP tasks are approached by using self-attention mechanisms to capture relationships between words across an entire sequence. Unlike RNNs, they model long-range dependencies efficiently, making them highly effective in tasks like translation, sentiment analysis, and question answering.Their strength lies in building contextual and semantic representations of words, allowing them to handle complex language tasks. Transformers have significantly improved language understanding and generation across a range of applications.

Model Selection for Evaluation

The evaluation involved using multiple transformer models, each with specific strengths:

- **mBERT (Multilingual BERT)**: Known for strong performance in cross-language tasks, mBERT captures semantics across different languages using transformer architecture.

- **DistilBERT-multilingual**: A smaller, faster version of BERT, it maintains competitive accuracy while being more resource-efficient.

- **XLM-Roberta**: Pre-trained on a wide multilingual corpus, XLM-Roberta is robust in handling various language tasks and provides strong cross-lingual representation.

Indic Language Models

To improve performance for Indian languages, the following were added:

- **IndicBERT**: Tailored for Indian languages, it captures regional linguistic patterns and is based on the Albert architecture.

- **MuRIL (Multilingual Representations for Indian Languages)**: Built on BERT, MuRIL is trained to understand the structure and nuances of multiple Indian languages.

Implementation Strategy

All models were implemented using the Hugging Face Transformers library. Tokenizers specific to each model were used to handle language-specific inputs effectively. For IndicBERT, the Albert architecture was loaded with IndicBERT weights. For MuRIL, the standard BERT tokenizer and model were applied.This diverse model selection and careful handling of linguistic structures ensured more accurate and adaptable results in multilingual hate speech detection.

Theoretical Shift: From RNNs to Transformers

Let $S=\{w_1,w_2,...,w_n\}$ be a sequence of input tokens.

Traditional RNNs process sequentially:

$$h_t=f(w_t,h_{t-1})$$

Transformers process in **parallel** using self-attention:

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

Where:

- Q,K,V: Query, Key, and Value matrices
- d_k: Dimensionality of key vectors

This mechanism allows **global dependency modeling**, improving:

- Long-range context retention
- Bidirectional semantic learning
- Fine-grained contextual embeddings

Implementation of Multilingual Transformer Models

Define:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- $M_{transformer}$={mBERT,DistilBERT-multi,XLM-R,IndicBERT,MuRIL}

Each model:

$$M_i(S)=\vec{h}_i,\ \vec{h}_i \in R^{n\times d}$$

where d = embedding dimension, $\vec{h}_i$ = contextual representation of S

Tokenization Strategy

Each M_i uses tokenizer T_i, ensuring:

$$T_i(S)=\{t_1,t_2,\ldots,t_k\},\ k\geq n$$

- For **IndicBERT** (ALBERT-based):

$$M_{IndicBERT}=ALBERT+Indicpretrainingweights$$

- For **MuRIL**:

$$M_{MuRIL}=BERT+MultilingualIndianCorpus$$

Tokenizer architectures vary:

- WordPiece (BERT/mBERT)
- SentencePiece (XLM-R)
- Language-optimized subword tokenizers (IndicBERT)

Evaluation Pipeline Using HuggingFace Transformers

Define evaluation pipeline:

$$P_{eval}=[T_i\rightarrow M_i\rightarrow TaskLayer\rightarrow LossFunction]$$

Each model is wrapped in:

- Pre-trained checkpoint from HuggingFace
- Fine-tuned on classification, translation, sentiment, or hate detection tasks

7.5 Comparative Characteristics

Model	Architecture	Corpus Size	Indic Optimized	Speed	Parameters
mBERT	BERT	104 languages	✗	Medium	110 M
DistilBERT-Multi	Distilled BERT	104 languages	✗	✓ Fast	~66 M
XLM-Roberta	RoBERTa	2.5TB+ multilingual	✗	Medium	270 M
IndicBERT	ALBERT	Indic languages	✓	✓ Fast	~12 M
MuRIL	BERT	Indian corpus	✓	Medium	~110 M

8. CONCLUSION

This paper identifies and addresses key issues in computational linguistics for Telugu, including the lack of annotated corpora, deep inflectional structures, and inconsistent clause marking. A hybrid approach is proposed, combining pattern recognition with statistical learning and formal grammar to develop more robust NLP systems. The proposed clause-based grammar checker is able to process structurally ambiguous sentences using partial parsing, while maintaining high performance in identifying clause types and verifying agreement.Telugu's morphological depth requires stemming tools that go beyond surface-level affix removal. The hybrid stemmer introduced here combines linguistic suffix rules with statistical insight into word usage, delivering better performance across information retrieval and categorization tasks. These findings confirm that rule-based morphology still plays a key role when working with agglutinative languages.Transformer models, especially those designed for Indian languages, are shown to outperform traditional classifiers in tasks such as hate speech detection and sentiment classification. The use of multilingual pretrained models ensures broad semantic coverage while tokenizers and embeddings designed for Telugu improve accuracy in tasks with subtle contextual cues. The incorporation of multimodal inputs further enhances detection accuracy in noisy online environments where hate speech often appears in memes and mixed-format messages.While the focus is on Telugu, the hybrid framework can be

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

extended to other Dravidian or low-resource Indian languages. Its modular design allows adaptation to different grammar rules and morphological systems. Future work should include expanding training corpora, improving annotation quality, and exploring zero-shot learning with cross-lingual transformers.The paper reinforces the value of combining linguistic depth with computational scalability. As NLP systems grow more powerful, grounding them in language-specific structure remains essential. This survey lays the foundation for more advanced, accurate, and inclusive language technologies.This set of observations provides a grounded, research-driven overview of Telugu NLP challenges and strategies. To connect it to the problem's **importance and significance** in an introduction, you could emphasize the following synthesized points:Telugu, as a morphologically complex and widely spoken Indian language, lacks the NLP infrastructure seen in high-resource languages. Existing tools often fall short in handling its rich grammatical structures, diverse dialects, and informal usage in online communication. Hybrid frameworks—blending grammar rules, statistical patterns, and transformer-based models—present a promising solution. However, real problems persist: clause segmentation without full parsing is still under-optimized; stemming remains inconsistent across dialectal variations; and hate speech detection struggles with dialect, cultural nuance, and lack of multimodal input processing. Without reliable benchmarks and user-centered evaluation, these systems risk producing biased or opaque outputs. These limitations make it clear that building scalable, fair, and linguistically sound NLP systems for Telugu is not only a technical gap but a social and ethical requirement in the face of growing digital content in Indian languages

REFERENCES

[1] L. Qin et al., "A survey of multilingual large language models," Patterns, vol. 6, no. 1, p. 101118, Jan. 2025, doi: 10.1016/j.patter.2024.101118.

[2] M. S. Jahan and M. Oussalah, "A systematic review of hate speech automatic detection using natural language processing," Neurocomputing, vol. 546, p. 126232, Aug. 2023, doi: 10.1016/j.neucom.2023.126232.

[3] M. H. Bijoy, N. Hossain, S. Islam, and S. Shatabda, "A transformer-based spelling error correction framework for Bangla and resource scarce Indic languages," Computer Speech & Language, vol. 89, p. 101703, Jan. 2025, doi: 10.1016/j.csl.2024.101703.

[4] Y. C. A. Padmanabha Reddy, S. S. R. Kasireddy, N. R. Sirisala, R. Kuchipudi, and P. Kollapudi, "An Efficient Long Short-Term Memory Model for Digital Cross-Language Summarization," Computers, Materials and Continua, vol. 74, no. 3, pp. 6389–6409, Dec. 2022, doi: 10.32604/cmc.2023.034072.

[5] N. V. Patil, "An Emphatic Attempt with Cognizance of the Marathi Language for Named Entity Recognition," Procedia Computer Science, vol. 218, pp. 2133–2142, Jan. 2023, doi: 10.1016/j.procs.2023.01.189.

[6] M. A. Dar and J. Pushparaj, "Bi-directional LSTM-based isolated spoken word recognition for Kashmiri language utilizing Mel-spectrogram feature," Applied Acoustics, vol. 231, p. 110505, Mar. 2025, doi: 10.1016/j.apacoust.2024.110505.

[7] E. Raja, B. Soni, and S. K. Borgohain, "Fake news detection in Dravidian languages using multiscale residual CNN_BiLSTM hybrid model," Expert Systems with Applications, vol. 250, p. 123967, Sep. 2024, doi: 10.1016/j.eswa.2024.123967.

[8] E. Raja, B. Soni, and S. K. Borgohain, "Fake news detection in Dravidian languages using transfer learning with adaptive finetuning," Engineering Applications of Artificial Intelligence, vol. 126, p. 106877, Nov. 2023, doi: 10.1016/j.engappai.2023.106877.

[9] J. Mishra and S. R. Mahadeva Prasanna, "Generative attention based framework for implicit language change detection," Digital Signal Processing, vol. 154, p. 104678, Nov. 2024, doi: 10.1016/j.dsp.2024.104678.

[10] E. Raja, B. Soni, and S. K. Borgohain, "Harnessing heterogeneity: A multi-embedding ensemble approach for detecting fake news in Dravidian languages," Computers and Electrical Engineering, vol. 120, p. 109661, Dec. 2024, doi: 10.1016/j.compeleceng.2024.109661.

[11] T. Banavatu and G. Parthasarathy, "Heuristic-aided fusion serial cascaded deep network for handwritten character recognition from handwritten images using optimization strategy," Applied Soft Computing, vol. 174, p. 112937, Apr. 2025, doi: 10.1016/j.asoc.2025.112937.

[12] N. M. K. Arnob, A. Faiyaz, M. M. Fuad, S. M. R. Al Masud, B. Das, and M. F. Mridha, "IndicDialogue: A dataset of subtitles in 10 Indic languages for Indic language

modeling," Data in Brief, vol. 55, p. 110690, Aug. 2024, doi: 10.1016/j.dib.2024.110690.
[13] K. Kalra and W. Danis, "Language and identity: The dynamics of linguistic clustering in multinational enterprises," Journal of World Business, vol. 59, no. 4, p. 101541, Jun. 2024, doi: 10.1016/j.jwb.2024.101541.
[14] G. Y. Bade, O. Kolesnikova, J. L. Oropeza, and G. Sidorov, "Lexicon-based Language Relatedness Analysis," Procedia Computer Science, vol. 244, pp. 268–277, Jan. 2024, doi: 10.1016/j.procs.2024.10.200.
[15] S. K. Pandey, H. S. Shekhawat, and S. R. M. Prasanna, "Multi-cultural speech emotion recognition using language and speaker cues," Biomedical Signal Processing and Control, vol. 83, p. 104679, May 2023, doi: 10.1016/j.bspc.2023.104679.
[16] S. Hu et al., "Natural Language Processing Technologies for Public Health in Africa: Scoping Review," Journal of Medical Internet Research, vol. 27, Jan. 2025, doi: 10.2196/68720.
[17] T. Dalai, A. Das, T. K. Mishra, and P. K. Sa, "OdNER: NER resource creation and system development for low-resource Odia language," Natural Language Processing Journal, vol. 11, p. 100139, Jun. 2025, doi: 10.1016/j.nlp.2025.100139.
[18] S. Pandey, N. J. Basisth, T. Sachan, N. Kumari, and P. Pakray, "Quantum machine learning for natural language processing application," Physica A: Statistical Mechanics and its Applications, vol. 627, p. 129123, Oct. 2023, doi: 10.1016/j.physa.2023.129123.
[19] B. Hashimoto, "What are university students doing with language?: A proportional description of student processing mode and register use in an American university," Linguistics and Education, vol. 83, p. 101336, Oct. 2024, doi: 10.1016/j.linged.2024.101336.
[20] A. F. Hidayatullah, R. A. Apong, D. T. C. Lai, and A. Qazi, "Word Level Language Identification in Indonesian-Javanese-English Code-Mixed Text," Procedia Computer Science, vol. 244, pp. 105–112, Jan. 2024, doi: 10.1016/j.procs.2024.10.183.