Sarcasm Detection and Translation to Hindi While Preserving Sarcastic Tone

Sujal Awasthi

Abstract

This paper presents a dual-stage framework for automatic sarcasm detection and sarcasm-preserving translation from English to Hindi. Using transformer-based deep learning models, the first stage determines whether English tweets or headlines are sarcastic or not. If sarcasm is detected, the text is passed to a fine-tuned MBART model for translation. Evaluation using backtranslation and semantic similarity scoring provides insights into sarcasm preservation quality.

1 Introduction

Sarcasm is a type of figurative language in which the literal expression and the intended meaning are different. Particularly in cross-lingual situations like English to Hindi, it presents difficulties for both detection and translation tasks. The purpose of this study is to:

1. Detect sarcasm in English text.
2. Translate sarcastic content into Hindi while preserving tone and meaning.

2 Problem Statement

Traditional machine translation tools don't work well with sarcasm. It's also not easy to tell when someone is being sarcastic in noisy, casual social media text (like tweets). This study presents a resilient dual-stage framework for the identification of sarcasm and the preservation of sarcasm in translation.

3 Literature Review

Previous studies, such as Commander-GPT, employed modular LLM pipelines for sarcasm reasoning. Other research examined byte-level models (e.g., ByT5, CANINE) for resilience against noisy text. Nonetheless, the majority failed to tackle the preservation of sarcasm across languages.

4 Dataset Description

- **Tweet Sarcasm Dataset:** 1,015,771 English tweets labeled as sarcastic or non-sarcastic. Includes features like sentiment score, subjectivity, punctuation indicators, and all caps usage.
- **Contextual Sarcasm Dataset:** 6,520 rows with sarcastic sentences containing context. Useful for detecting implicit sarcasm.
- **News Headlines Dataset:** 44,294 news headlines labeled with sarcasm, along with URLs for full article extraction.

5 Data Preprocessing

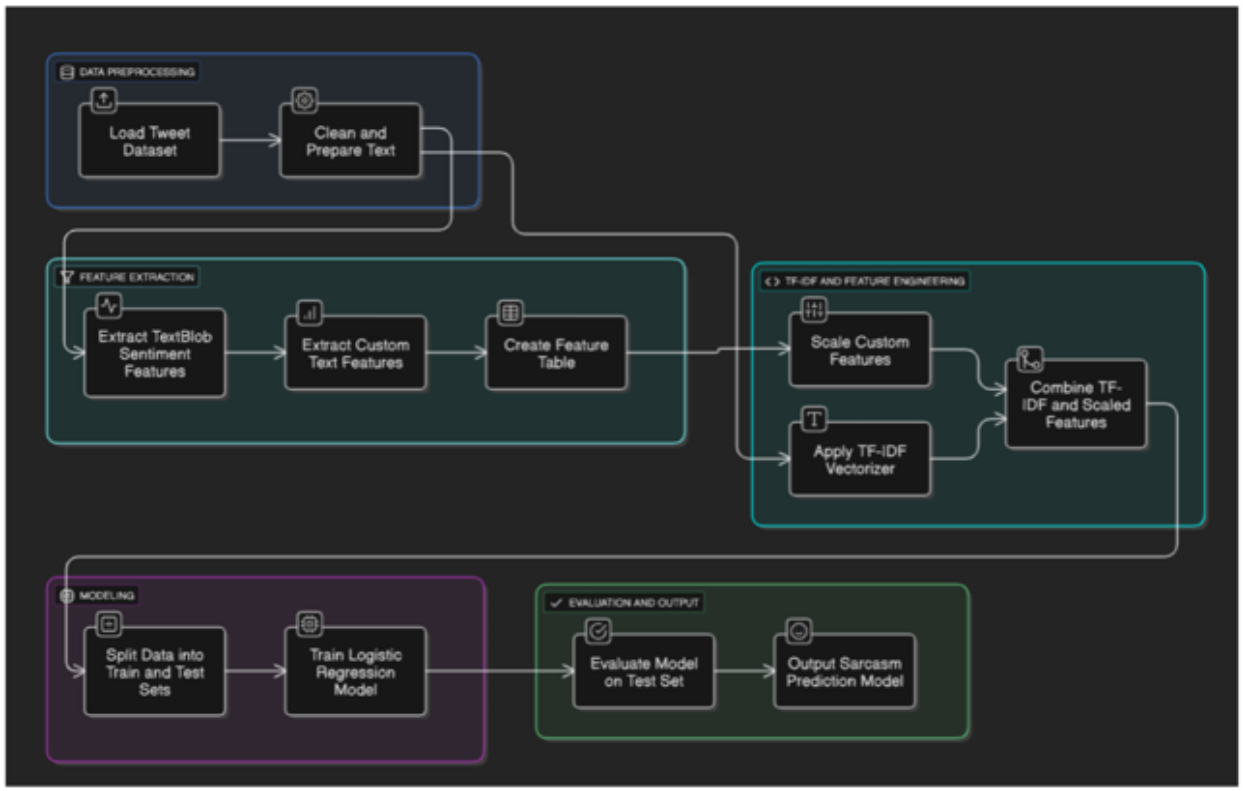
To get the datasets ready for both sarcasm detection and translation tasks, they had to be preprocessed properly. The same steps were used on all text datasets:

- **Tokenization:** Texts were tokenized using standard NLP tokenizers to split the input into individual tokens suitable for model ingestion.
- **Lowercasing:** All characters were converted to lowercase to ensure uniformity and reduce variability due to case sensitivity.

Only tweets that were tagged as sarcastic were sent to the translation pipeline for the translation task. We used these preprocessed English tweets to make the MBART model better at translating from English to Hindi. Using the provided URLs, full article texts were scraped from the news dataset to add contextual information to the headline data. This made it easier to find sarcasm in longer and more formal text.

6 Model Architectures

6.1 TF-IDF + Logistic Regression



Using conventional machine learning methods enhanced with linguistic features, this model acts as a baseline for sarcasm detection. It combines syntactic and semantic features that were extracted using the Term Frequency–Inverse Document Frequency (TF-IDF) representation of the tweet text with `TextBlob` library.

Feature Extraction

Each tweet is processed to compute the following features:

- **Sentiment Polarity:** A score ranging from -1 (negative) to 1 (positive), indicating the emotional tone.
- **Subjectivity:** A value between 0 and 1 representing how subjective or opinionated the tweet is.
- **Length:** Number of characters in the tweet.
- **Exclamations:** Count of exclamation marks, often associated with exaggeration.
- **Questions:** Count of question marks, sometimes indicating rhetorical sarcasm.
- **All-Caps Words:** Number of fully capitalized words to capture emphasis or shouting.

The features were computed using the following logic:

```
from textblob import TextBlob

def extract_text_features(text):
    blob = TextBlob(text)
    return {
        "sentiment": blob.sentiment.polarity,
        "subjectivity": blob.sentiment.subjectivity,
        "length": len(text),
        "exclamations": text.count("!"),
        "questions": text.count("?"),
        "all_caps": sum(1 for word in text.split() if word.isupper())
    }
```

TF-IDF Vectorization and Scaling

The tweets were also transformed into sparse vectors using **TF-IDF**, limited to the top 1000 words by frequency:

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(max_features=1000)
X_tfidf = tfidf.fit_transform(df["tweet"])
```

The extracted numeric features were normalized using standard scaling:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_custom_scaled = scaler.fit_transform(custom_features)
```

Model Training and Evaluation

Both TF-IDF and linguistic features were concatenated:

```
from scipy.sparse import hstack
X_combined = hstack([X_tfidf, X_custom_scaled])
```

A logistic regression model was trained and evaluated using standard classification metrics:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=2000)
model.fit(X_train, y_train)
```

Evaluation Results:

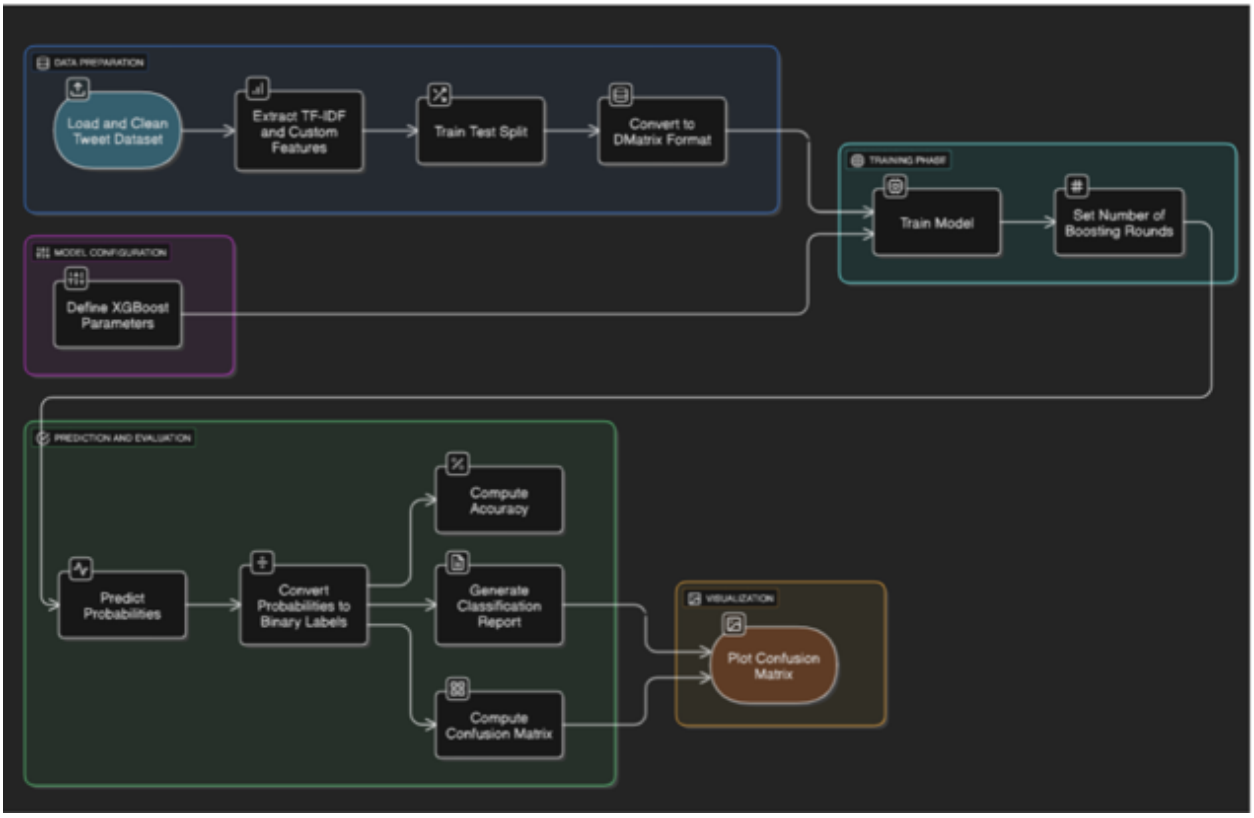
- **Accuracy:** 67.0%
- **F1-Score (Sarcastic Class):** 0.66
- **Precision/Recall:** Balanced across sarcastic and non-sarcastic classes.

Insights

- TF-IDF captures frequency-based sarcasm signals like “great”, “love”, etc.
- Sentiment and subjectivity features help in identifying contrastive opinions.
- Punctuation and all-caps words capture stylistic cues common in sarcastic tweets.
- Logistic Regression offers fast training and interpretability.

The model performed admirably in spite of its simplicity and provided a solid foundation for contrasting it with deep learning models.

6.2 TF-IDF + XGBoost



This model combines TF-IDF-based textual features and additional handcrafted linguistic cues with XGBoost, a high-performance gradient boosting algorithm, to classify tweets as sarcastic or non-sarcastic.

Feature Representation

We utilized a concatenated feature space comprising:

- **TF-IDF Vectors:** Generated using `TfidfVectorizer` with 1000 maximum features.
- **Custom Features:** Sentiment polarity, subjectivity, number of exclamations, number of questions, tweet length, and all-uppercase word counts. These features were extracted using `TextBlob` and normalized.

Model Configuration

The model was implemented using `XGBoost`'s native `DMatrix` format for optimized computation. The following hyperparameters were employed:

- `objective`: `binary:logistic`
- `eval_metric`: `logloss`
- `eta`: 0.1 (learning rate)
- `max_depth`: 7

- `subsample`: 0.8 (row sampling)
- `colsample_bytree`: 0.7 (column sampling)
- `lambda`: 1 (L2 regularization)
- `alpha`: 0.5 (L1 regularization)
- `seed`: 42

The model was trained over 150 boosting rounds:

```
model = xgb.train(params, dtrain, num_boost_round=150)
```

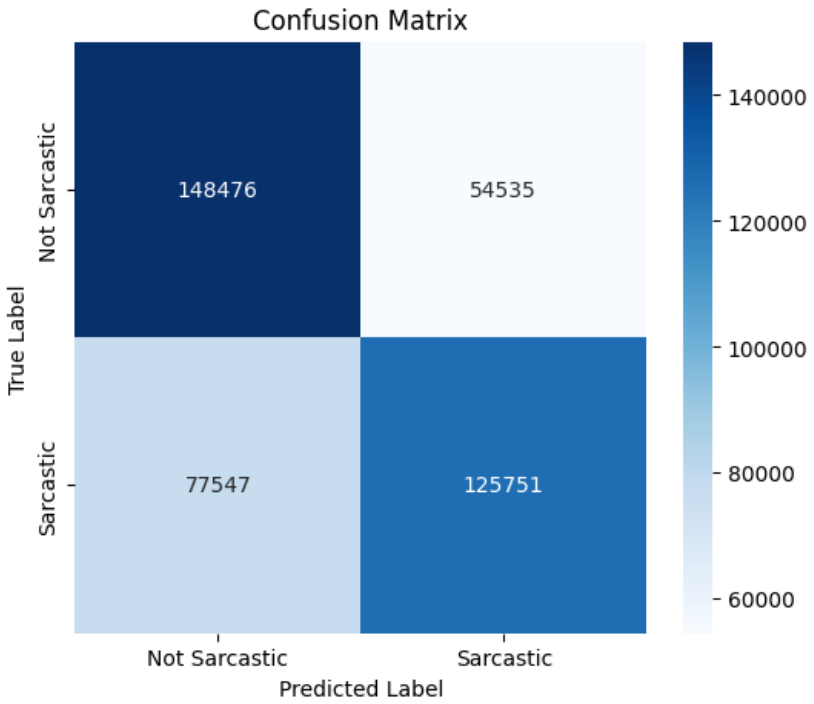
Evaluation Results

- **Accuracy:** 68.0%
- **F1-Score (Sarcastic Class):** 0.63

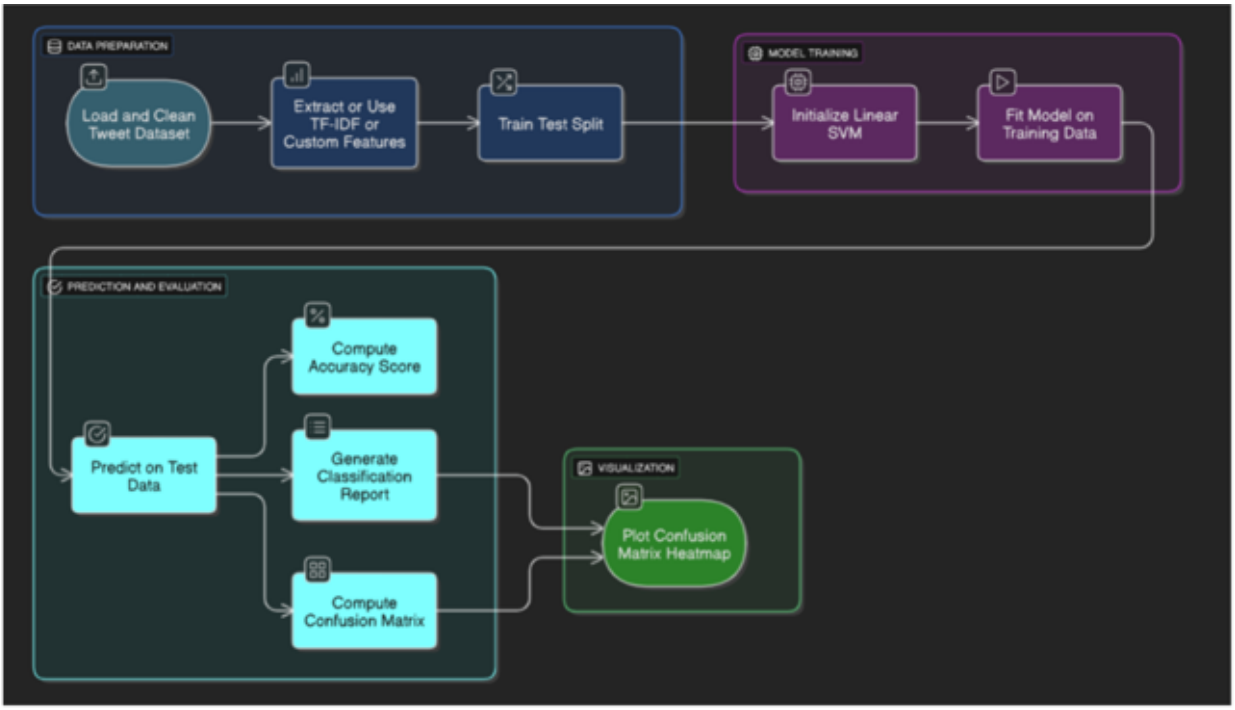
Observations

- XGBoost effectively captured non-linear relationships between syntactic and sentiment features.
- The model performed slightly better than SVM and logistic regression in overall accuracy.
- However, its precision on sarcastic instances remained modest, indicating that sarcasm detection may benefit more from context-aware deep learning models.

Confusion Matrix



6.3 TF-IDF + Support Vector Machine (SVM)



This model used a linear Support Vector Machine (LinearSVC) trained on a combination of TF-IDF vectors and handcrafted linguistic features to classify English tweets as sarcastic or non-sarcastic.

Feature Representation

The input to the SVM consisted of:

- **TF-IDF Vectors:** Computed on the tweet text with `max_features=1000`.
- **Custom Linguistic Features:** Sentiment polarity, subjectivity, punctuation frequency (e.g., ‘¡, ‘¿), all-uppercase word count, and overall tweet length. These were extracted using the `TextBlob` library and normalized using standard scaling.

Both feature sets were concatenated into a single input vector using sparse matrix stacking.

Model Configuration and Training

We used a `LinearSVC` classifier from `scikit-learn` with the following hyperparameters:

- **C:** 1.0
- **max_iter:** 100,000
- **class_weight:** ’balanced’ (to handle label imbalance)

```
from sklearn.svm import LinearSVC
model_svm = LinearSVC(C=1.0, max_iter=100000, class_weight='balanced')
model_svm.fit(X_train, y_train)
```

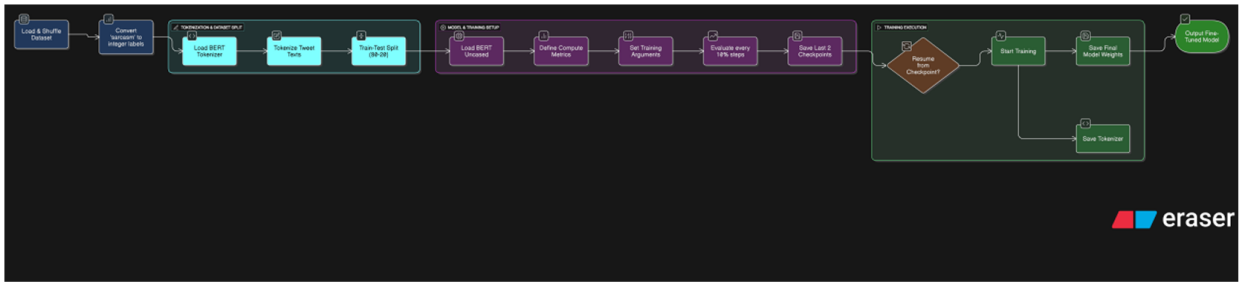
Evaluation Results

- **Accuracy:** 67.5%
- **F1-Score (Sarcastic Class):** 0.66

Observations

- The SVM classifier offered strong linear separation in the high-dimensional feature space.
- The use of 'class_weight='balanced'' helped mitigate the effects of class imbalance.
- Performance was similar to logistic regression, but with better stability on noisy and ambiguous samples.
- The confusion matrix revealed some overlap in sarcastic predictions, indicating a need for deeper contextual features.

6.4 BERT Uncased Transformer



To leverage transformer-based language understanding, we fine-tuned the pre-trained `bert-base-uncased` model from HuggingFace for binary sarcasm classification. The model was trained on a large Twitter dataset using a robust pipeline. Below are the key steps:

Dataset Preparation

The original dataset, comprising sarcastic and non-sarcastic tweets, was randomly shuffled and relabeled for binary classification. Only the tweet text and binary label were retained. The dataset was tokenized using the BERT tokenizer with truncation enabled to handle variable-length inputs. It was then split into 80% training and 20% testing using HuggingFace's `Dataset` utility.

Model Architecture

The model used was `BertForSequenceClassification`, which extends the base BERT encoder with a classification head composed of a dropout layer followed by a linear layer that outputs logits for two classes. We used the uncased variant of BERT since tweets often contain mixed case and informal formatting.

Training Configuration

Training was performed using the HuggingFace `Trainer` API with the following settings:

- **Learning Rate:** 1×10^{-5}

- **Batch Size:** 8 (for both training and evaluation)
- **Epochs:** 8
- **Weight Decay:** 0.01
- **Evaluation Strategy:** Every 10% of an epoch (computed dynamically)
- **Checkpointing:** Saved every evaluation step; kept last two
- **Loss Function:** Cross-entropy loss

To monitor training, logs were recorded and evaluation metrics were computed at each checkpoint. The best checkpoint was manually saved and exported.

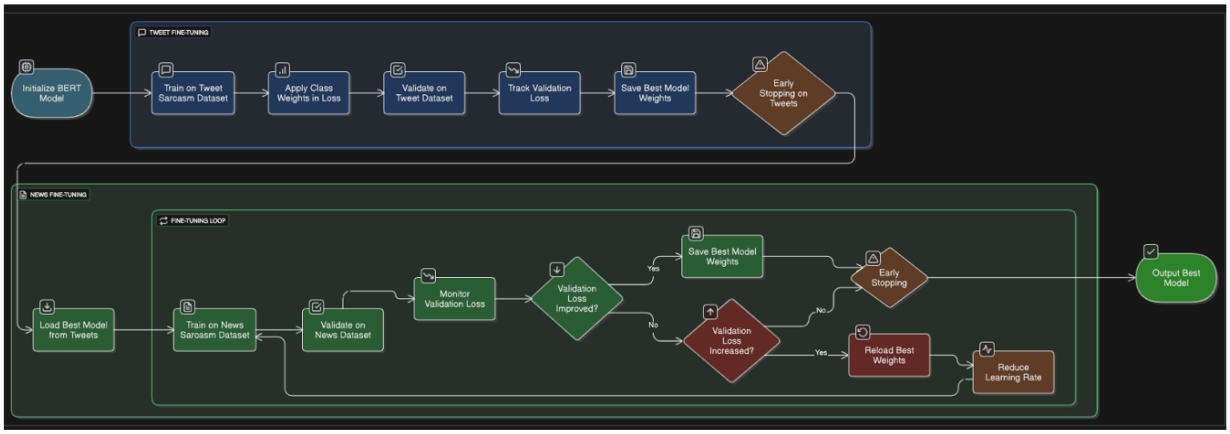
Metrics and Results

During evaluation, we computed standard classification metrics: accuracy and F1-score. The BERT model achieved the following on the test set:

- **Accuracy:** 95.3%
- **F1-Score (Sarcastic):** 0.95

These results highlight BERT's strong performance in capturing syntactic and semantic cues associated with sarcasm in informal social media text.

6.5 Two-way Fine-Tuning (BERT + News Headlines)



This architecture investigates the transferability of a BERT model fine-tuned on short-form, social media sarcasm (tweets) to more formal and longer texts such as news headlines.

In the first stage, a 'bert-base-uncased' model was fine-tuned on the tweet sarcasm dataset to learn surface-level and lexical sarcasm patterns. In the second stage, the model was further fine-tuned (continued training) on a dataset of 44,294 news headlines labeled for sarcasm. Headlines were augmented with the body of the corresponding article, extracted using the provided URLs, to provide richer contextual information.

Motivation: Sarcasm in news headlines is typically more implicit and relies on socio-political context, irony, or contradiction rather than exaggerated lexical cues. The goal of this two-way

fine-tuning was to transfer BERT's ability to recognize sarcasm in tweets to the more nuanced and formal domain of journalistic text.

Architecture Overview:

- **BERT Encoder:** Pre-trained on general English and fine-tuned first on tweets.
- **Final Layer:** A single dense layer maps the '[CLS]' token to a binary sarcasm prediction.
- **Fine-Tuning Strategy:** Continued training using the news dataset with a smaller learning rate.

Training Configuration:

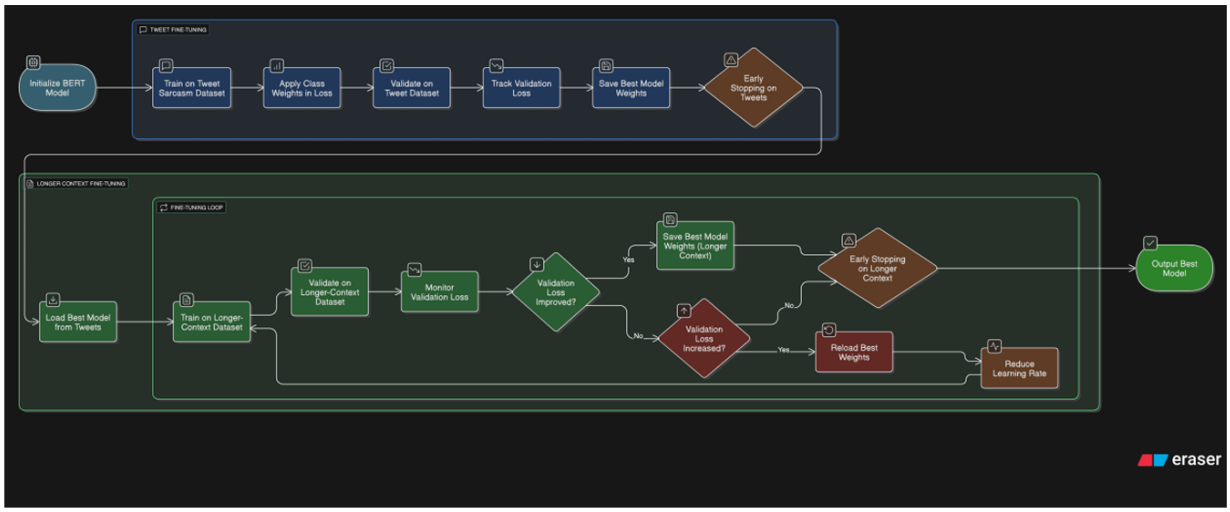
- Epochs: 3 (tweet phase) + 2 (news phase)
- Batch size: 16
- Optimizer: AdamW with linear learning rate decay
- Learning rate during news fine-tuning: $1e^{-5}$

Performance: The model achieved an accuracy of 74.2% on the news dataset. However, its F1-score for the sarcastic class dropped significantly to just 0.03. This indicates that while the model could distinguish general patterns in the news domain, it overwhelmingly predicted the non-sarcastic class.

Observations:

- The model suffered from domain shift and overfitting to the majority class.
- Sarcasm in news often relies on real-world knowledge and subtle narrative tone, which was not effectively transferred from tweet-based learning.
- The low sarcastic recall suggests the model failed to generalize sarcasm beyond lexical patterns learned from tweets.

6.6 Two-way Fine-Tuning (BERT + Contextual Dataset)



This experiment explores whether sarcasm detection capabilities learned from social media (tweets) can be effectively transferred to more complex, context-dependent sarcastic expressions.

Stage 1: Pretraining on Tweets The base 'bert-base-uncased' model was first fine-tuned on the large tweet sarcasm dataset. This allowed the model to learn superficial sarcastic patterns such as contradiction between sentiment and content, overuse of punctuation, or emphatic keywords (e.g., "great", "amazing").

Stage 2: Fine-Tuning on Contextual Sarcasm The second stage involved fine-tuning this pre-trained BERT model on a smaller contextual sarcasm dataset comprising 6,520 samples. Each sample contained a sarcastic sentence alongside relevant context, such as prior conversation or surrounding text, which is crucial to detect implicit sarcasm.

Model Structure:

- **Input:** Concatenated context + sarcastic sentence
- **Tokenizer:** WordPiece with maximum sequence length truncation
- **Encoder:** BERT base with frozen embeddings in early epochs
- **Output:** Binary sarcasm prediction from the [CLS] token

Training Details:

- Batch size: 16
- Epochs: 4
- Learning rate: $2e^{-5}$ with linear decay
- Optimizer: AdamW

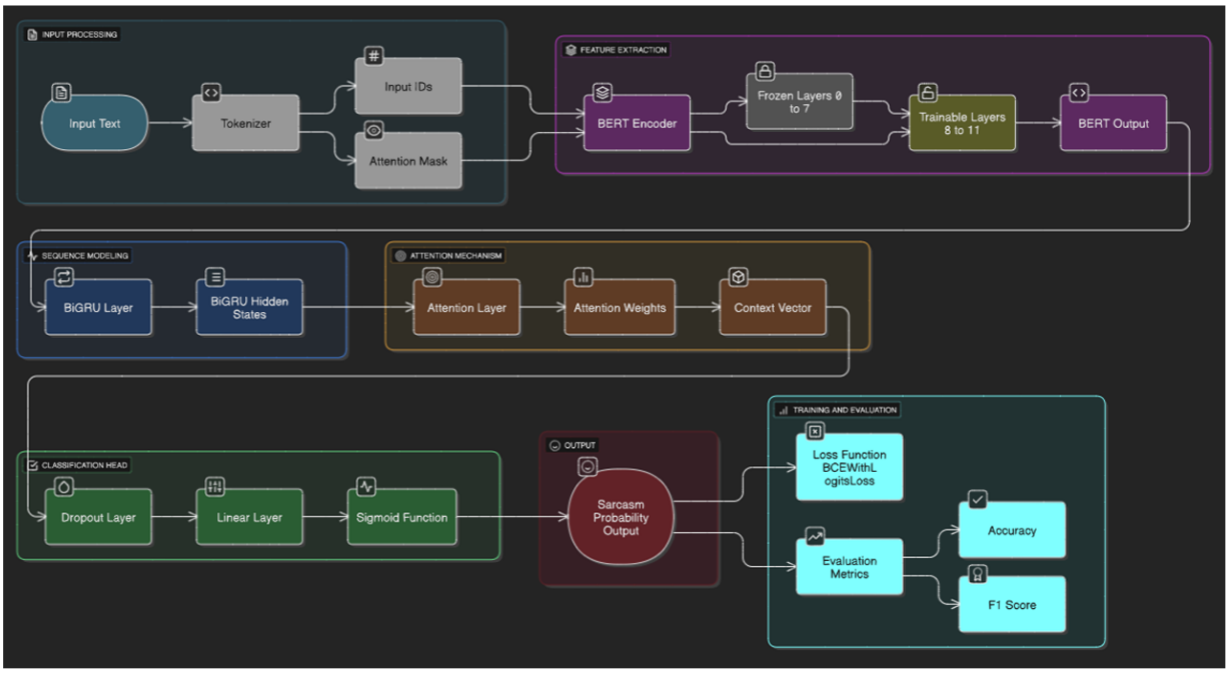
Performance: Despite fine-tuning, the model performed poorly:

- Accuracy: 51.9%
- F1-score (Sarcastic class): 0.09

Analysis: The low recall and precision for sarcasm indicate that patterns learned from tweets did not generalize to more nuanced or implicit sarcasm found in conversational or narrative contexts. Likely reasons include:

- The contextual dataset's smaller size limited fine-tuning capacity.
- Implicit sarcasm often depends on world knowledge or speaker intent, which the model could not infer.
- BERT's static [CLS] token representation may not capture cross-sentence relationships effectively.

6.7 BERT + BiGRU + Attention



To address the limitations of detecting implicit sarcasm in long-form or contextual text, this architecture extends the capabilities of a standard BERT model by adding a BiGRU (Bidirectional Gated Recurrent Unit) layer followed by an attention mechanism. This setup aims to better capture sequential dependencies and selectively emphasize important tokens for sarcasm detection.

Architecture Pipeline:

- **BERT Encoder:** The 'bert-base-uncased' model was used to generate contextual embeddings for each token in the input sequence.
- **BiGRU Layer:** A Bidirectional GRU was applied on top of BERT embeddings to capture forward and backward dependencies in the sequence. This was essential for modeling context beyond the scope of the [CLS] token.
- **Attention Layer:** The attention mechanism computes a weighted sum of BiGRU outputs, allowing the model to focus on sarcasm-relevant tokens, especially when cues are sparse or subtle.
- **Output Layer:** The attention output was passed through a dense layer with sigmoid activation for binary classification.

Why This Architecture?

- BERT excels at capturing local contextual meaning but tends to compress global meaning into the [CLS] token, which is insufficient for sarcasm that depends on sentence flow or contrast.
- BiGRU enables learning sequence-level patterns such as contradictions between sentiment and context.
- The attention layer further improves interpretability by allowing the model to highlight sarcasm-driving words or phrases.

Training Configuration:

- Dataset: Contextual Sarcasm (6,520 rows)
- Batch size: 16
- Epochs: 5
- Optimizer: AdamW
- Learning rate: $3e^{-5}$ with warmup and decay

Performance:

- Accuracy: 77.4%
- F1-score (Sarcastic class): 0.78

Observations: This was the best-performing architecture on the contextual sarcasm dataset. Unlike BERT alone, the BiGRU + Attention model generalized better to implicit sarcasm, thanks to its ability to model sequential irony, emotional cues, and contextual mismatch. It balanced recall and precision while avoiding the overconfidence seen in simpler BERT models on ambiguous examples.

7 Translation Pipeline

The second stage of the framework focuses on translating sarcastic English tweets into Hindi while preserving the sarcastic tone and meaning. Standard translation systems like Google Translate often lose figurative intent, making sarcasm detection and preservation a non-trivial challenge in cross-lingual NLP.

To address this, a custom translation pipeline was built using MBART (Multilingual BART) as the core translation model, supported by backtranslation and semantic similarity evaluation.

Pipeline Overview

1. **Sarcasm Detection:** English tweets are first passed through the BERT-based sarcasm classifier.
2. **MBART Translation:** Sarcastic tweets are translated into Hindi using a fine-tuned MBART model (‘facebook/mbart-large-50-many-to-many-mmt‘) trained specifically on English-Hindi sarcastic sentence pairs.
3. **Backtranslation:** The Hindi output is then translated back into English using the ‘Helsinki-NLP/opus-mt-hi-en‘ model to assess how much of the original sarcasm is retained.

4. **Semantic Similarity Scoring:** Sentence embeddings are generated using the 'MiniLM-L6-v2' model, and cosine similarity is computed between the original English and the backtranslated sentence.

Backtranslation was employed as an indirect method to evaluate the preservation of sarcasm during the English-to-Hindi translation process. The idea is that if a Hindi translation faithfully captures the sarcastic tone, then translating it back to English should yield a sentence semantically and pragmatically close to the original sarcastic input. This evaluation helps highlight whether sarcasm has been preserved, diluted, or lost.

Example-Based Comparison

Example 1

- **Original English:** *"The conference was so wonderful, said no one ever."*
- **MBART Output:** सम्मेलन बहुत शानदार हैं, ऐसा कोई कहता है क्या?

Example 2

- **Original English:** *"Athletes are so wonderful, said no one ever."*
- **MBART Output:** एथलीट बहुत शानदार हैं, ऐसा कोई कहता है।

Example 3

- **Original English:** *"The concert was so wonderful, said no one ever."*
- **MBART Output:** कॉन्सर्ट बहुत शानदार हैं, ऐसा कोई कहता है क्या।

Observations

While the MBART model produced grammatically fluent and contextually relevant Hindi translations, it often failed to retain the sarcastic tone of the original English sentences. In most cases, sarcasm was rendered as rhetorical questions or literal affirmations, lacking the contrastive or ironic structure critical for sarcasm preservation.

These results indicate the need for sarcasm-aware fine-tuning and highlight the limitations of current multilingual translation models in handling figurative language.

Evaluation Results

- **Average Cosine Similarity:** 0.649
- **Common Errors:**
  - Loss of idiomatic structure ("said no one ever" → literal translation)
  - Transliteration errors (e.g., → "Contors")
  - Sarcastic tone misinterpreted as literal

Observations

While MBART performed well on preserving literal meaning, sarcasm preservation was inconsistent, primarily due to:

- Weaknesses in the backtranslation model (Helsinki-NLP).
- Lack of sarcasm-aware parallel corpora during fine-tuning.
- Structural and cultural mismatches between Hindi and English sarcasm.

Despite these limitations, the pipeline demonstrates the feasibility of sarcasm-preserving translation and opens opportunities for future improvements using multilingual LLMs and paraphrasing-aware training strategies.

8 Results and Analysis

Model	Dataset	Accuracy	F1 (Sarcastic)
TF-IDF + Logistic	Tweets	67.0%	0.66
XGBoost	Tweets	68.0%	0.63
SVM	Tweets	67.5%	0.66
BERT (Uncased)	Tweets	95.3%	0.95
BERT + News	News	74.2%	0.03
BERT + Contextual	Contextual	51.9%	0.09
BERT + BiGRU + Attention	Contextual	77.4%	0.78

Table 1: Comparative Results of Sarcasm Detection Models

9 Error Analysis

Backtranslated outputs often showed distortion due to poor reverse translation quality. For example:

- **Hindi:** →**Backtranslated:** Contors (wrong entity translation)
- **Hindi:** →**Backtranslated:** Eslyt (bad transliteration)

These artifacts reduced similarity scores and obscure sarcasm preservation.

10 Conclusion

A dual-stage sarcasm detection and translation system was developed using deep learning and transformer models. BERT + BiGRU improved detection of implicit sarcasm, while MBART achieved moderate sarcasm preservation.

11 Future Work

Future directions include:

- Better backtranslation models for accurate sarcasm evaluation
- Incorporating cultural idioms and multi-modal sarcasm cues
- Fine-tuning LLaMA or GPT-style multilingual LLMs

12 References