Sarcasm Detection in Hindi-English Code-Mixed Social Media Text

Anonymous ACL submission

Abstract

In this work, we investigate sarcasm detection in Hindi-English code-mixed (Hinglish) social media text. A key challenge in this domain is the scarcity of clean, large-scale annotated resources. It is additionally challenging to process noisy code-mixed language that involves inconsistent transliteration, slang, and emoji usage. We curate and release a balanced Hinglish sarcasm corpus with human and LLM-assisted annotations, and we propose a reproducible preprocessing pipeline that normalises emojis, standardises slang, and handles script variation. We also provide baseline experiments to illustrate the utility of the resulting resource. By focusing on data creation and cleaning, our work establishes a foundation for future research in code-mixed Hinglish sarcasm and related tasks.

1 Introduction

With the proliferation of multilingual social media usage in India, Hindi-English code-mixed (commonly referred to as "Hinglish") text has become increasingly prevalent on online platforms. This unique and informal structure often blends languages within a single sentence, presenting new challenges for Natural Language Processing (NLP) systems, particularly in subjective tasks like sarcasm detection where context, tone, and language style are critical (Aggarwal et al., 2020). Sarcasm detection itself is a non-trivial problem, as it involves recognizing subtle contradictions between literal and intended meanings. In code-mixed scenarios, this complexity is amplified by inconsistent grammar, informal spellings, transliterated words, and the heavy use of emojis and internet slang (Wadhawan and Aggarwal, 2021).

We explore sarcasm detection in Hinglish social media text using a transformer-based model, HingBERT, pre-trained on Hindi-English code-mixed data. Unlike prior works that used multilingual models like mBERT (Devlin et al., 2019) or handcrafted features, our approach integrates linguistic preprocessing, such as emoji normalization, slang handling, and Devanagari-to-Roman transliteration, to better model Hinglish text. Our contributions are: (i) a unified *raw* corpus built from multiple public and private sources with a hybrid (human + LLM) labeling strategy; (ii) a carefully engineered cleaning and normalisation pipeline tailored to emojis, Romanised Hindi, and social-media artefacts; (iii) detailed dataset analyses (code-mix bands and emoji prevalence) that surface where sarcasm signals concentrate; and (iv) reproducible benchmarks with a Hinglish-specific encoder. Ultimately, our goal is not to develop a production-ready system, but to investigate how HingBERT performs on code-mixed sarcasm classification when paired with lightweight preprocessing strategies. The insights gained from this study contribute to a better understanding of how language-specific models can be adapted for real-world multilingual social media tasks.

2 Related Work

Irony and sarcasm detection has largely shifted from feature-based models popular in the early 2000s and 2010s (Parde and Nielsen, 2018) to transformer encoders, often augmented with social or conversational context. Code-mixed settings amplify orthographic variation and semantic drift; our findings align with this trend and emphasise the value of domain-adapted pretraining and style-aware normalisation. Sarcasm detection in multilingual and code-mixed contexts, and especially Hinglish, remains challenging because of noisy user-generated text and frequent code-switching. Aggarwal et al. (2020) created a 100K-instance Twitter corpus and showed that embeddings plus CNN/BiLSTM models can detect sarcasm, but their approach suffered from hashtag-

based noisy labels and lacked contextual depth. Swami et al. (2018) highlighted the difficulties of annotation and token-level language tagging in this setting in their early Hinglish sarcasm dataset.

Jain et al. (2020) proposed deep-learning approaches for English-Hindi tweets, while Shah and Maurya (2021) modeled incongruity via subword embeddings, showing that capturing lexical mismatch is critical for sarcasm. More recently, Mazumder and Naskar (2025) explored data-centric strategies (adding synthetic monolingual samples, multitasking, and instruction fine-tuning) for sarcasm and humour in Hinglish. Emotion recognition studies such as Wadhawan and Aggarwal (2021) confirmed the usefulness of transformer encoders for Hinglish, yet they did not target sarcasm and ignored multimodal cues. Multimodal sarcasm detection has since been explored by Bedi et al. (2021), who built a Hindi–English dataset combining textual and visual cues.

On the code-mixed modelling side, Patil et al. (2023) benchmarked multiple BERT variants on Hinglish sentiment tasks, while Singh and Lefever (2020) projected Hinglish and English embeddings into a shared space for SemEval-2020 sentiment analysis. Advani et al. (2020) and Singh (2021) demonstrated that strong feature engineering and preprocessing remain competitive baselines. Several newly released Hinglish data resources further motivate advanced modeling. The English-Hindi Code-Mixed Parallel Corpus (Dhar, 2021) provides aligned codemixed/monolingual pairs; IIITH-CodeMix sentiment data (Tatariya, 2022) and Indo-HateSpeech (Kaware, 2023) expand the label space beyond sarcasm to sentiment and hate speech. HINMIX (Aggarwal, 2023) offers 4.2M synthetic parallel sentences, while the Hinglish-TOP semantic-parsing dataset (Vinayan, 2021) supplies 180K intent utterances. PHINC, a parallel Hinglish–English MT corpus (Srivastava and Singh, 2020), underlines the importance of large-scale curated codemixed resources.

We incorporate samples from these corpora along with additional in-house and synthetically augmented texts into a unified dataset and analyse how code-mixing intensity affects sarcasm classification. Unlike previous studies, our work jointly (i) fine-tunes a Hinglish-specific transformer, (ii) injects multimodal cues (emojis and punctuation), and (iii) conducts an ablation on code-switching levels across heterogeneous corpora.

ID	Source	Domain	Rows
EH-Parallel	(Dhar, 2021)	Twitter + news	25k
IIITH-CM	(Tatariya, 2022)	Twitter (sentiment)	18k
IndoHateSpeech	(Kaware et al., 2023)	Twitter + YouTube	15k
HINMIX	(Aggarwal, 2023)	Synthetic MT pairs	40k
Hinglish-TOP	(Vinayan, 2021)	WikiHow utterances	12k

Table 1: External datasets incorporated as data sources.

3 Dataset and Preprocessing

3.1 Data Sources

We sourced our data from the five public Hinglish resources outlined in Table 1, along with additional instances downloaded from Twitter and YouTube scrapes (~2k) and WhatsApp and Discord logs (~1k). We then augmented these sources with synthetically-generated text using ChatGPT (~4k). In total, the dataset included ~122k rows after de-duplication. To keep annotation costs manageable, we drew a stratified random sample of 25,000 instances from the overall dataset for further annotation, preserving the source/domain proportions of the full pool.

We make the raw data (i.e., post-deduplication but prior to additional cleaning) publicly available `[link redacted for review]`. All processed splits were produced deterministically using the pipeline described in §3.3, and exact scripts and command line prompts to reproduce these will also be provided in the data repository.

3.2 Labelling Protocol

The 25k sample was annotated using *Sarcastic* and *Not Sarcastic* class labels using a hybrid strategy designed to balance quality and cost. 4 Volunteers were provided with a plain-text list of Hinglish comments (exported from the CSV into a non-specialist readable format, typically with batches of about 100 examples at a time). For each entry, they were asked to assign one of two labels: *Sarcastic* (if the comment conveyed irony, ridicule, or mocking intent) or *Not Sarcastic* (otherwise). To ensure consistency, the dictionary meaning of sarcasm was included in the instructions as: *"the use of irony to mock or convey contempt."* Annotators were instructed to rely only on the text provided, without basing their decision on outside context. They were also informed that the data consisted

of publicly available social media comments, that their participation was voluntary, and that there were no risks associated with the task.

A subset of the overall data (40%; 10,000 instances) was double-annotated by the authors together with three fluent Hinglish speakers who volunteered their time. Inter-annotator reliability calculated on this subset was high (Cohen's κ=0.84). To reduce annotation costs, the remaining data (60%; 15,000 instances) was labelled using five-shot prompting with GPT-3.5. A manual audit of 1,000 synthetically labelled instances revealed a 93.7% agreement with human labels. For the remaining instances, the labels were corrected and these representative cases were added as additional demonstration samples in the few-shot prompts to improve synthetic annotation consistency.The resulting dataset is manually class-balanced (12,543 sarcastic / 12,457 non-sarcastic).

3.3 Cleaning Pipeline

Every sentence is normalised using `clean_tweet` (taking ~1.3 ms per sentence), a function that sequentially applies 18 cleaning operations outlined in Listing 5. The cleaning operations were designed to handle noisy social media text, retain affective cues, and enable consistent modelling across scripts. Table 2 summarises key transformations resulting from the cleaning process. Four representative normalisation resources are shown in Listings 1–4, covering profanity detection, emoticon parsing, and Hindi word filtering.[1]

Listing 1: Roman–English swear filter

```
censored_swear_words = {
  # s!@#, s\#!t, s**t → shit
  r'\b[Ss][^A-Za-z\s]{3,4}\b'
    ↪ : 'shit',
  r'[Ss][\!\@\#\$%\^&\*]*[i1\|].*?[t]+'
    ↪ : 'shit',
}
```

Listing 2: Hindi Roman/Devanagari profanity map

```
hindi_swear_map = {
  # madarchod (Roman + leet + Devanagari
    ↪ )
  r'\b(m[a@]d[ae]?[^ \w]*|' +
    r'\u092E\u093E.\u0926\u0930.\u091A\
    ↪ u094B\u0926)\b' : 'madarchod',

  # bhenchod / behenchod
  r'\b(bh?e+h[e]*n[^ \w]*|' +
    r'\u092D\u0947.\u091A\u094B\u0926)\b
    ↪ '                : 'bhenchod',
}
```

[1]Full code and dictionaries are available at: `[link redacted for review]`.

Stage	Purpose
Emoji → tokens	`:joy:` → `:face_joy:` (retain affect)
Encoding repair	Fix mojibake / HTML (`&` → `&`)
Profanity normalisation	70 English & Hindi obfuscations (`m@d@rch*d` → *madarchod*)
User/hashtag masking	`@rahul` → *SomeTaggedAccount*; `#Budget2025` → *Budget2025*
Contraction expansion	*can't* → *cannot*
Unicode-safe punctuation	Keep `?` `!` `.` `,` + Latin/Devanagari letters/digits
Optional transliteration	*tum* → *tum* (ITRANS)
Whitespace squash	Final trim (remove surplus spaces)

Table 2: Key steps in the preprocessing pipeline.

Listing 3: Legacy ASCII emoticons

```
ascii_emoticons = {
  ':-)' : 'smiley_face',
  ':-D' : 'grinning_face',
  ':-(' : 'sad_face',
}
```

Listing 4: Sample entries from the Roman-Hindi lexicon

```
# Top entries from
    ↪ roman_hindi_lexicon.txt
baat
baata
baate
baatein
baaten
baaton
pyar
dil
zindagi
sach
sahi
kya
kyun
haan
nahi
banda
```

Specific cleaning processes included:

- *Emoji and HTML handling*: Emojis are demojized (→ `:face_joy:`), while HTML entities and tags are decoded and stripped.
- *Profanity filters*: Regex-based normalisation of Roman-leet and Devanagari obfuscations (Listings 1 and 2).
- *Noise masking*: URLs and email addresses are anonymized, hashtags are stripped of

their symbols, and mentions like `@user` are mapped to a generic token.

- *Text simplification*: Common contractions are expanded (*can't* → *cannot*), and non-essential punctuation is removed.
- *Code-mix features*: Devanagari text is optionally transliterated to Roman using ITRANS, and ASCII emoticons are replaced with affect tags (Listing 3).
- *Lexicon match*: Words with high Hinglish phonetic plausibility are collected into a custom Roman-Hindi dictionary (Listing 4) to support token-level features and filtering.

The complete pipeline is executed in the following fixed order:

Listing 5: Order of operations in `clean_tweet`

```
cleaning_functions = [
    replace_emojis,
    normalize_text,
    decode_html_entities,
    remove_html_tags,
    replace_urls,
    replace_emails,
    replace_censored_swear_words,
    replace_hindi_swears,
    replace_mentions,
    replace_at_symbol,
    remove_hashtags,
    expand_contractions,
    clean_unicode_punct,
    transliterate_devanagari,
    remove_emoji_colons,
    replace_ascii_emoticons,
    replace_trade_mark,
    remove_extra_whitespace
]
```

Finally, each instance in the dataset was assigned to one of five buckets (0–20%, 20–40%, 40–60%, 60–80%, and 80–100% Hindi tokens) to enable a stratified evaluation across different types of Hinglish code-switching. Here, "Hindi%" denotes the proportion of tokens identified as Hindi; mid-range values (40–60%) indicate peak code-mixing, whereas high values (80–100%) are closer to monolingual Hindi and low values are closer to monolingual English.

4 Experimental Methods

To assess the utility of our data and preprocessing pipelines within this task domain, we performed empirical analyses benchmarking both standard statistical machine learning and neural network models and more contemporarily popular transformer-based models. We outline experimental methods and conditions in §4.1–§4.4.

4.1 Modeling Conditions

We performed benchmarking experiments using support vector machine (SVM), bidirectional long short-term memory network (BiLSTM), and two transformer-based architectures (mBERT and HingBERT). For *SVM*, our statistical machine learning baseline, a linear SVM was trained on TF-IDF vectors extracted from the pre-processed text; preprocessing was applied as described for our pipeline. We note that although computationally efficient, TF-IDF ignores word order and context, giving it limited suitability to capture the nuanced irony present in code-mixed Hinglish. Nevertheless, its inclusion establishes a performance floor and is a useful point of comparison against modern neural and transformer architectures to highlight the improvements gained from context-aware approaches.

We implemented a two-layer *BiLSTM* initialised with randomly generated word embeddings since there are no large publicly available Hinglish embeddings. This condition presents a step beyond what we anticipated to be achievable using statistical machine learning models. However, although bidirectional recurrence models temporal context in both directions, we anticipated that performance for this condition may be limited by data noise and the model's comparatively shallow representation power.

For transformer-based architectures, we first fine-tuned the `bert-base-multilingual-cased` checkpoint, pretrained on 104 languages (Devlin et al., 2019), on our data; this model is commonly referred to as *mBERT*. mBERT provides strong multilingual contextual embeddings, yet it is not optimised for informal syntax, Romanised Hindi spellings, or the idiosyncratic slang commonly observed in social-media code-mixing.

Finally, we fine-tuned *HingBERT*, a transformer-based model pre-trained on the L3Cube-HingCorpus (Nayak and Joshi, 2022) that is explicitly adapted to Hinglish. Our expectation was that fine-tuning HingBERT on our data would enable the model to leverage prior exposure to non-standard spellings, frequent code-switching, and emoji- heavy discourse, giving it a performance edge over other more

generic multilingual encoders.

4.2 Model Architecture and Tokenisation

We otherwise held most settings consistent across modeling conditions. All transformer-based models shared the same classification head, differing only in the frozen encoder weights from which they were initiated. We used the HuggingFace `AutoTokenizer` that belonged to each checkpoint, thus also applying identical sub-word vocabulary and normalisation rules to what the encoder saw during pre-training.

Embedding layers for each token were defined as the concatenation of token, positional, and (where applicable) segment embeddings. Twelve transformer layers were used in the encoder stack (containing multi-head self-attention and feedforward layers) to contextualise each subword. The final hidden state of the `[CLS]` token was taken as the aggregate representation of the sentence, or the sentence vector. We applied a dropout layer in the classification head followed by a single fully-connected neuron to produce one logit, converted to a probability using the `sigmoid` function. We applied a binary cross-entropy loss with logits, and implemented dynamic padding using `DataCollatorWithPadding` so that each batch contained sequences of identical length, reducing GPU waste.

4.3 Training Configuration

All transformer variants were also trained using identical hyper-parameters. We used an AdamW optimiser with weight decay set to 0.01, a learning rate of 2×10^{-5} with linear warm-up and then decay, and a batch size of 16 (selected to allow training to fit on a single consumer-grade GPU), training over four epochs (with early stopping based on validation F_1).

All experiments were conducted on a single NVIDIA GeForce RTX 4090 GPU (24 GB VRAM) paired with a 12th Gen Intel Core i7–12700K CPU and 64 GB system RAM. We used PyTorch version 2.5.1 with CUDA version 12.1 and Hugging Face Transformers version 4.33. Fine-tuning HingBERT for four epochs with batch size 16 typically required 3–4 hours; classical baselines trained in under 30 minutes.

4.4 Dataset Split and Evaluation

From the 25,000 labelled sentences (§3) we allocate 70% for training and 30% for validation using stratified sampling to preserve the sarcasm ratio (50.8% vs. 49.2%). We evaluated performance on the basis of accuracy, recall, and F_1 (the primary metric). Accuracy provided the fraction of correct predictions from among all predictions, recall ($\frac{TP}{TP+FN}$) was used to measure the extent to which all sarcastic instances were captured, and F_1 provided the harmonic mean of precision and recall. All experiments were repeated with three random seeds, and we report mean scores for all results.

Model	Accuracy	Recall	F1
TF–IDF + SVM	63.56 %	63.56 %	77.51 %
BiLSTM	47.93 %	47.93 %	64.37 %
mBERT	77.61 %	77.61 %	87.24 %
HingBERT	**87.35 %**	**87.35 %**	**93.18 %**

Table 3: Validation results on 25k instances. HingBERT improves F1 by **+5.9** over mBERT and **+15.7** over the best non-transformer baseline (TF–IDF+SVM).

Hindi %	SVM	BiLSTM	mBERT	HingBERT
0–20	0.635	0.317	0.534	**0.878**
20–40	0.664	0.565	0.771	**0.904**
40–60	0.710	0.597	0.858	**0.905**
60–80	0.796	0.671	0.891	**0.944**
80–100	0.795	0.662	0.885	**0.934**

Table 4: F1 by code-mix level. HingBERT sustains > 0.87 across the board, peaking in the 60–80 % range.

5 Results

We evaluate all models on the 30% validation split (§4.4). Our findings establish the utility of our data curation and preprocessing pipelines, confirm the limitations of traditional and generic multilingual models, and underline the importance of culturally tailored NLP systems. HingBERT outperformed every alternative modeling condition on all metrics (Table 3), including by a decisive $+5.9$ F_1 over the multilingual BERT encoder. The illustrates both the value of Hinglish pre-training and the effect of the emoji/slang–aware cleaning pipeline. To probe the robustness of this finding, we compared performance for each Hinglish code-switching level (introduced in §3.3) and report F_1-scores per code-switching bucket in Table 4.

We observe that HingBERT remains stable, whereas SVM and BiLSTM degrade as code-mixing increases. Here, the "Hindi%" values denote the proportion of tokens identified as Hindi in a comment. To recall, high values (80–100%) cor-

Ablation	F1 Δ
– Emoji aliasing	−1.1
– ASCII emoticons	−0.8
– Roman Hindi normalisation	−1.7

Table 5: Ablation of cleaning steps on HingBERT. Negative values indicate F1 drop relative to the full pipeline.

respond to texts that are mostly Hindi and therefore less code-mixed, while mid-range values (40–60%) indicate balanced Hindi–English usage, i.e., peak code-mixing. We observe that moderate mixing (60–80%) is most informative (Table 4), being rich enough to reveal sarcasm cues yet not so Hindi-heavy that Romaniszed spellings dominate.

To further assess the contributions of our preprocessing enhancements, we ablated three cleaning steps on HingBERT. Table 5 shows the F_1 drops observed when each preprocessing component was removed. A key finding of this ablation was that Hinglish sarcasm is strongly signalled by non-standard orthography and pictographic emotion markers. Overall, our experiments highlight three important insights. First, domain-specific pre-training on Hinglish data produces substantial accuracy gains compared to generic multilingual models. Second, we find that sarcasm cues are most salient under moderate levels of code-mixing, where Hindi and English tokens co-occur in balanced proportion. Finally, we show that targeted preprocessing, and particularly the normalisation of emojis and slang, provides an additional 1–2 F_1 points, underscoring the role of non-standard orthography and informal markers in Hinglish sarcasm.

6 Discussion

Our experiments highlight two overarching themes: *(i)* sarcasm in Hinglish is best modelled with resources that mirror its informal, culture-laden style, and *(ii)* seemingly "decorative" tokens (emojis, emoticons, creative spellings) are, in fact, essential semantic cues. Importantly, HingBERT, pre-trained on the L3Cube-HingCorpus, encounters the same non-standard spellings, romanised Hindi, and emoji sequences that pervade our dataset. This prior exposure explains its +5.9 F_1 edge over mBERT and its robustness across code-mixing bands.[2] Our experiments also broadly establish the utlity of our data curation and preprocessing pipelines.

6.1 Additional Analyses

Some work proposes translating Hinglish to English before downstream tasks. Srivastava and Singh (2020) show that this is feasible for machine translation, yet our pilot tests confirm that translation blunts sarcasm. Table 6 contrasts two sentences: one whose ironic bite vanishes on translation and one that remains literal. The first tweet alludes to *notebandi* (the 2016 Indian currency demonetisation); the pun on the word "note" is untranslatable and its sarcasm disappears in English. In contrast, the second example relies on a literal factual assertion, "No chance of falling from this seat," that retains its meaning under translation and does not depend on culture-specific wordplay. These cases illustrate why training directly on Hinglish is preferable; namely, it preserves both the cultural cues and the informal orthography that signal sarcasm.

We observed that ablating emoji aliasing, ASCII-face normalisation, or Roman-Hindi lexicon resulted in performance costs of up to 1.7 F_1. This suggests that these tokens encode meaningful dimensions of affect. Models ignorant of emoji semantics mis-classify emotionally charged remarks, echoing observations by Aggarwal et al. (2020) when studying sarcasm in Twitter data. Finally, combining human and GPT-3.5 annotations (40/60 split) yielded a reliable dataset of 25 k instances ($\kappa = 0.84$ among humans, with 93.7 % human–GPT agreement in a manual analysis of a subset of the synthetically generated labels). While large-language-model labelling clearly accelerates corpus creation, we emphasize that periodic audits remain vital when building classifiers.

6.2 Implications

Our findings yield several broader implications for sarcasm detection in code-mixed settings. First, tailored pre-training appears to be indispensable. HingBERT consistently outperformed the more generic mBERT, which struggled with Romanised Hindi spellings such as *zindagi* (life) or *pyaar* (love) that are poorly represented in multilingual vocabularies. Second, we observed that moderate code-switching (~70% Hindi) delivered the richest sarcasm signal. For instance, sentences like

[2] Peak performance in the 60–80 % band suggests that *moderate* switching injects maximal sarcastic signal without overwhelming the model with out-of-vocabulary forms.

Code-mixed sentence	Literal EN translation	Meaning preserved?
Note kr lijiye.. *Bandi chal rahi h ;)*	Please note. . . the girl is going on ;)	No
is seat me *girne ka chance nhi hai*	No chance of falling from this seat	Yes

Table 6: Literal translation often erases cultural or pun-based meaning (top row), while purely literal statements (bottom row) are preserved.

"Bhai, tum toh real hero ho, asli Salman Khan nikle!" ("Bro, you are a real hero, turned out to be the real Salman Khan!") combine Hindi syntax with English cultural references to cue sarcasm, whereas near-monolingual Hindi texts lose this contrast.

Third, our experiments confirmed that relying on translation may result in unstable performance. The tweet *"Note kr lijiye, bandi chal rahi h ;)"*, which sarcastically alludes to the 2016 Indian *notebandi* policy, loses its pun and ironic bite when translated literally as "Please note, the girl is going on ;)". Such culturally loaded sarcasm cannot be captured through machine translation alone, suggesting that directly incorporating both code-switching languages into the modeling approach may yield more reliable gains.

Finally, our findings confirm the utility of normalising informal artefacts such as emojis and slang. Expressions like *"Lo or kro Rahul Gandhi :face_with_tears_of_joy: :cross_mark:"* carry sarcastic intent through the juxtaposition of text and pictographic cues; removing the emoji entirely would erase the signal. Normalisation preserves their affective value while keeping the text machine-readable. Taken together, these observations position HingBERT plus a stylistic pre-processing pipeline such as that proposed here as a strong benchmark approach for Hinglish sarcasm. This may translate to a more generalisable template for handling other low-resource, code-mixed languages settings for which sarcasm intertwines with culture-specific cues; future experiments could probe this further.

7 Future Work

While the current study shows that HingBERT with tailored pre-processing is highly effective for sarcasm detection in Hinglish, several promising directions remain. One immediate extension is multimodal sarcasm detection, since sarcasm on social media is often conveyed through the interaction of text with images, memes, GIFs, or short videos. Future work can therefore combine the present text pipeline with vision–language models such as CLIP (Radford et al., 2021) or Flamingo (Alayrac et al., 2022) to capture cross-modal incongruity. Another promising direction is context-aware modelling. Replies, quote-tweets, and conversational threads often provide pragmatic cues that are unavailable in isolated utterances, suggesting the value of hierarchical transformers or retrieval-augmented encoders that incorporate broader dialogue context. Beyond that, it would be valuable to study cross-lingual generalisation to other code-mixed settings such as Spanglish, Taglish, or Arabizi, where similar orthographic noise and stylistic play are present. Adapting the current pre-processing and fine-tuning pipeline to these and other settings may support a broader family of code-mixed sarcasm detectors.

Further work is also needed on interpretability, robustness, and deployment. Attention visualisation, token-level attribution methods such as SHAP, (Lundberg and Lee, 2017) or exemplar-based explanations could make model decisions more transparent and increase trust in downstream moderation or analytics settings. At the same time, robustness to orthographic variation remains essential, since Hinglish slang evolves rapidly. Phonetic subword representations, meta-learning for unseen slang, or continual fine-tuning on streaming data may improve resilience to such drift. A human-in-the-loop setup based on active learning could further improve annotation efficiency by prioritising low-confidence or high-disagreement examples. Finally, future systems should be evaluated for fairness and bias, especially because sarcastic content may target political or demographic groups, and should also be studied from an engineering perspective for real-time deployment. Compact variants such as DistilHingBERT may enable low-latency use in chatbots, moderation pipelines, or sentiment dashboards, making inference speed and memory profiling an important next step.

8 Conclusion

This study introduced a complete pipeline for sarcasm detection in Hindi–English code-mixed social-media text. We (i) curated and labelled a 25k Hinglish dataset using a combination of manual and synthetic annotations; (ii) proposed a linguistically motivated preprocessing suite that normalises emojis, ASCII emoticons, Roman-Hindi spellings and Devanagari tokens; and (iii) benchmarked multiple statistical machine learning and transformer-based models for Hinglish sarcasm detection, establishing utility of the data curation and preprocessing pipeline in this setting.

Experimental results show that HingBERT, when fine-tuned on the cleaned dataset, achieves the best overall F_1 (93.18%), outperforming mBERT by 0.06 and the strongest non-transformer baseline by 0.16. Performance is consistent across all five code-mixing bands, with the 60–80% Hindi bucket yielding the highest F_1, suggesting that moderate code-switching can be most effectively captured by this approach. Ablation studies confirm that emoji and slang normalisation contribute a further 0.01–0.02 in F_1, underscoring the importance of stylistic cues in sarcastic Hinglish.

Taken together, these findings emphasise that culturally aware pretraining plus task-specific cleaning is crucial for robust sarcasm detection in low-resource, code-mixed settings. Beyond sarcasm, our methodology may provide a template transferable to sentiment, hate-speech or stance analysis in other mixed-language communities such as Spanglish, Taglish or Arabizi. All code, cleaned data splits, and baseline checkpoints are available at: `[link redacted for review]`.

9 Ethics Statement

Our corpus aggregates publicly available social-media text and private logs provided with consent. We remove direct identifiers when present, anonymise mentions and URLs, and release only raw text that complies with the respective sources' terms of use. Sarcasm frequently targets individuals or groups; we caution against downstream use for automated moderation without human oversight. We audit label quality and report potential biases (e.g., topic and platform skew; devanagari, dialects) in our dataset analysis.

10 Limitations

Our labels partly rely on LLM-assisted annotation; despite audits, residual errors and style biases may persist. The corpus is primarily short-form text; multimodal sarcasm (memes, images) is out of scope. Our cleaning pipeline is tuned to Hinglish orthographic conventions and may require adaptation for other code-mixed varieties.

A Dataset Card (Summary)

Language variety: Hindi–English code-mixed (Hinglish), Roman and Devanagari scripts.
Domains: Twitter, YouTube, chats; plus public Hinglish resources.
Task: Binary sarcasm detection.
Size: ∼25k labelled instances after sampling; class-balanced.
Labeling: 40% double-annotated by fluent speakers; 60% LLM-assisted with audits.
Known limitations: Topic/platform skew; evolving slang; partial LLM labels.
Potential risks: Misclassification of sensitive content; demographic bias.
License/usage: Use subject to original sources' terms; redistribution of raw text may be limited.

References

Gaurav Advani, Ankit Singh, Asif Ekbal, and Pushpak Bhattacharyya. 2020. C1 at semeval-2020 task 9: Feature engineering for sentiment analysis of code-mixed social media data. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, pages 1251–1256, Barcelona, Spain (Online). International Committee for Computational Linguistics.

Akshita Aggarwal, Anshul Wadhawan, Anshima Chaudhary, and Kavita Maurya. 2020. "Did You Really Mean What You Said?": Sarcasm detection in hindi–english code-mixed data using bilingual word embeddings. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 7–15. Association for Computational Linguistics.

Kartik Aggarwal. 2023. HINMIX_hi-en: A massive parallel Hindi–English code-mixing dataset. Accessed: 2025-10-03.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Isaac Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhengqi Gong, Aidan Clark, Matthew Botvinick, Andrew Zisserman, Karen Simonyan,

and Oriol Vinyals. 2022. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*.

Kanika Bedi, Praveen Chandra, and Dipti Misra Sharma. 2021. Multi-modal sarcasm detection and humor classification in code-mixed conversations. In *Proceedings of the 23rd ACM International Conference on Multimodal Interaction (ICMI)*, pages 580–584, Montreal, QC, Canada. Association for Computing Machinery.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186. Association for Computational Linguistics.

Mrinal Dhar. 2021. English–Hindi Code-Mixed to Monolingual English Parallel Corpus. GitHub repository. Accessed: 2025-10-03.

Anubhav Jain, Yogesh Kumar, and Manish Shrivastava. 2020. Sarcasm detection in mash-up language using soft attention-based bidirectional lstm and convolutional neural network. *Applied Soft Computing*, 91:106198.

Pravin Kaware. 2023. Indo-HateSpeech: A Hindi–English Code-Mixed Hate-Speech Dataset. Accessed: 2025-10-03.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Sayantan Mazumder and Sudip Kumar Naskar. 2025. Revealing the impact of synthetic native samples and multi-tasking strategies in hindi-english code-mixed humour and sarcasm detection. *arXiv preprint arXiv:2412.12761*.

Ravindra Nayak and Raviraj Joshi. 2022. L3Cube-HingCorpus and HingBERT: A code mixed hindi–english dataset and BERT language models. In *Proceedings of the WILDRE-6 Workshop within the 13th Language Resources and Evaluation Conference*, pages 7–12, Marseille, France. European Language Resources Association.

Natalie Parde and Rodney Nielsen. 2018. Detecting sarcasm is extremely easy ;-). In *Proceedings of the Workshop on Computational Semantics beyond Events and Roles*, pages 21–26, New Orleans, Louisiana. Association for Computational Linguistics.

Pranav Patil, Rohit Patil, and Raviraj Joshi. 2023. Comparative study of pre-trained bert models for code-mixed hindi–english data. *arXiv preprint arXiv:2305.15722*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR.

Shubham Shah and Kavita Maurya. 2021. How effective is incongruity? implications for code-mixed sarcasm detection. In *Proceedings of the 18th International Conference on Natural Language Processing (ICON)*, pages 243–251, Silchar, India. NLP Association of India.

Gaurav Singh. 2021. Sentiment analysis of code-mixed social media text (hinglish). *arXiv preprint arXiv:2102.12149*.

Gaurav Singh and Els Lefever. 2020. Lt3 at semeval-2020 task 9: Cross-lingual embeddings for sentiment analysis of hinglish social media text. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, pages 1219–1224, Barcelona, Spain (Online). International Committee for Computational Linguistics.

Vivek Srivastava and Mayank Singh. 2020. PHINC: A parallel hinglish social media code-mixed corpus for machine translation. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 41–49, Online. Association for Computational Linguistics.

Sahil Swami, Ankush Khandelwal, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. 2018. A corpus of english–hindi code-mixed tweets for sarcasm detection. *arXiv preprint arXiv:1805.11869*.

Kushal Tatariya. 2022. IIITH-CodeMix Hindi–English Sentiment Dataset (split revisions). GitHub repository. Accessed: 2025-10-03.

Karthik Vinayan. 2021. Hinglish-TOP: Code-Switched Semantic Parsing Dataset. Accessed: 2025-10-03.

Anshul Wadhawan and Akshita Aggarwal. 2021. Towards Emotion Recognition in Hindi-English Code-Mixed Data: A Transformer Based Approach. In *Proceedings of the Eleventh Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA)*, pages 195–202, Online. Association for Computational Linguistics.