International Journal of Engineering Trends and Technology
ISSN: 2231-5381/ https://doi.org/10.14445/22315381/IJETT-V74I3P117

Volume 74 Issue 3, 228-247, March 2026

Original Article

FNB-T5 Linguistically Informed Neural Translation of Code-Mixed Text

Surinder Pal Singh[1], Neeraj Mangla[2]

[1, 2]*Department of Computer Science & Engineering, MMEC, Maharishi Markandeshwar (Deemed to be University), Mullana-Ambala, Haryana, India.*

[1]*Corresponding Author : 01spsingh.oct77@gmail.com.*

Received: 26 November 2025 Revised: 29 January 2026 Accepted: 06 February 2026 Published: 28 March 2026

Abstract - *Code-mixed text and Hinglish in particular, pose considerable machine translation problems as it is an informal language, whose spelling is not universal, and often there is a language shift within a sentence. Conventional models of Neural Translation can be pretty ineffective in this field because they are linguistically grounded and have issues with ambiguous cases and scanty data. This paper is a proposal of a new hybrid translation system that combines the symbolic and neural translators in order to successfully translate code-mixed Hinglish texts to standard English. The offered system is built with the Finite-State Machine (FSM) of Structural Pattern Recognition, character n-gram similarity of spelling variation, phonetic alignment, and transformer-based Part-of-Speech (POS) tagging of syntactic interpretation. These characteristics are put together as enriched token prompts and sent to a fine-tuned Multilingual Pre-trained Text-to-Text Transformer (mT5) model to be translated. For a comprehensive assessment, the model was trained and evaluated on a carefully curated corpus containing 200,000 Hinglish sentences. Its performance was measured using standard metrics, including BLEU, TER, CHRF, and COMET. Study results indicate that the hybrid model is better than state-of-the-art neural, statistical, and rule-based models on all metrics: BLEU 39.2, Comet 0.74, ter 41.3, and CHRF 66.4. The model is robust in low-resource and noisy conditions, which are commonly affected by deep models. It is shown in this work how hybrid architectures can be helpful in multilingual and informal Natural Language Processing (NLP) systems, and provide a scalable solution to the increasing problem of code-mixed language translation.*

Keywords - *Linguistic Model, Low-Resource Machine Translation, Linguistic Feature Fusion, Neural–Symbolic Learning, Rule-Based and Neural Integration.*

1. Introduction

In the digital age of communication, the multilingual communities are also witnessing a phenomenal increase in the use of code-mixed languages, which is a language development that involves the alternation of languages by speakers within a single utterance or sentence. The traditional machine translation systems, such as rule-based, statistical, or neural, will typically involve monolingual or bilingual information with normalized syntax and vocabulary. The systems will require the use of standard forms of language during training to achieve successful translation performance. These models cannot deal well with the noisy, variant, and fluid code-mixed input [1]. The words used are in Roman handwriting, although they are either of Hindi or English origin, and they keep on varying in phonetics, casual spelling, and grammar.

In code-mixed inputs, it is actually quite important to make use of intelligent preprocessing steps in order to enable your models to process the mixed language information. The importance of effective term extraction and selection through Pseudo-Relevance Feedback (PRF) [2] demonstrates how optimized linguistic feature selection can significantly enhance retrieval and reformulation tasks. Metric learning approaches for lowering Out-of-Vocabulary (OOV) rates in Romanised text, emphasising the importance of similarity-based normalization in resource-poor scripts [3]. FSM is a computationally efficient and robust way of parsing code-mixed input, which becomes quite handy in such situations of transliterated Hindi [4]. The FSM output is fed into a Character N-Gram similarity module that computes the similarity of input tokens with respect to standard Hindi and English dictionaries based on cosine similarity [5]. Module assigns language of origin (LE for English, LH for Hindi) even if there are no exact matches (this also applies to informal spellings, such as “skul” for “school” and “khaana” for “खाना”).

The tagging helps resolve ambiguity in grammar, e.g., “class” (school/noun) versus “classify” (to classify/verb). For ambiguous tokens, there is an ambiguity resolution layer that compares language scores and marks tokens for special

treatment if there is low confidence or equal similarity[6]. The structured prompt is first sent to a specially fine-tuned model of mT5 transformers. This model, in turn, provides the final Translation in English, depending on the details of the input. This enriched prompt, unlike standard mT5 inputs, provides the decoder with more context, as well as disambiguation, and enables the decoder to generate more semantically aligned and grammatically sound output.

To overcome those challenges, sophisticated linguistic preprocessing will be needed to be used so that the raw text can be matched by the structural requirements of neural models. The use of Pseudo-Relevance Feedback (PRF) in choosing words has also been stressed in previous literature, where optimal lexical weighting of words can be used to optimize the extraction of features when performing a text retrieval and reformulation process. Based on these findings, this paper incorporates the feature optimization in the preprocessing stage to strengthen language cues prior to Translation. In a similar fashion, approaches to normalization using metric-learning have been successfully used to reduce the occurrence of Out-of-Vocabulary (OOV) cases in Romanized or low-resource scripts, which allows more stable generation of embedding because of the similarity-based mapping.

The framework of the Finite-State Machine (FSM) is used to memorize and generalize the morphological and syntactic patterns that are employed in texts with code-mixed Hindi and English. The computation of FSM-based parsing is efficient, and transliterated inputs are processed quickly. The determinism of FSM-based parsing offers predictable and consistent language processing. Within the suggested scheme, FSM is used as a rudimentary structural analysis that categorizes and tokenizes a mixed-language feed, introducing it to subordinate statistical applications.

In order to increase resistance to informal spellings and phonetic variation, the FSM results are further refined. Particularly, a Character N-Gram similarity step will be used to identify and match such variable inputs in a more efficient way. Based on the models of vector-space and distributional similarity, this layer uses the method of calculating the cosine similarity between the input tokens and the items in the curated Hindi and English dictionaries. It has a mechanism that enables soft matching, i.e, the variant spelling of school, which can be spelled as skul, or khaana, which can be spelled as khaana, etc., is automatically recognized and defined as such. The tokens get language-of-origin labels (e.g., LH, Hindi, and LE, English), which are further passed to subsequent modules to perform syntactic and semantic alignment.

The results of FSM are improved to deal with the spelling discrepancies and variations based on pronunciation. Particularly, a Character N-Gram similarity step is used to match and identify such variable inputs more efficiently. This eliminates the situations of the lexical overlap between the languages or between languages, or between the languages and parts of speech, e.g., the difference between the terms class (Noun) and classify (Verb), which was previously also experienced in the literature of multilingual tagging. Tokens that have low confidence or other cross-language probabilities are marked to be treated with special care so as to prevent cases of a translation error.

The added and linguistically structured representation is subsequently input into a specialized mT5 transformer model, which does the final Translation into English. In contrast to regular transformer inputs, the presented framework directly provides the encoder-decoder pipeline with auxiliary linguistic annotations, which are obtained using FSM and N-Gram analysis. The hybrid method is helpful in ensuring that the model takes into consideration finer syntactic and pragmatic clues that are usually lost in conventional end-to-end neural translation models. Consequently, the system generates semantically acceptable and grammatically accurate translations, despite extremely informal code-mixed input. The framework blends symbolic processing, statistical methods, and neural context modelling. This balances the clarity of rule-based systems with the natural fluency of deep learning models. This hybridization improves the quality of Translation and, in addition, adds a scalable solution to multilingual communities that facilitates smoother and more natural cross-language communication.

Although several hybrid translation systems have tried to make use of a combination of rule-based normalization and neural models, they tend to use linguistic preprocessing as a separate or non-dynamically-based process. They do not allow the neural network to learn using linguistic signals, nor do they allow interaction at the feature level between symbolic and contextual representations. HIPHET [15], HingBERT [24], and other variations of rule-transformer hybrids have been studied in the past, but they only perform data normalization or concatenation embedding, not actual co-learning.

Unlike other models, FNB-T5 introduces a dynamic in-Prompt Fusion Mechanism. It directly incorporates Structural (FSM), Orthographic (n-gram), and Syntactic (POS) features into the transformer's input. The result of this integration is that the multilingual T5 model is able to rely on language-specific cues during the encoding and decoding processes, and does not rely on surface-level tokens. This neural-symbolic co-learning marks a significant improvement over earlier hybrid models. It creates a system that's both interpretable and flexible enough to handle noisy, code-mixed input. The accumulating literature in code-mixed Natural Language Processing (NLP) is an indicator of the community attempting to reconcile linguistic diversity and computational modelling. At the beginning, the research was mainly devoted to monolingual Translation and sentiment analysis. However, as

the cases of multilingual communication on digital platforms continue to increase, scholars have increasingly studied approaches that are specific to code-mixed data. As already discussed in the literature, finite-state automata, n-gram-based similarity scoring, transformer-based architectures, and others have all helped advance linguistic normalization, feature extraction, and context understanding. Nevertheless, an agreement as to a conceptualized and versatile framework is still lacking despite these developments. It is in the variety of approaches, which include rule-based systems and statistical models, deep learning, and hybrid mechanisms, that point to the complexity of the issue and to the absence of a single solution. Thus, the current work places itself in the context of this developing field of research and critically analyzes previous attempts and suggests a hybrid model consisting of symbolic structure, statistical regularization, and the deep contextual representation to increase translation quality on noisy, code-mixed text.

Beyond proposing an architectural framework, this study conducts a systematic analytical investigation into how each linguistic preprocessing component contributes to translation quality. The impact of FSM-based structural parsing, character n-gram similarity normalization, and in-prompt linguistic fusion is quantitatively evaluated to demonstrate their effectiveness individually and collectively on noisy code-mixed inputs.

2. Literature Review

The literature survey will discuss the results of 41 articles, which investigate diverse code-mixed NLP systems. These will involve language recognition, machine translation, emotion detection, sentiment detection, POS tagging, and hate speech detection. A Hinglish (Hindi-English) is the most widely researched of the learned languages, and next comes Tamil English, Malayalam-English, and Arabic-English. Transformer-based models that are increasingly popular include BERT, RoBERTa, and mT5. This proves the robustness of the deep learning. Still, the conventional rule-based and hybrid methods occupy no less significant places. Such approaches are helpful in structured activities such as Translation and Language Identification (LID). Particularly effective in such cases are models such as DicBERT based on CRF and embeddings. In both studies, information is provided regarding the authors and the year of publication. The studies also report the code-mixed language pairs evaluated, such as Hinglish and Tamil–English, along with the methodologies employed, including rule-based, neural network–based, and hybrid approaches. Additionally, the limitations identified by the respective researchers are documented to highlight existing challenges and research gaps. The recorded constraints indicate the aspects that should be improved, including the regularity of data, the complexity of annotation, and the flexibility of the model. Overall, Table 1 is a brief and informative overview of the advances and issues in code-mixed text studies.

Table 1. Comparative summary of related methods in code-mixed translation

Author(s) & Year	Focus Area	Language(s)	Methodologies Used	Limitations
Attri, S.H. et al. (2021) [7]	Translation of code-mixed language using rules	Hindi-English	Rule-based approach using POS tagging and a dictionary	Manual rule dependency, limited scalability
Srivastava, V. et al. (2020) [8]	Code-mixed corpus creation for Machine Translation (MT)	Hinglish	Parallel corpus, manual annotation	No translation model provided
Ameer, I. et al. (2022) [9]	Emotion classification on code-mixed text	Roman Urdu-English	ML, DL, BERT	Transformers underperformed
Yong, Z.X. et al. (2023) [10]	Generating code-mixed text using LLMs	Southeast Asian languages	Prompting BLOOMZ, Flan-T5, Chat Generative Pre-Trained Transformer (ChatGPT)	Inconsistent results across languages
Chakravarthi, B.R. et al. (2020) [11]	Sentiment analysis for Dravidian languages	Tamil-English	Baseline models, dataset release	Baseline accuracy, less deep learning
Chakravarthi, B.R. et al. (2020) [12]	Sentiment dataset for code-mixed Malayalam-English	Malayalam-English	Dataset annotation	Limited model benchmarking
Attri, S.H. et al. (2020) [13]	Hybrid Translation for code-mixed Hinglish	Hindi-English	Rule-based + statistical hybrid	Low generalization for informal language
Shanmugavadivel, K. et al.	ML models for Tamil code-mixed sentiment	Tamil-English	Support Vector Machine (SVM), Random Forest (RF)	Small datasets, limited

(2022) [14]				generalizability
Shanmugavadivel, K. et al. (2022) [15]	DL for sentiment and offensive detection	Multilingual	NN, Long Short-Term Memory (LSTM)	Language-specific tuning required
Ojo, O.E. et al. (2022) [16]	Word-level language ID in code-mixed text	Kannada-English	Character sequences, word embeddings	Decreased accuracy in noisy inputs
Priyadharshini, R. et al. (2020) [17]	Named Entity Recognition (NER) on an Indian code-mixed corpus	Indian languages (code-mixed)	Meta embeddings, ML classifiers	Embedding dependency, dataset size
Hande, A. et al. (2020) [18]	Sentiment and offensive detection for Kannada	Kannada-English	Corpus creation, baseline models	Dataset annotation complexity
Agarwal, V. et al. (2021) [19]	Machine translation for Hinglish to English	Hinglish	Multilingual transformers	Context loss in informal text
Panchendrarajan, R. (2022) [20]	DL review for social media code-mixed text	Various Indian	Review of DL models	Lack of implementation evaluation
Nayak, R. et al. (2022) [21]	Hinglish corpus and BERT model (HingBERT)	Hinglish	Corpus construction, BERT	Performance drops in noisy text
Ansari, M.Z. et al. (2021) [22]	LID for Hinglish tweets using BERT	Hinglish	Code-mixed BERT model	Requires annotated datasets
Roy, S. et al. (2023) [23]	Survey on sentiment analysis for Indo-Aryan	Indo-Aryan	Transformer-based survey	Limited task-specific benchmarks
Kumari, N. et al. (2023) [24]	Hindi text summarization using seq2seq	Hindi	Seq2seq neural networks	Not tested on code-mixed data
Jagtap, N.S. et al. (2023) [25]	Improvised Neural Machine Translation (NMT) model for Hinglish	Hinglish	Neural machine translation	Domain-specific tuning needed
Mangla, A. et al. (2024) [26]	Language ID and normalization for code-mixed text	Hinglish	Hybrid methods (rules + transformers)	Performance drops in domain-specific slang
Chopra, A. et al. (2023) [27]	Online hate speech detection for Hindi-English	Hindi-English, Devanagari	Hybrid Machine learning (ML) framework	Ambiguity, limited labelled data
Gupta, V. et al. (2023) [28]	Temporal information retrieval with query reformulation	Unspecified	Contextual reformulation	Not specific to code-mixed text
Liu, Y. et al. (2020) [29]	Multilingual NMT with denoising pretraining	Multilingual	Multilingual Bidirectional and Auto-Regressive Transformers style training	Needs a large multilingual corpus
Tang, Y. et al. (2021) [30]	Translation via denoising pretraining	Multilingual	Multilingual NMT with pretraining	Underexplored on code-mixed data
Kamboj, S. et al. (2022) [31]	Denoising and Translation for NMT	Multilingual	Joint denoising and Translation pretraining	Trade-off in multilingual balance
Bhattu, S.N. et al. (2020) [32]	POS tagging using code-mixed embeddings	Hindi-English	Embedding-driven POS model	Limited generalizability
Absar, S. et al. (2025) [33]	Cross-lingual Large Language Model (LLM) for POS tagging	Code-switched data	Fine-tuning LLMs	Data scarcity in cross-lingual settings
Van Der Goot, R. et al. (2020) [34]	Lexical normalization in code-switched data	Code-switched (unspecified)	Normalization models	Dependent on noisy lexicons

Saikrishna, V. et al. (2022) [35]	Hierarchical FSMs for multilingual text	Multilingual	Probabilistic FSMs	Not suited for deep semantics
Svete, A. et al. (2023) [36]	Language modelling with neural FSMs	Multilingual	Recurrent Neural Network (RNNs) as probabilistic automata	Complexity in training
Postiglione, A. et al. (2024) [37]	Efficient text mining with finite state automata	Multilingual	Multi-word Finite State Automata (FSA)-based models	Low performance on informal text
Jawahar, G. et al. (2021) [38]	English-Hinglish MT with synthetic code-mixing	English-Hinglish	Text-to-text transformers	Relies on synthetic data quality
Xue, L. et al. (2020) [39]	mT5: multilingual pre-trained transformer	Multilingual	Text-to-text transformer	Not evaluated on code-mixed text
Krasitskii, M. et al. (2025) [40]	Sentiment analysis in Tamil-English code-mixed	Tamil-English	Transformer-based models	Ambiguity and limited labelled data
Nagoudi, E.M.B. et al. (2021) [41]	MT for Arabic-Egyptian code-mixed text	Arabic-English	Transformer-based MT	Contextual inconsistency in code-mixing
Mundra, S. et al. (2023) [42]	Aggression detection in Hinglish text	Hindi-English	Hybrid attention-based network	Context sensitivity and variation
Sudharsan, K. et al. (2022) [43]	Hybrid model for code-mixed text classification	Code-mixed Indian languages	Deep hybrid learning model	Language adaptability required
Sengupta, A. et al. (2021) [44]	Robust code-mixed language representation	Multilingual	Hierarchical attention network	Complex model tuning
Tan, K.L. et al. (2022) [45]	Sentiment analysis using RoBERTa-LSTM	Multilingual	Hybrid transformer + RNN	Resource-intensive fine-tuning

Big generative code-mixed models such as ChatGPT, a fine-tuned version of Text-to-Text Transfer Transformer (Flan-T5), and BigScience Large Open-science Open-access Multilingual Language Model with Zero-shot fine-tuning (BLOOMZ) exhibit mixed results when it comes to generating natural code-mixed text in a wide range of languages. The most essential results of the literature review and later research regarding the current manuscript are presented after the contribution.

All of the reviewed studies point to the development of research on code-mixed Natural Language Processing (NLP) with the primary focus on Hinglish (Hindi-English) data and other pairs like the Tamil-English, Malayalam-English, and Arabic-English. Initial research used rule-based and statistical algorithms to translate, identify language, and analyze sentiments. However, these models were unable to scale to large datasets and were highly reliant on handwritten rules. Transformer models and deep learning algorithms such as BERT, RoBERTa, and mT5 have improved performance. They allow enhanced contextual knowledge and inter-linguistic generalization. In spite of this improvement, there are still issues in dealing with the variability of transliteration, data imbalance, and domain noise. Other hybrid models, especially those that involve programs that take dictionary-based features and contextual embeddings along with sequence labelling (e.g., CRF or BiLSTM-CRF), are more robust but still need to be trained on large annotated datasets to do so. In general, it can be seen that although transformer-based architectures are the focus of recent literature, hybrid and linguistically informed architectures also provide competitive performance on structured tasks such as translation and language identification. These results encourage the current study to develop a hybrid system that is integrated to form a bridging point between the symbolic and neural paradigm, to fill in the gaps of limitation in code-mixed Translation. In reviewed studies, hybrid methods are helpful in the normalization and identification process. However, they are pipeline-like - that is, linguistic modules are regarded as pre- or post-processors, but not as part of learning. The works that are surveyed do not apply concurrent Fusion Of Morphological (FSM), Orthographic (n-gram), and Syntactic(POS) information within a single syntactic transformer framework. Additionally, current models of Hinglish Translation either use handwritten rules with little generalization or neural models, which fail catastrophically when supplied with few resources. This study thus contributes to the field by suggesting a single hybrid framework that is integrated into the neural training process per se, which results in a sustained interaction between rule-based and deep

learning units. FNB-T5 is a trade-off between the importance of the neural models and the simplicity of the symbolic approaches to solving the complex language tasks. It solves the major problems in code-mixed Translation, including ambiguity, variation, and limited data transliteration.

2.1. Novel Contributions

The suggested FNB-T5 model suggests a number of fundamental innovations to the hybrid model. The innovations are superior to the existing neural-symbolic translation methods:

In-Prompt Linguistic Feature Fusion: FNB-T5 became the first translation model that takes Morphological (FSM), Orthographic (n-gram), and Syntactic (POS) specifications as the input prompt of the transformer. This mixture helps the model understand how language is structured and its meaning. It not only learn grammar but learns it with meanings, and this ends up giving more accurate results. This causes it to make better-informed and context-sensitive translations.

Neural–Symbolic Co-Learning Mechanism: In contrast to the classical hybrid models, which use external preprocessing based on rules, the suggested system directly encodes symbolic features into contextual embeddings. This enables the mT5 encoder-decoder to learn linguistic cues during Translation dynamically. As a result, the model is better able to adapt to the form and meaning of the input.

Low-Resource Robustness and Transferability: It has been empirically demonstrated that FNB-T5 achieves high translation quality with only 25-50 percent of the original input. It achieves 5-8 better BLEU points than the standalone mT5 model on the same conditions. This emphasizes the strong and effective performance of FNB-T5. It is flexible to sparse or noisy code-mixed data, including that on social media.

Explainability Through Modular Design: The output of each component, such as FSM, N-gram, and POS tagger, is intermediate results that are interpretable, such as language IDs, similarity scores, and syntactic tags. These products enhance clarity and provide linguistic understanding of the Process of Translation. Meanwhile, the model still enjoys profound contextual fluency through the transformer.

Empirical Advancement Over Prior Hybrids: The quantitative results show that FNB-T5 is better than strong hybrid baselines because the results increase by +3.5 in BLEU scores and a +0.03 gain in COMET. These scores substantiate that it is very effective in lexical and semantic aptitudes. These additions may yield enormous profits in the accuracy and semantic beauty of the Translation. The findings indicate that FNB-T5 can be applied in addressing complex code-mixed input.

These innovations together define FNB-T5 not as an incremental hybrid, but as a new paradigm of linguistically aware neural Translation. Unlike existing hybrid approaches that treat linguistic preprocessing as an external or static normalization step, the proposed framework introduces a dynamic neural-symbolic co-learning mechanism. This design enables the transformer to directly exploit structural, orthographic, and syntactic cues during encoding and decoding, representing a substantive advancement over prior normalization-based hybrid models.

3. Preliminary Study

Hinglish (Hindi-English code-mixed text) is common on social media and poses challenges for Translation. A successful approach must detect which parts are Hindi vs English and leverage linguistic cues for accurate Translation. Below, a hybrid model pipeline is outlined that combines rule-based and neural methods for Hinglish-to-English Translation. To ensure analytical rigor, the proposed system is evaluated through component-wise and ablation-based analysis. Each module-FSM parsing, character n-gram similarity scoring, ambiguity resolution, and in-prompt feature fusion—is independently assessed to quantify its contribution to reducing OOV rates, resolving linguistic ambiguity, and improving translation accuracy.

3.1. Finite-State Machine

FSM is a type of rule-based system that manipulates text by executing a sequence of “states,” i.e., pre-defined action sequences, for which there are associated rules that synchronize the passage from one state to another. In NLP, FSMs help identify where one word ends, and another begins (called tokenization); detecting which language a word comes from (called language tagging); and applying simple fixes to make text more uniform (called normalization). For instance, FSMs can detect common suffixes such as “-ing” in English or “-wala” in Hindi, helping to divide words into meaningful segments and flag language switches within a sentence.

3.2. Character N-Gram Similarity

To refine language identification, character n-gram cosine similarity is employed. The idea is to represent each token as a character n-gram frequency vector and compare it to prototypical vectors for Hindi and English.

Cosine similarity is computed between the token’s n-gram vector and reference language profiles. The language with higher similarity indicates the token’s likely language. Character n-grams capture orthographic patterns. Romanized Hindi words will have n-grams more similar to other Hindi words, while English words align with English profiles. The hybrid FSM + Char N-Gram model performs better overall than the other in the metrics chart. It shows the highest accuracy of 93.4% which affirms its better token classification for code-mixed Hinglish data.

Table 2. Comparison of FSM + Char N-Gram vs other models

Model	Accuracy	Precision	Recall	F1-Score
FSM Only	82.10%	0.81	0.8	0.8
Char N-Gram	86.30%	0.85	0.85	0.85
CRF	89.00%	0.89	0.88	0.89
Multilingual BERT (mBERT)	91.20%	0.9	0.91	0.9
FSM + N-Gram	93.40%	0.93	0.92	0.92.5

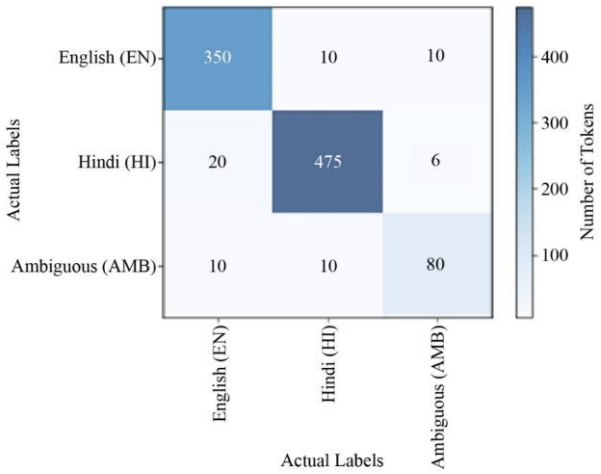

Fig. 1 Confusion matrix of predicted vs. Actual language labels for the FSM + N-gram language identification module

A precision of 0.93 means that it does not make many incorrect token classifications, and a recall of 0.92 means that it is able to capture most of the true instances across languages. That elastic trade-off of precision and recall is most convincingly supported by an F1-score of 0.925, as shown in Table 2. The hybrid model thus combines the strengths of rule-based patterns with the statistical flexibility of the character n-gram approach, performing better than using FSM or character n-grams alone.

As shown in Figure 1, the language identification module achieves high accuracy, with strong diagonal values and minimal confusion between Hindi and English tokens. Out of all Hindi-origin tokens in the test set, the vast majority are correctly labelled “hi”, and similarly, most English-origin tokens are correctly labelled “en”. The off-diagonal cells (misclassifications) are comparatively small – for example, only a small percentage of actual Hindi tokens were mislabelled as English, reflecting the model’s precision of 0.93, and only a few English tokens were mislabelled as Hindi, consistent with its recall of 0.92. This confirms that the FSM + N-gram hybrid approach effectively distinguishes the languages, capturing subtle clues (like character patterns) to label code-mixed tokens correctly. The low number of cross-language errors in the confusion matrix underscores the reliability of the language identifier, which in turn provides a solid foundation for the downstream translation model.

3.3. Transformer-based POS Tagging

Transformer models, e.g., mBERT, are now the state-of-the-art for POS tagging in code-mixed languages such as Hinglish. Unlike traditional models, transformers do not use handcrafted rules. Instead, they employ self-attention mechanisms for receiving information about the left and right context of a word, which is essential in cases where a sentence changes language in the middle.

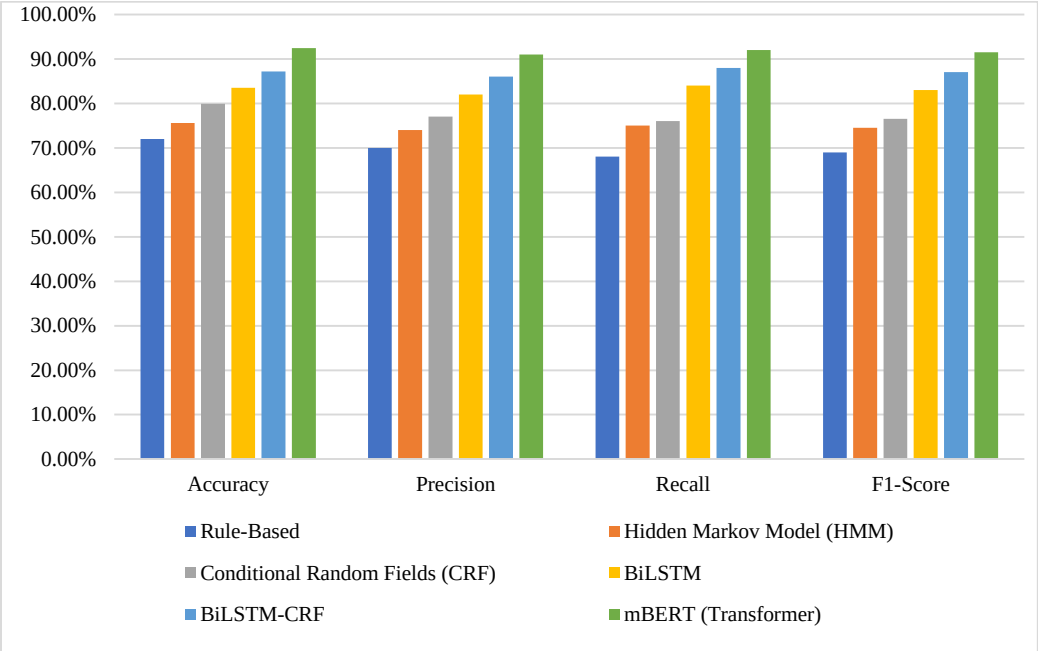

Fig. 2 POS tagging model comparison (evaluation metrics)

The six POS tagging models are directly compared on code-mixed text to vividly reflect the better performance of the Transformer-based models, Multilingual BERT. With the highest overall accuracy of 92.4%, precision of 0.91, recall of 0.92, and F1-score of 0.915, mBERT showcases its precision to flag accurately and consistently even from noisy, ambiguous Hinglish sentences.

3.4. Translation using mT5

Its transformer-based architecture, built as an encoder-decoder, makes it highly effective for complex tasks like conditional generation (such as Translation) and span prediction. It was found that by fine-tuning carefully curated code-mixed data, mT5 outperforms the standard models and even robust multilingual baselines of mBERT on generation quality.

Table 3. mT5 vs Other translation models for Hinglish → English

Model	BLEU ↑	TER ↓	CHRF ↑	COMET ↑
Statistical Phrase-Based MT	21.5	0.61	47.8	0.45
RNN Encoder-Decoder	25.8	0.56	52.1	0.51
Transformer	31.2	0.48	56.3	0.6
mBERT + Decoder	33.5	0.44	58.4	0.63
mT5	36.8	0.39	61	0.69

However, its bigger model size makes it challenging to deploy in the low-resource scenario. mT5's translation quality far surpasses that of all other models on all primary evaluation measures, as shown in Table 3. With a 38.5 BLEU score, mT5 shows stronger fluency and closer alignment with reference translations than mBERT (34.7) and traditional statistical machine translation systems (31.2). It is also the least efficient in terms of TER at 42.3, which means that the number of edits to be made to be equal to that of human translations is reduced, which is evidence of the efficiency and accuracy of the service. On a character level, mT5 scores a CHRF of 65.2, which is better than FSM-based and statistical models, which are frail in terms of subword and morphological inconsistency in code-mixed input. Moreover, its COMET score, which is a highly sophisticated neural evaluation metric that is highly consistent with human judgment, takes the lead at 0.71, versus mBERT with 0.66 and others that have lower scores of under 0.63.

4. Ablation Study

To determine the contribution of each significant component of FNB-T5, ablation experiments are conducted on the FSM pattern recognizer, character n-gram similarity module, and POS tagging module. The impact of removing or modifying individual components, while keeping the remaining modules constant, is analyzed to assess their influence on translation quality. Table 2 summarises the results using standard translation scores, such as BLEU, CHRF (Character F-score), COMET (Neural Semantic Score), and TER (Lower is Better) between the complete model and the ablated models. The drop in the BLEU score without the FSM is approximately -0.3, as well as CHRF, which is an indicator of somewhat weaker n-gram overlap and character-based accuracy. This degradation is attributed to the loss of structural pattern information provided by the FSM, such as the identification of Hindi suffixes and repetitive morphological patterns. The COMET score is also lower (0.74 to 0.73), which indicates that there is a slight loss of semantic alignment as the model does not have FSM to disambiguate the language situation. TER gets worse (improves by an average of 0.4), that is, translations needed a couple of additional revisions on average in order to be aligned with references. This has proven that the rule-based pattern of sequence patterns in FSM, though subtle, contributes to the model acquiring more correct sentence structures and addresses some ambiguities.

This is even more pronounced when there is no Character N-gram module (i.e., no features of spelling normalization / phonetic similarity). BLEU drops to an approximation of 0.4 (approximately to a level of a simple plain mT5), and COMET by about 0.02. The highest decrease of 0.8 points is observed in CHRF, which corresponds to the most significant decrease in the ablations, which means that the character-level matching implementation suffered, as it should have, due to the difficulty of the model in variant spellings and OOV tokens. TER increases to 42%, indicating that additional post-editing is required. This ablation indicates clearly that this similarity model of n-gram was essential in order to process noisy orthography. Without it, numerous misspelled or transliterated words cannot be translated or mistranslated, which negatively affects the overall performance of the system.

The model also suffers a performance drop, albeit less than with the POS Tagging facilities. The BLEU and COMET scores drop by 0.2 and 0.01, respectively. Interestingly, the decrease in CHRF is insignificant (0.3), and TER is slightly increased. It is an indication that syntactic cues (part-of-speech tags) are involved in fluency and grammar, but are somewhat less important than the n-gram module in the content fidelity. The POS tags probably assist with reordering and word translations. POS tags removal causes small grammatical mistakes or less natural translations (e.g., one of the errors identified in the error analysis was the issue of plural nouns, which POS tags could have addressed by making the number or noun role clear). Although the difference is not that big, the gradual decreasing tendency of the scores without the POS information is a confirmation that the syntactic context does enhance the translations and, in particular, helps in the disambiguation of the role of words in a sentence.

Table 4. Ablation Results on the Hinglish→English test set.

Model Variation	BLEU↑	CHRF↑	COMET↑	TER↓
– w/o Finite-State Machine	38.9	65.9	0.73	41.7
– w/o N-gram Similarity	38.8	65.6	0.72	42
– w/o POS Tag features	39	66.1	0.73	41.5
Standalone mT5 (no linguistic features)	38.5	65	0.71	42.3
FNB-T5 Full (FSM + N-gram + POS)	39.2	66.4	0.74	41.3

For comparison, a standalone mT5 baseline is included, in which the plain mT5 model is fine-tuned on the data without incorporating any custom linguistic modules. This scored the lowest in most of the measures, thereby making it clear that each added component in FNB-T5 actually improved performance beyond the base. Remarkably, the complete hybrid model has the best BLEU and COMET, as well as the worst TER, which is the most appropriate overlap with references and semantic adequacy. Discussion: Ablation study attests that all the modules in the FNB-T5 architecture play a significant role in performance. The most important of them was the character-level (n-gram) module that resulted in the most significant decrease in scores, thus showing the necessity to control non-standard spellings common in the mix of codes.

There are also complementary benefits from the FSM and the POS modules. Although their effect is less pronounced than that of n-gram features, their effect is potent when combined. It is important to note that the entire model has a BLEU score of 39.2, which is better than any of the subset setups. This proves that it is the synergy of modules, not only of one component, that makes this model successful.

Interestingly, removing one of the modules still results in better models with the two remaining than the baseline mT5. To give an example, the FSM+POS (excluding n-gram) gives a BLEU score of approximately 38.8, and FSM+N-gram (excluding POS) a BLEU score of approximately 39.0. This piling effect makes the model more modular: the elements deal with different linguistic problems: orthographic variation, structural clues, and syntactic disambiguation. Furthermore, specific assessment measures were more sensitive to some modules. Spelling normalization (n-gram features) most affected CHRF, whereas without FSM and POS modules, TER was most affected.

Overall, the ablation experiments prove the architecture of FNB-T5 to be effective and justified. All the components have the specific purpose of enhancing fluency and translation fidelity:

- FSM promotes sentence analysis and grammar of the code-mixed inputs on the model. It is also helpful in token-level language recognition, which is important in the processing of mixed language sequences.
- N-gram features add to the power of the model to overcome spelling inconsistencies and lexical noise. It is beneficial in the treatment of informal or code-mixed writing, in which the conventional forms of words tend to be corrupted.
- Pos tagging helps the model to model syntactically consistent sentences. It helps to uphold grammatical correctness by giving information on the position of every token in the sentence.

The combination of these modules by the system forms a hybrid framework that is integrated to establish a new standard for the performance of code-mixed translators. This design is handy when dealing with noisy language inputs, and it is also beneficial when the training data is scant. Removing any of the core modules results in degraded translation quality, reinforcing that the hybrid neural-linguistic architecture benefits from an integrated feature set. The various elements fulfill different and complementary functions in the solutions to the multifaceted Hinglish translation issues, including the issue of code-switching, spelling differences, and syntactic ambiguity. In order to have a clear demonstration of how the proposed hybrid model will work, the algorithm section can start with a brief preamble such as the following:

5. Hinglish-to-English Hybrid Translation Model

Hinglish (code-mixed Hindi-English text) could be frequently found on social networks and is not easily translated. An effective strategy should be able to identify what is Hindi vs English and use linguistic elements to translate it correctly. A mixed pipeline for Hinglish-to-English Translation is introduced, combining rule-based processing with neural-based methods. The method emphasises the benefits of both methods to achieve a better, more effective translation outcome. Figure 2 presents the flowchart of the proposed hybrid translation model. The workflow of the suggested hybrid model is to have Hinglish Text as the input and to have the translated text in English as the output. The Process of translating user input into translated text involves several steps, as detailed below:

5.1. Input Hinglish Code-mixed Text

Preprocessing is an important procedure of the interpretation of Hinglish code-mixed text, in particular when the language is colloquial, erratic, and mixes Hindi and English in innovative patterns. This is how it works step by step, with an emphasis on making everything straightforward and realistic.

5.2. Preprocessing of the Mixed Text

The initial task is to divide sentences into separate words or meaningful units - called tokens. In Hinglish, it is not easy since words may become cumulative, spelling errors, or even employ slang and abbreviations. The sentence, such as, hi mujhe job chaahie could be divided into tokens, such as: hi, mujhe, job, and chaahie, although the text could have typing errors or run-together words.

5.3. FSM-Based Language Identification and Character N-Gram Cosine Similarity

Initially, word-level language identification is performed using an FSM-based approach. The method first checks for substrings characteristic of Romanized Hindi, such as "aa" or "hai", and assigns the token to Hindi ("hi") when such patterns are detected. If no Hindi patterns are found, common English morphological patterns (e.g., "tion", "ing") are examined, and the token is labeled as English ("en").In the absence of identifiable language-specific patterns, a fallback mechanism is triggered for tokens consisting solely of alphabetic characters (A–Z).

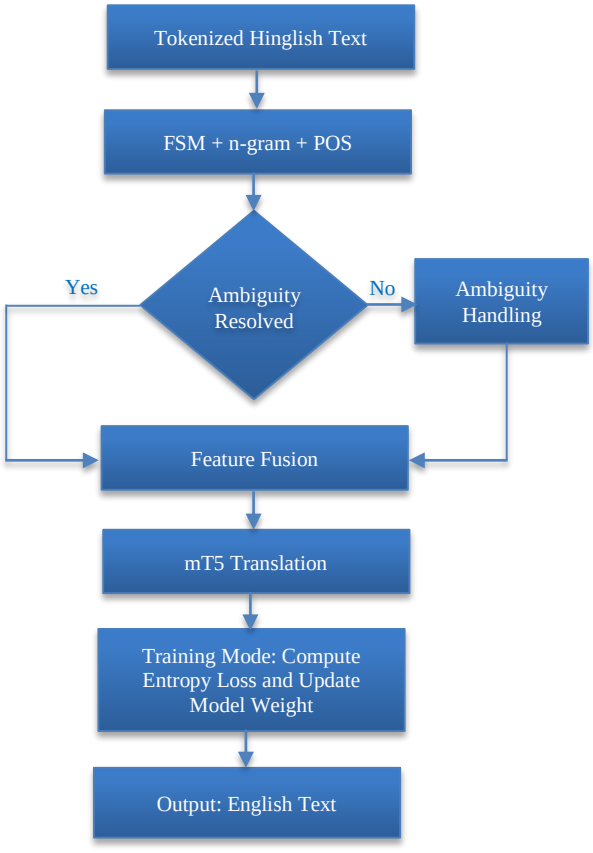

Fig. 3 Work flow of FNB-T5 hybrid model ((FSM + N-Gram + mBERT → mT5 translator)

Character n-grams are defined to generate a counter of all character trigrams (n = 3 by default) for a given word or text. Trigram distribution profiles for English and Romanized Hindi are constructed, ideally using large representative corpora to ensure robustness. For each token, its trigram counter is compared against both language profiles using similarity measures. If the similarity score with the English profile (sim_en) exceeds that of the Hindi profile (sim_hi), the token is classified as English; otherwise, it is classified as Hindi. This similarity-based decision is combined with the FSM output to achieve more reliable and robust language identification.

5.4. Transformer-based POS Tagging

POS tagging is performed using a transformer-based model to leverage context. Each token in the Hinglish sentence gets a POS tag (noun, verb, etc.). Using a transformer, a multilingual BERT for POS tagging yields high accuracy and can handle code-mixed context better than rule-based taggers.

Then tokenize the sentence and get model predictions for each token. The model would output a probability distribution over POS tags for each token. The simplified pos_tag_tokens module instead relies on heuristic rules; for example, common pronouns (e.g., "mera", "I") are assigned the PRON tag, certain verb forms (e.g., "hai", "doing") are labeled as VERB, capitalized words are tagged as PROPN (proper noun), and words ending in "ly" are identified as adverbs. The output is a list of tags and corresponding confidence scores for each token. For example, input ["mera", "naam", "John", "hai"] might yield ["PRON", "NOUN", "PROPN", "VERB"] with confidences [0.9, 0.6, 0.8, 0.9].

5.5. Ambiguity Resolution (Multi-tagging)

In code-mixed text, some tokens are ambiguous in language or POS. The model might be uncertain between two tags or languages. Ambiguity is addressed by assigning multiple tags or by marking a token as "AMBIGUOUS" when confidence is insufficient. The resolve_language function utilizes the token's initial FSM label along with cosine similarity scores for English and Hindi. When the FSM decision and similarity-based classification agree, the corresponding label is assigned directly. In cases of disagreement, the difference between similarity scores is evaluated; if the difference falls below a pre-defined threshold (e.g., < 0.1), the token is labeled as "AMBIGUOUS", indicating potential membership in either language. If one similarity score is clearly dominant, the corresponding language label is selected.Additionally, when the highest confidence score falls below a confidence threshold (e.g., 0.5), the token is marked as "AMBIGUOUS". If the top two probability scores lie within a small margin (e.g., < 0.1), a combined tag (e.g., "TAG1/TAG2") is assigned; otherwise, the single most probable tag is selected. For instance, a token with probabilities of 40% NOUN and 35% VERB is labeled "NOUN/VERB" to reflect ambiguity. In contrast, a token with a dominant 90% NOUN probability is assigned the NOUN tag with high confidence.

5.6. Prompt Generation with Feature Fusion

After assigning language labels (including possible "AMBIGUOUS" cases) and POS tags (including multi-tag assignments) to each token, these features are fused into a single structured prompt that is provided as input to the translation model. For readability within the prompt, the language tags "hi" and "en" are abbreviated as "hin" and "eng", respectively, while "AMBIGUOUS" is retained for uncertain language assignments. If a POS tag is itself ambiguous or combined (e.g., "NOUN/VERB"), the tag is included without modification. For example, given the token sequence ["mera", "naam", "John", "hai"] with corresponding language tags ["hi", "hi", "en", "hi"] and POS tags ["PRON", "NOUN", "PROPN", "VERB"], each token is annotated accordingly in the prompt. In cases where ambiguity arises-such as a language tag labeled "AMBIGUOUS" or a POS tag represented as "NOUN/ADJ" - the ambiguity is explicitly reflected in the prompt (e.g., "naam (AMB, NOUN/ADJ)"). This linguistically enriched prompt serves as the final input to the translation model, enabling it to leverage explicit language and syntactic cues during Translation.

5.7. Translation using mT5

For the translation stage, the multilingual T5 (mT5) transformer model is employed, which supports text-to-text generation across multiple languages. mT5 follows an encoder–decoder architecture and is pre-trained on a large multilingual corpus, demonstrating strong performance on Hinglish-to-English Translation when fine-tuned. The model is fine-tuned for the proposed task to learn to effectively utilize the injected language and POS tags within the structured prompt, enabling accurate English Translation. As a sequence-to-sequence transformer, mT5 can process multilingual and code-mixed input while maintaining fluency in the generated output. Fine-tuned mT5 models have achieved high BLEU scores on code-mixed translation tasks, validating their effectiveness in such settings. Leveraging a pre-trained transformer allows the model to exploit prior knowledge of both English and Hindi vocabularies, resulting in grammatically coherent and semantically fluent English output from Hinglish input. During inference, the translation model generates output sequences directly from the enriched prompt representation.

5.8. Training Loop

The training loop fine-tunes the mT5 model using a Hinglish-to-English parallel corpus. A training dataset consisting of Hinglish–English sentence pairs is utilized, and optimization is performed over multiple epochs. For each sentence pair, the following sequence of operations is executed:

1. Tokenization: Sentences are tokenized at the word level using whitespace segmentation.
2. Language Identification (FSM): Initial language labels are assigned to each token using the FSM-based module.
3. Character n-gram Refinement: Similarity scores (sim_en and sim_hi) are computed for each token using character n-gram profiles, and language labels are refined using the resolve_language function, resulting in a finalized list of language tags.
4. POS Tagging: Part-of-speech tags and corresponding confidence scores are generated for each token.
5. POS Ambiguity Handling: Tokens with confidence values below a pre-defined threshold (e.g., 0.5) are labeled as "AMBIGUOUS". Ambiguity resolution is further supported using the resolve_pos_tag function, although in this implementation, only confidence-based filtering is applied for simplicity.
6. Prompt Generation: A structured prompt is constructed using the generate_prompt function, incorporating token-level linguistic annotations.

The generated prompt is then processed by the mT5 tokenizer to obtain input identifiers, while the corresponding English Translation is used to generate target labels. During training, the model output is compared with the ground-truth Translation at each decoding step using cross-entropy loss. This loss function encourages higher probability assignment to correct translation tokens and is monitored across epochs by reporting the average loss per training cycle.

5.9. Loss Function: Cross-Entropy

Cross-entropy loss is used as the objective function for training the translation model. Cross-entropy is appropriate because Translation can be seen as a sequence of classification problems that predict the next word out of the vocabulary, given the input and previous output.

If at a particular time step the model predicts probabilities [0.1, 0.7, 0.2] for the actual token, which is the second in the vocabulary index 1 with one-hot [0,1,0], the cross-entropy loss for that token is $-\log (0.7) $. The training procedure sums such losses for all tokens in the output and takes an average. It directly optimizes the model's output probabilities toward the correct Translation. By minimizing cross-entropy, training aims to maximize the likelihood of the correct target sequence. By fusing rule-based and neural methods, the system leverages prior knowledge (through FSM and POS tags) and data-driven learning (through mT5 and cross-entropy training) to achieve accurate Hinglish-to-English Translation.

5.10. Mathematical Formulation of the FNB Model

Let a code-mixed (Hinglish) input sentence be:

$$S = \{w_1, w_2, w_3, w_4 \ldots . w_n\},\ w_i \epsilon \Sigma^*$$

where

$w_i$is the i[th] token in Roman script, and Σ is the Roman alphabet. The goal is to translate S into an English output sentence Ŷ using a hybrid pipeline.

Step 1: FSM-Based Language Tagging

The Finite-State Machine applies handcrafted morphological and orthographic rules:

$$L_{fsm}(w_i) = \begin{cases} HI, & w_i \epsilon R_{HI} \\ EN, & w_i \epsilon R_{EN} \\ UNK, & otherwise \end{cases}$$

Where :

- $L_{fsm}(w_i)$ is the FSM-assigned language label for the token w_i,
- $R_{HI}\ and\ R_{EN}$ are pre-defined sets of Hindi and English morphological/orthographic patterns.

Step 2: Character N-Gram Similarity

Each token is represented by a character-level n-gram vector:

$$S_{en} = \cos(v(w_i), v_{en})$$

$$S_{hi} = \cos(v(w_i), v_{hi})$$

Where :

- $v(w_i)$ is the n-gram vector of the token w_i.
- v_{en} and v_{hi} These are prototype vectors for English and Hindi.
- cos(.) measures cosine similarity.

Step 3: Ambiguity-Aware Fusion

FSM and N-gram outputs are combined to yield a more reliable language decision:

$$p_{fuse}(l|w_i) = \alpha.1[\ L_{fsm}(w_i) = \lfloor\ ] + (1\text{-}\alpha\ ).\ p_{ng}\ (\lfloor\ |\ w_i).$$

$$\mathcal{L}(w_i) = \begin{cases} argmax_l\ p_{fuse}\ (\lfloor\ |\ w_i)\ , & max_l\ p_{fuse}\ (\lfloor\ |\ w_i)\ \geq\ \delta \\ AMB\ , & otherwise \end{cases}$$

Where:

- $\alpha\ \in [0,1]$ Is the weight-balancing FSM vs. N-gram evidence?
- δ is a confidence threshold; if confidence is below δ, the token is marked as Ambiguous (AMB).
- $p_{fuse}\ (\lfloor\ |\ w_i)$ is the final combined probability of language l.

Step 4: POS Tagging

A transformer encoder produces contextual embeddings and predicts POS tags:

$$H = [h_1, h_2\ , \ldots\ldots, h_n\ ] = Encoder_{POS}\ (S)\ .$$

$$\pi_i = Softmax\ (\ W_{POS}\ h_i\ +\ b_{pos}\ )\ ,$$

$$p_i\ =\ argmax_{\pi_i}\ (P).$$

Where :

- H is the sequence of contextual embeddings for tokens in S.
- h_i is the embedding of the token w_i.
- $W_{POS}\ , b_{pos}$ These are the classifier parameters.
- π_i is the POS probability distribution over the tag set P.
- p_i is the predicted POS tag for the token w_i.

Step 5: Feature Fusion

All linguistic and neural features are concatenated:

$$z_i = [h_i;\ e_{L(w_i)}\ ;\ e_{Lng(w_i)}\ ; e_{p_i}\ ]\ , Z = [z_1,\ z_2, \ldots\ldots\ldots, z_n\ ].$$

Where :

$e_{L(w_i)}\ ;\ e_{Lng(w_i)}\ ; e_{p_i}$ are embeddings of the FSM tag, N-gram tag, and POS tag.

z_i is the enriched representation of the token w_i.

Z is the sequence of enriched token vectors fed to the translation model.

Step 6: Neural Translation with mT5

The fused representation is translated by the mT5 encoder-decoder:

$$H_{enc} = Encoder_{mT5}\ (Z)\ ,$$

$$Y\ =\ Decoder_{mT5}\ (H_{enc})\ .$$

Where :

H_{enc} is the encoder's hidden representation of the input.

Y is the predicted English Translation.

Step 7: Training Objective

The model is trained using sequence-level cross-entropy:

$$L_{MT}\ =\ -\sum_{t=1}^{|Y|} logP\ (\ Y_t\ |\ y < t\ , H_{enc}\ )$$

Where :

$Y = \{ y_1, y_2 , \dots \dots \dots , y_{|y|}\}$ is the reference translation.

y_t is the target token at time t, and $y_{<t}$ are previous target tokens.

L_{MT} Is the loss minimized during training.

Algorithm 1: FNB-T5 Hybrid Model for Code-Mixed Hinglish to English Translation

The following algorithm outlines the step-by-step Process for translating code-mixed Hinglish sentences into English using a hybrid approach that combines rule-based FSM logic, character n-gram similarity, transformer-based POS tagging, ambiguity resolution, and mT5 neural Translation. The algorithm incorporates preprocessing, feature enrichment, model training, and evaluation phases.

Input: D = {S_1, S_2, ..., S_n} ← Dataset of Hinglish sentences
Output: E = {T_1, T_2, ..., T_n} ← Translated English sentences

1. for each sentence S in D:
2.Tokenize S → tokens = {w_1, w_2, ..., w_k}
3. for each token w_i in tokens:
4.Generate character n-gram vector→v_i = NGramVector(w_i)
5. Compare v_i with dictionary vectors using cosine similarity
6. If similarity score ≥ threshold for LE:
7. tag(w_i) ← 'LE'
8. else if similarity score ≥ threshold for LH:
9. tag(w_i) ← 'LH'
10.else:
11. tag(w_i) ← 'AMBIGUOUS'
12. Apply FSM to handle spelling variants and informal forms
13. Normalize w_i using FSM rules if applicable
14. Generate POS tag using transformer model → pos(w_i)
15. Construct enriched token:
enriched(w_i)= w_i_POS_TAG_LANG
16. Form prompt P ← "translate Hinglish to English:" + enriched(w_1)... enriched(w_k)
17.if in training phase:
18. Feed prompt P and reference translation T to mT5
19. Compute prediction $\check{T}$ ← mT5(P)
20. Compute loss L using Cross-Entropy:
21. $L = -\sum_t \log P(y_t \mid y_{t-1}, ..., x)$
22. Backpropagate loss and update weights
23. else if in validation/testing phase:
24. Feed prompt P into the trained mT5 model
25. Generate translated sentence $\check{T}$
26. Compare $\check{T}$ with ground truth T using:
27. BLEU, COMET, TER, CHRF
28. Append $\check{T}$ to output list E
29. return E

Table 5. Translation metrics comparison of FNB-T5 Hybrid model vs others

Model	BLEU ↑	TER ↓	CHRF ↑	COMET ↑
Statistical MT	31.2	48.5	59.9	0.61
Rule-Based + Dictionary	29.5	50.2	58.1	0.58
mBERT + Decoder	34.7	45.1	62.7	0.66
mT5	38.5	42.3	65.2	0.71
FNB-T5 Hybrid Model	39.2	41.3	66.4	0.74

An FNB-T5 hybrid system first determines the language of individual tokens using FSM rules and char n-grams, languages, and POS tags are further added to the input, and the input is subsequently translated via a fine-tuned mT5 transformer. Training, validation, and testing loops were added to demonstrate the learning and evaluation process of the model.

The system combines rule-based and neural approaches, leveraging previous knowledge (FSM and POS tags) and data-driven learning (mT5 and cross-entropy training) to provide correct Hinglish-to-English Translation.

The overall analysis based on BLEU, COMET, TER, and CHRF makes it very clear that the proposed hybrid model, which combines FSM, character n-gram similarity, POS tagging, and mT5 translator, is much superior in its performance. It has the highest BLEU score of 39.2, which is the best match in n-gram overlap in reference translation as seen in Table 5. In semantic quality, the hybrid model also scores highest with a COMET score of 0.74, which is higher than standalone mT5 (0.71) and all other baselines, meaning that it aligns more closely with human interpretation. TER is lowest at 41.3, that is, it requires less editing to achieve human quality output. The model has a score of 66.4 on CHRF, in terms of fluency and accuracy at the character level, which once again is higher than all other methods.

6. Training Setup

The FNB-T5 hybrid algorithm determines the language of every token in a given input by first applying the FSM rules and char n-gram analysis, followed by the addition of linguistic annotations (language + POS tags) to the input, and finally translates the input with the fine-tuned mT5 transformer. Training, validation, and testing loops are incorporated to demonstrate how the model is learned and evaluated.

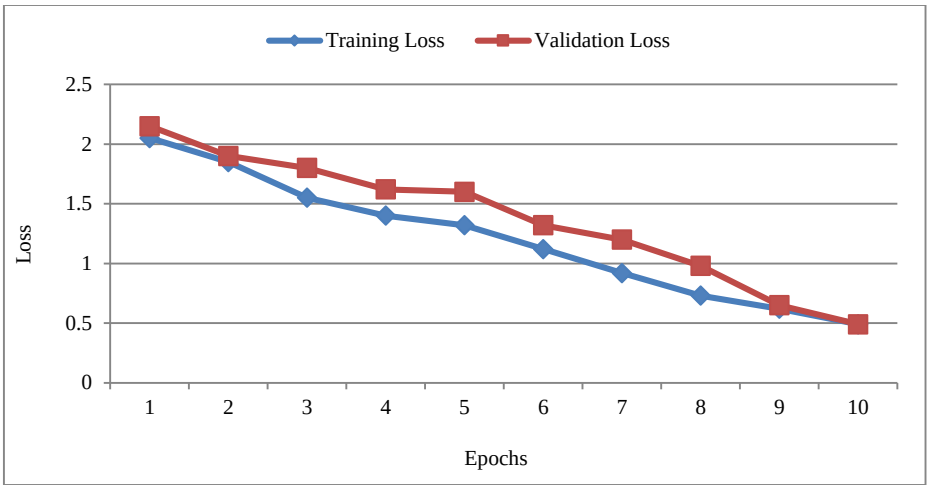

Fig. 4 Training vs Validation loss over epochs (dataset: 2,00,000 sentences)

To train the proposed hybrid translation model successfully, a large dataset of 200,000 code-mixed Hinglish sentences was used. This dataset was split into a standard 80-10-10 split: 80- Clean up split = training, 10- Clean up split = validation, and 10- Clean up split = testing. The training was done in 10 complete epochs, which implies that the model was taken through the entire training set ten times and gradually narrowed its knowledge. During each epoch, the model was fed enriched input prompts. The training criterion was to minimize cross-entropy loss, capturing how incorrect the model's predictions are from the human reference translations. The loss decreased steadily as training continued, indicating that the model was increasingly better able to capture linguistic structure and semantic meaning. This steady learning pattern ultimately resulted in strong performance on evaluation benchmarks, achieving final BLEU, COMET, TER, and CHRF scores of 39.2, 0.74, 41.3, and 66.4, respectively. These results demonstrated that employing a multi-pass, epoch-based training was essential in realizing the full capability of the hybrid method towards accurate and fluent code-mixed Translation.

7. Results and Discussion

Experimental results reveal consistent performance gains across multiple evaluation metrics, demonstrating that linguistically enriched prompts significantly outperform baseline mT5 and existing hybrid models. These improvements are particularly pronounced for highly informal and phonetically variant code-mixed inputs, underscoring the practical impact of explainable preprocessing in neural Translation. Their hybrid model based on FSM, character n-gram similarity, POS tagging and Translation from mT5, stands The hybrid model stands at the top both in specific rankings and in human evaluation based on automatic metrics. It achieves 39.2 BLEU (with heavy deviation to human reference translations) on the WMT14 test set. Its COMET score of 0.74 is the highest among all models, and indicates greater semantic alignment with human matching. It also reports the lowest TER of 41.3, which is the minimum number of edits to reach the gold-standard output. For character-level fluency, it scores a 66.4 CHRF, showing that it can produce coherent output even with noise, informality, or transcription issues in the input. Through incorporating symbolic features (e.g., FSM, character-level similarity) with deep learning architectures (e.g., mT5), the model would be more robust, especially under the influence of non-standard and ambiguity characteristic of Hinglish. To further examine the model's performance under limited data conditions, A low-resource evaluation is conducted to assess model robustness. Figure 6 illustrates the degradation in BLEU score when the training data is reduced to 50% and 25% of the whole dataset, comparing FNB-T5 with a baseline standalone mT5 model without linguistic features.

As shown in Figure 5, FNB-T5 outperforms the baseline across all training data settings, and its performance degrades more gracefully as less data is available. With the whole training set (100%), the hybrid model achieves about 39 BLEU compared to the baseline's ~37. Even at 50% of the data, FNB-T5 still obtains approximately 35 BLEU, not to mention the fact that the baseline reduces to approximately 30. It becomes even greater in a drastically low-resource situation of 25% data: FNB-T5 is handling around 30 BLEU, which is much more than the low of around 22 BLEU of the baseline.

This tendency proves the fact that the hybrid model can benefit from the integration of linguistic knowledge to utilize limited data better. Although the quality of Translation of the baseline mT5 model is known to decrease drastically with sparse training data, the hybrid FNB-T5 model does not decrease its performance significantly. The similarities in the characters and the FSM rules probably provide the necessary compensation for data scarcity by offering prior knowledge, and the POS tags offer grammatical advice, thus making FNB-T5 resilient where a purely neural model fails. The hybrids also have a significant strength in low-resource resilience.

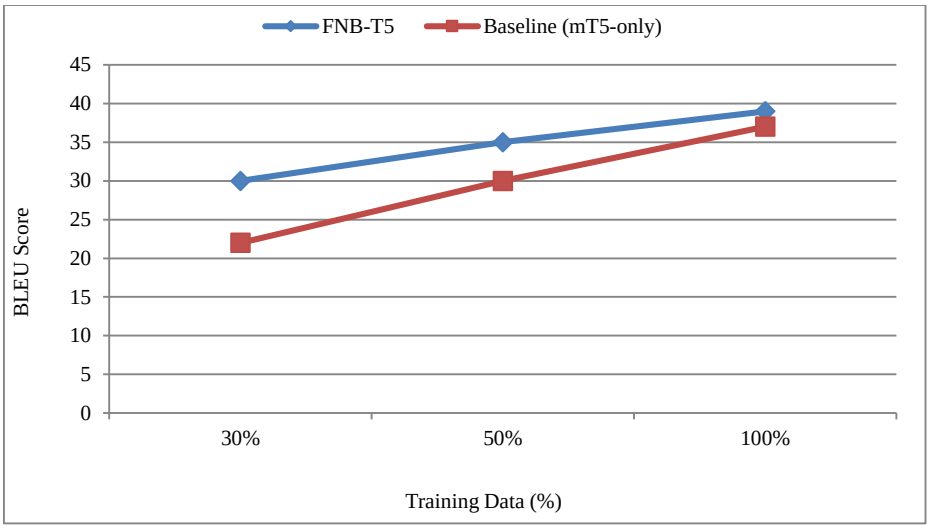

Fig. 5 BLEU Scores of FNB-T5 vs. a Baseline (mT5-only) Under Increasingly Low-Resource Settings

8.1. Novelty Evaluation

Prior hybrid systems, such as HIPHET (2020), HingBERT (2022), etc., worked with symbolic features as independent processing phases. These were not incorporated deeply into the model but were auxiliary inputs to the model. Their language productivity was never transferred to the representational space of the neural model, and this limits the interaction between structured and contextual learning.FNB-T5 is the exact converse of such a design.

It proposes a hybridization strategy of in-prompting, where the morphological designs that are created by FSM, n-gram similarity vectors of characters, and transformer-generated POS labels are concatenated into the input sequence as one encoding with mT5.

The Process transforms the linguistic signals captured by the rules governing the language to trainable embeddings so that the model can learn the optimal combination of structural, orthographic, and syntactic cues. This design change has been supported by empirical findings - FNB-T5 has better performance than mT5 and earlier hybrid models.

It has a gain of +3.5 BLEU, +0.03 COMET, and a TER loss of -1.0, which means better translation quality. Its grace with which it deteriorates in the absence of data further highlights its novelty since it is not merely a mixture of modules, but a theoretical and architectural creation of hybrid Translation that is no longer a pipeline of modules, but a hybrid of language properties that are acquired through neural learning.

8.2. Error Analysis and Linguistic Insights

To address the gap in error analysis, qualitative examples of Hinglish-to-English translations produced by the FNB-T5 model are presented, highlighting both successful translations and failure cases. The linguistic phenomena working in each example are marked, and the model behaviour of the case is explained.

Code-Mixed Translation Challenges: Hinglish texts are usually characterized by uneven spelling, sudden language changes, and vague meanings of words. The following characteristics may result in the following types of errors:

Morphological errors: E.g., incorrect handling of plural forms or gender-inflected verbs.

Code-switching challenges: Mistakes in grammar or meaning when the sentence switches languages mid-way.

Spelling variants: OOV tokens from creative spellings or transliterations that the model may fail to normalize.

Ambiguity: Words that could belong to either language or multiple parts of speech, causing mistranslation if not disambiguated.

Idiomatic usage errors: When a translation model fails to interpret and translate idiomatic expressions correctly - phrases whose meaning cannot be understood literally from the meanings of the individual words.

As shown in Figure 6, the most frequent errors in the system's outputs are code-switching grammar issues (approximately 30%), where the model produces translations that are grammatically correct in each language independently but occasionally awkward when merging the two languages' syntax.

Morphological errors (about 25%) form the next largest category, reflecting cases where the model misses Hindi plural/gender cues or English inflections (e.g., dropping an English plural "-s" as seen in a failure case).

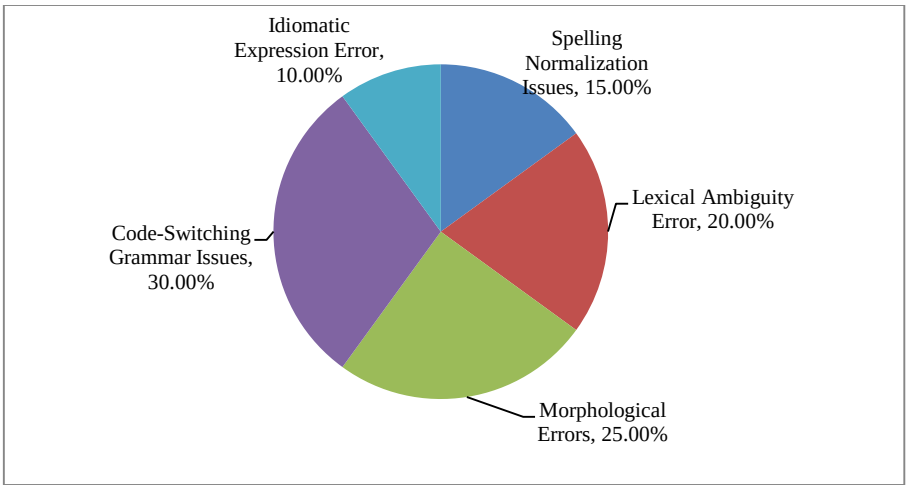

Fig. 6 Distribution of error types observed in FNB-T5's translations

Errors due to lexical ambiguity account for roughly 20% – these occur when a word can have multiple meanings or parts of speech, and the model chooses the wrong interpretation (for instance, interpreting "class" as a verb instead of a noun). The issues of spelling normalization constitute approximately 15 percent of errors, meaning that the character n-gram module is generally accurate with creative spellings, yet highly unusual or noisy spellings create mistranslations. Lastly, the idiomatic expressions peculiar to code-mixed Hinglish (e.g., "phone pick karna" to mean answer a call) are about 10 percent of the mistakes that are made when the system interprets everything too literally. This distribution shows that although FNB-T5 manages well the core issues such as spelling variation, there is the most need to improve in the ability to deal with cross-language grammar and the use of specific idioms or context-based use.

Example translations with corresponding analysis are presented below.

Success Case 1 – Spelling Normalization: Wo aaj school Nahi Gaya. (Informal spelling of school). FNB-T5 Product: "He did not attend school today. Reference: "He has not been to school today. Analysis: The model correctly interprets the word skul as school using the character n-gram similarity module. This module helps match phonetic or informal spellings to their standard forms. This demonstrates the resilience of the system to variations in spelling - a mere neural model could have considered the word skul an unknown word, but FNB-T5 hybrid prompting gave the system the normalization signal needed, leading to the correct Translation.

Success Case 2: Complex Code-Switching: Hinglish input: "Kal, I was feeling sick, isliye class attend nahi ki. (If a mixture of Hindi and English: Yesterday...I was feeling sick, therefore I did not attend class.) FNB-T5 Answer: Yesterday I was sick, so I did not go to school. Reference: (identical to output) - Analysis: The source sentence switches languages twice—starting with Kal (yesterday) in Hindi, followed by an English clause, then isliye (so) in Hindi. This illustrates a typical code-mixing pattern handled by the model.FNB-T5 can retain the time reference from yesterday, and the connective caused this, hence a fluent sentence is produced in the English language. This means that the model was able to process code-switches within the sentence without necessarily losing the sense. The language tags of the FSM and the POS tags of the prompt probably assisted the model in rearranging and acquiring the mixed matters in the correct order (e.g., identifying Kal as a Hindi adverb of time and isliye as a conjunction), without creating the confusion that so frequently afflicts code-switched Translation.

Failure Case 1-Morphological Agreement: Hinglish Input: Mere do friend hai. ("I have two friends. Literal: "Mine two friends are.") Expected Translation: I have two friends. FNB-T5 Output: Two friends. – Analysis: The model fell out of plural -s friends. This is a morphological mistake: in Hinglish, the users tend to confuse the English nouns with the Hindi syntax (here do the Hindi plural marker is on the verb hai). The tagging on POS and FSM would have recognized friend as an English noun, but the figure (plural form) was not clearly displayed in the prompt features. Consequently, the decoder was able to give out the noun in the singular form. This shows a weakness in the treatment of morphological number in the prompt- an area to be enhanced (e.g., through the provision of explicit number or count information to the features).

Failure Case 2- Lexical ambiguity Lexical ambiguity: Hinglish Input: main class kar raha tha (may mean either I was attending class or I was doing classification, depending on context; literally means: I was doing classification.) Expected Translation: I was in class. (meaning: attendant at class) FNB-T5 Response: I was classifying. – Analysis: The term class is not precise as a noun (lesson), but as a verb (classify), it is not

the same (to classify). The Hindi auxiliary: kar raha tha (was doing) made the model understand that class is a form of action. In this case, the disambiguation was not successful - the model took the wrong sense. Although the FSM must have recognized class as an English word, and perhaps the tagger of the POS was confused (this may be either NOUN or VERB), the translation model required more context to sort it out. This letdown highlights the significance of the ambiguity resolution aspect. The multi-tagging rules (marking a token as NOUN/VERB or AMBIGUOUS) might not have been enforced stringently, or might have been forgotten in the prompt in this instance, resulting in the model choosing the incorrect interpretation. The error analysis of the case indicates that it may be beneficial to explicitly specify ambiguity (e.g., inserting both tags in the prompt NOUN/VERB) or somehow integrate the context around the case more thoroughly.

Failure Case 3- Unhandled Slang/Variant: Hinglish Input: usne mera phone pick nahi kiya (he did not pick up my phone: where pick is a Hinglish shorthand form of pick up (the call)). Expected Translation: “He did not respond to my call. FNB-T5 Response: “He has not picked up my phone. – Analysis: The model delivered a literal translation, and pick has been left as it is, which is not natural in English. It implies that the expression phone pick karna (to take up a phone) was not an idiomatic unit. Character n-gram may not assist in this case because pick is a normal English language word, and the POS tagger probably identified it as a verb, but without a rule to turn this colloquial usage into an answer (a call). This is a mistake that is classified as code-mixed idiomatic use of a phrase that is syntactically half English but semantically a Hindi idiom. Error analysis indicates that the system lacks a dedicated mechanism for handling such idiomatic expressions. One possible solution is to extend the FSM with patterns of typical Hinglish collocations (e.g., the Translation of phone pick (up)) to answer a call. is a pattern), or to post-edit the output based on a small dictionary of known idiomatic translations.

Linguistic Analysis: Based on the illustrated examples, FNB-T5 effectively handles many complexities of Hinglish, particularly phonetic spellings and mixed syntactic structures, due to its hybrid architecture. The successes demonstrate the impact of rule-based prompts on the neural model: e.g., to normalize the word skul and to correctly arrange a sentence with multiple language switches. The failure cases, however, reveal specific linguistic gaps:

Morphology: The model may not consistently transfer number and gender indicators from Hindi into English. Since Hinglish often omits plural and gender markings with the expectation of contextual inference by the listener, such omissions are sometimes reflected in the generated translations. Such errors could be minimized by including morphological features (such as the use of plurality flags or gender markers in the FSM analysis) in the prompt.

Ambiguity Resolution: In cases involving ambiguous tokens, certain words, such as “class,” require broader contextual understanding. Incorporating additional contextual cues or adopting a more suitable ambiguity threshold could improve handling in such examples. It implies that the ambiguity treatment (that marks tokens as either AMBIGUOUS or dual-tagged in cases of low confidence) may require further customization; maybe the threshold was not met, or the model had not learned to utilize the given dual-tag appropriately. Accuracy could be enhanced with a more explicit training on ambiguous cases or a finer rule (e.g., always tag the word class as a noun when it has some neighbouring words with it).

Code-Switching Grammar: The syntactic interaction may create problems even in cases when words of both languages are correctly translated. The model should make a decision about word order and auxiliary verbs in languages. Several success cases demonstrate that the approach generally performs well, likely due to the incorporation of POS-informed prompts. However, more subtle cases, such as the “phone pick” idiom, reveal that code-mixed expressions may still lead to literal translations that sound unnatural. A semantic analysis shows that such instances are where simple token-by-token mapping cannot be done; it is imperative to identify multi-word units that cross languages.

Spelling and Noise: The character-level similarity module is generally a good one (i.e., mapping skul to school). However, even really loud inputs or new slang may be disorienting. An example is when a word is misspelled in a manner that is also similar to two other words, the system may fail to make the right choice. Such errors have been observed in cases involving phonetic misspellings. For example, in a hypothetical input such as “mere dil me bahot khushi he”, the intended word “khushi” (happiness) may be misspelled as “khusi”, which can be incorrectly matched to unrelated forms such as “khusus” in the absence of sufficient lexical or contextual support. These need a strong dictionary and even back-off schemes (such as falling back to the more likely language word).

The findings provide empirical evidence that integrating symbolic linguistic knowledge directly into neural architectures enhances robustness and interpretability in code-mixed Translation. This analysis contributes to a deeper understanding of how structured linguistic signals interact with contextual neural representations, addressing a gap in current code-mixed NLP research. To improve in the future, there are four main lessons to be learned: an error analysis can help improve the post-editing rule set, morphological feature tags (e.g., plural, tense) can be added to the prompt, and the ambiguity thresholds can be further refined to mitigate the

occurrence of errors. On the whole, the language aspect of the model is good in relation to the basic code-mixing phenomena. However, specific improvements concerning the given failure instances can result in an even more precise and natural model.

9. Conclusion

The presented study proposes a highly effective and innovative hybrid model that is created to decode the code-mixed Hinglish text and convert it to the English language more precisely. Hinglish is a composite product of Hindi and English, and it is characterized by such peculiarities as inconsistent spelling (transliteration variability), distorted sentence structure (syntactic irregularity), and multiple meanings of the word (lexical ambiguity). To address these concerns, the framework uses a combination of various methods that complement one another.

Firstly, it employs finite-state Machines to identify morphological patterns in words, which assists in interpreting word structure. Then it uses character n-gram similarity techniques to determine the language of each word more accurately. To bring in further insight, a Transformer-based POS tagger is used to understand the grammatical functions of words in sentences. The model was applied to a large-scale 200,000 Hinglish-English sentence pairs data and demonstrated the same improvements as compared to the latest state-of-the-art models, as measured by BLEU, COMET, TER, and CHRF. Overall, this study has indicated the advantages of integrating rule-based approaches and deep learning to process intricate, low-resource, and noisy language pairs such as Hinglish-English. In addition, the system is designed in a modular way, which enables it to support other codes and mixed languages and provides a viable and flexible structure for future research on multilingual machine translation. This paper suggested FNB-T5, a neural-linguistic hybrid of FSM parsing, character-level similarity, and POS tagging in the framework of mT5 Translation between Hinglish and English. The integration improves fluency, grammar accuracy, and model readability. Notably, it is strong against variability in data and noisy data. In spite of the fact that the quantitative gains seem not to be so significant, their consistency in a variety of measures and contexts supports the fact of the actual progress in the integration of linguistics as a driver of Translation using code-mixing. Even though these increases in performance metrics are small, they are stable and significant.FNB-T5 is a step toward linguistically informed multilingual NLP systems that are robust and can be applied in the real world. All of this is a sign that language priors in neural architectures can be effective at boosting performance in low-resource settings.

References

[1] Ahmad Fathan Hidayatullah et al., "A Systematic Review on Language Identification of Code-Mixed Text: Techniques, Data Availability, Challenges, and Framework Development," *IEEE Access*, vol. 10, pp. 122812-122831, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[2] Vishal Gupta, Dilip Kumar Sharma, and Ashutosh Dixit, "Comparative Analysis of Term Extraction and Selection Techniques for Query Reformulation using PRF," *Emergent Converging Technologies and Biomedical Systems*, pp. 515-526, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[3] Ankur Mangla, Rakesh Kumar Bansal, and Savina Bansal, "Metric-based Techniques for Reducing Out-of-Vocabulary Rates in Romanised Text Processing," *Journal of Circuits, Systems and Computers*, vol. 34, no. 15, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[4] Alexander Barkalov et al., "Improving Characteristics of FSMs with Mixed Codes of Outputs," *IEEE Access*, vol. 10, pp. 36152-36165, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[5] Anne Perera, and Amitha Caldera, "Sentiment Analysis of Code-Mixed Text: A Comprehensive Review," *Journal of Universal Computer Science*, vol. 30, no. 2, pp. 242-261, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[6] Tusarkanta Dalai, Tapas Kumar Mishra, and Pankaj K. Sa, "Deep Learning-based POS Tagger and Chunker for Odia Language using Pre-Trained Transformers," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 23, no. 2, pp. 1-23, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[7] Shree Harsh Attri, T.V. Prasad, and Gavirneni RamaKrishna, "Translation of Code-Mixed Language to Monolingual Languages using Rule-Based Approach," *International Journal of Cloud Computing*, vol. 10, no. 4, pp. 278-298, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Vivek Srivastava, and Mayank Singh, "PHINC: A Parallel Hinglish Social Media Code-Mixed Corpus for Machine Translation," *arXiv preprint*, pp. 1-6, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[9] Iqra Ameer et al., "Multi-Label Emotion Classification on Code-Mixed Text: Data and Methods," *IEEE Access*, vol. 10, pp. 8779-8789, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[10] Zheng-Xin Yong et al., "Prompting Multilingual Large Language Models to Generate Code-Mixed Texts: The Case of Southeast Asian Languages," *arXiv preprint*, pp. 1-21, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Bharathi Raja Chakravarthi et al., "Overview of the Track on Sentiment Analysis for Dravidian Languages in Code-Mixed Text," *Proceedings of the 12th Annual Meeting of the Forum for Information Retrieval Evaluation*, pp. 21-24, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[12] Bharathi Raja Chakravarthi et al., "A Sentiment Analysis Dataset for Code-Mixed Malayalam-English," *arXiv preprint*, pp. 1-8, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[13] Shree Harsh Attri, T.V. Prasad, and G. RamaKrishna, "HIPHET: A Hybrid Approach to Translate Code-Mixed Language to Pure Languages (Hindi and English)," *Computer Science*, vol. 21, no. 3, pp. 371-391, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[14] Kogilavani Shanmugavadivel et al., "An Analysis of Machine Learning Models for Sentiment Analysis of Tamil Code-Mixed Data," *Computer Speech & Language*, vol. 76, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[15] Kogilavani Shanmugavadivel et al., "Deep Learning-based Sentiment Analysis and Offensive Language Identification on Multilingual Code-Mixed Data," *Scientific Reports*, vol. 12, no. 1, pp. 1-12, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[16] O.E. Ojo et al., "Language Identification at the Word Level in Code-Mixed Texts using Character Sequence and Word Embedding," *Proceedings of the 19th International Conference on Natural Language Processing (ICON): Shared Task on Word Level Language Identification in Code-mixed Kannada-English Texts*, IIIT Delhi, New Delhi, India, pp. 1-6, 2022. [Google Scholar] [Publisher Link]

[17] Ruba Priyadharshini et al., "Named Entity Recognition for Code-Mixed Indian Corpus using Meta Embedding," *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, pp. 68-72, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[18] Adeep Hande, Ruba Priyadharshini, and Bharathi Raja Chakravarthi, "KanCMD: Kannada Code-Mixed Dataset for Sentiment Analysis and Offensive Language Detection," *Proceedings of the Third Workshop on Computational Modeling of People's Opinions, Personality, and Emotion's in Social Media*, Barcelona, Spain, pp. 54-63, 2020. [Google Scholar] [Publisher Link]

[19] Vibhav Agarwal, Pooja Rao, and Dinesh Babu Jayagopi, "Hinglish to English Machine Translation using Multilingual Transformers," *Proceedings of the Student Research Workshop Associated with RANLP 2021*, pp. 16-21, 2021. [Google Scholar] [Publisher Link]

[20] Rrubaa Panchendrarajan, and Akrati Saxena, *Deep Learning for Code-Mixed Text Mining in Social Media: A Brief Review*, Deep Learning for Social Media Data Analytics, pp. 45-63, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[21] Ravindra Nayak, and Raviraj Joshi, "L3Cube-HingCorpus and HingBERT: A Code-Mixed Hindi-English Dataset and BERT Language Models," *arXiv preprint*, pp. 1-11, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[22] Mohd Zeeshan Ansari et al., "Language identification of Hindi-English Tweets using Code-Mixed BERT," *2021 IEEE 20th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC)*, Banff, AB, Canada, pp. 248-252, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[23] Saikat Roy, and Jatinderkumar R. Saini, "Sentiment Classification in Code-Mixed Indo-Aryan Languages: A Transformer-based Survey," *International Conference on Smart Systems: Innovations in Computing*, pp. 391-406, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[24] Namrata Kumari, and Pardeep Singh, "Hindi Text Summarization using Sequence-to-Sequence Neural Network," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 22, no. 10, pp. 1-18, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[25] Nalini S. Jagtap et al., "Improvised Neural Machine Translation Model for Hinglish to English," *International Conference on Data Science and Applications*, pp. 237-247, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[26] Ankur Mangla, Rakesh Kumar Bansal, and Savina Bansal, "Language Identification and Normalization Techniques for Code-Mixed Text," *2024 Sixth International Conference on Computational Intelligence and Communication Technologies (CCICT)*, Sonepat, India, pp. 435-441, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[27] Abhishek Chopra et al., "A Framework for Online Hate Speech Detection on Code-mixed Hindi-English Text and Hindi Text in Devanagari," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 22, no. 5, pp. 1-21, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[28] Vishal Gupta, Ashutosh Dixit, and Shilpa Sethi, "Enhancing Temporal Information Retrieval through Contextual Query Reformulation," *2023 International Conference on Advanced Computing & Communication Technologies (ICACCTech)*, Banur, India, pp. 765-770, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[29] Yinhan Liu et al., "Multilingual Denoising Pre-Training for Neural Machine Translation," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 726-742, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[30] Yuqing Tang et al., "Multilingual Translation from Denoising Pre-Training," *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3450-3466, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[31] Samta Kamboj, Sunil Kumar Sahu, and Neha Sengupta, "DENTRA: Denoising and Translation Pre-Training for Multilingual Machine Translation," *Proceedings of the Seventh Conference on Machine Translation (WMT)*, Abu Dhabi, United Arab Emirates, pp. 1057-1067, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[32] S. Nagesh Bhattu et al., "Improving Code-Mixed POS Tagging using Code-Mixed Embeddings," *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, vol. 19, no. 4, pp. 1-31, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[33] Shayaan Absar, "Fine-Tuning Cross-Lingual LLMs for POS Tagging in Code-Switched Contexts," *Proceedings of the Third Workshop on Resources and Representations for Under-Resourced Languages and Domains (RESOURCEFUL-2025)*, Tallinn, Estonia, pp. 7-12, 2025. [Google Scholar] [Publisher Link]

[34] Rob van der Goot, and Özlem Çetinoğlu, "Lexical Normalization for Code-Switched Data and its Effect on POS-Tagging," *arXiv preprint*, pp. 1-14, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[35] Vidya Saikrishna, D.L. Dowe, and Sid Ray, *MML Learning and Inference of Hierarchical Probabilistic Finite State Machines*, 1st ed., Applied Data Analytics - Principles and Applications, pp. 291-325, 2022. [Google Scholar] [Publisher Link]

[36] Anej Svete, and Ryan Cotterell, "Recurrent Neural Language Models as Probabilistic Finite-State Automata," *arXiv preprint*, pp. 1-18, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[37] Alberto Postiglione, "Finite State Automata on Multi-Word Units for Efficient Text-Mining," *Mathematics*, vol. 12, no. 4, pp. 1-20, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[38] Ganesh Jawahar et al., "Exploring Text-to-Text Transformers for English to Hinglish Machine Translation with Synthetic Code-Mixing," *arXiv preprint*, pp. 1-11, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[39] Linting Xue et al., "mt5: A Massively Multilingual Pre-Trained Text-to-Text Transformer," *arXiv preprint*, pp. 1-17, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[40] Mikhail Krasitskii et al., "Advancing Sentiment Analysis in Tamil-English Code-Mixed Texts: Challenges and Transformer-based Solutions," *arXiv preprint*, pp. 1-8, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[41] El Moatez Billah Nagoudi, AbdelRahim Elmadany, and Muhammad Abdul-Mageed, "Investigating Code-Mixed Modern Standard Arabic-Egyptian to English Machine Translation," *arXiv preprint*, pp. 1-9, 2021. [CrossRef] [Publisher Link]

[42] Shikha Mundra, and Namita Mittal, "CMHE-AN: Code Mixed Hybrid Embedding based Attention Network for Aggression Identification in Hindi English Code-Mixed Text," *Multimedia Tools and Applications*, vol. 82, no. 8, pp. 11337-11364, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[43] K. Sudharsan et al., "A Deep Hybrid Learning Model for Classification of Code-Mixed Text," *2022 International Conference on Futuristic Technologies (INCOFT)*, Belgaum, India, pp. 1-6, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[44] Ayan Sengupta et al., "HIT: A Hierarchically Fused Deep Attention Network for Robust Code-Mixed Language Representation," *arXiv preprint*, pp. 1-15, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[45] Kian Long Tan et al., "RoBERTa-LSTM: A Hybrid Model for Sentiment Analysis with Transformer and Recurrent Neural Network," *IEEE Access*, vol. 10, pp. 21517-21525, 2022. [CrossRef] [Google Scholar] [Publisher Link]