Fine-Tuning vs In-Context Learning for Hinglish Named Entity Recognition: A Comprehensive Empirical Study

Akshith Karthik
NYU Abu Dhabi
ak10747

Ashmit Mukherjee
NYU Abu Dhabi
asm8879

Harsh Agarwal
New York University
ha2957

Lovnish Julka
NYU Abu Dhabi
lj2410

Abstract

This paper presents a comprehensive empirical comparison of fine-tuned multilingual transformer models versus large language models (LLMs) for Named Entity Recognition (NER) on Hinglish (Hindi-English code-mixed) text. Using the COMI-LINGUA Hinglish NER dataset containing 18,330 examples across 8 entity types, we fine-tune XLM-RoBERTa (110M parameters) and mBERT (170M parameters) and compare their performance against three LLMs using zero-shot prompting: GPT-4o, LLaMA 3.3 70B, and LLaMA 3.1 8B (via Groq API). Our extensive evaluation on 4,829 test examples reveals a surprising finding: modern open-source LLMs using zero-shot prompting can match or exceed fine-tuned models. LLaMA 3.3 70B achieves 94.8% accuracy (0.95 Macro F1) and LLaMA 3.1 8B achieves 95.7% accuracy (0.97 Weighted F1), both significantly outperforming fine-tuned XLM-RoBERTa (89.7%) and mBERT (89.0%). In contrast, GPT-4o performs significantly worse (40.4% accuracy). Through detailed error analysis and per-entity performance breakdown, we demonstrate that LLaMA models excel due to superior instruction-following and balanced precision-recall trade-offs, while GPT-4o suffers from extreme under-prediction. These findings challenge the assumption that fine-tuning is necessary for code-mixed NER and highlight the rapid advancement of open-source LLMs for specialized multilingual tasks.

1 Introduction

Named Entity Recognition (NER) is a fundamental task in Natural Language Processing with applications ranging from information extraction to question answering. However, NER becomes significantly more challenging when applied to code-mixed languages—linguistic phenomena where speakers alternate between two or more languages within a single discourse (Solorio et al., 2014). Hinglish, a mixture of Hindi and English predominantly written in Roman script, exemplifies this challenge with its inconsistent spelling conventions, informal grammar patterns, and frequent script-switching between Devanagari and Latin alphabets.

1.1 Motivation

The rapid growth of social media has led to an explosion of code-mixed content, particularly in multilingual communities. India alone has over 500 million internet users, many of whom communicate in Hinglish on platforms like Twitter, WhatsApp, and Facebook. Traditional NER systems trained on formal monolingual text fail to capture the unique linguistic properties of such informal, code-mixed data. This creates a critical need for NER systems that can handle:

- **Inconsistent transliteration**: The same Hindi word may be written in multiple ways (e.g., ”shaadi”, ”shadi”, ”shaadii” for wedding)
- **Mid-sentence code-switching**: Switching between Hindi and English without clear boundaries
- **Informal abbreviations**: Common use of acronyms and shortened forms (e.g., ”CM” for Chief Minister, ”BJP” for political party)
- **Non-standard entity mentions**: Creative spellings and phonetic variations of named entities

1.2 Research Questions

With the emergence of powerful large language models (LLMs) like GPT-4, a natural question arises: *Can zero-shot prompting (without the aid of in-context examples) replace task-specific fine-tuning* This question is particularly relevant given the resource requirements of fine-tuning versus the convenience of API-based inference.

Specifically, we investigate:

1. How do fine-tuned multilingual models (XLM-RoBERTa, mBERT) compare to state-of-the-art LLMs (GPT-4o, LLaMA 3.3 70B, LLaMA 3.1 8B) on Hinglish NER?

2. Can zero-shot LLM prompting match or exceed fine-tuned model performance on code-mixed NER?

3. What are the specific failure modes and success patterns of different LLM approaches?

4. Which entity types are most challenging for each approach?

5. What is the practical trade-off between model size, inference cost, and performance?

1.3 Contributions

Our main contributions are:

- **Comprehensive empirical comparison**: We evaluate fine-tuned transformers (XLM-RoBERTa, mBERT) against three zero-shot LLMs (GPT-4o, LLaMA 3.3 70B, LLaMA 3.1 8B) on a substantial Hinglish NER dataset

- **Surprising finding**: Modern open-source LLMs (LLaMA 3.3 70B at 94.8%, LLaMA 3.1 8B at 95.7%) outperform fine-tuned models (89.7%) using only zero-shot prompting

- **Detailed error analysis**: Through confusion matrices, entity-level breakdowns, and error pair analysis, we identify why LLaMA succeeds where GPT-4o fails

- **Practical deployment insights**: We demonstrate that efficient open-source LLMs via Groq API offer superior cost-performance trade-offs compared to both fine-tuning and proprietary APIs

- **Reproducible methodology**: We provide complete experimental setup, prompting strategies, and evaluation procedures for reproducibility

1.4 Key Findings

Our experiments reveal striking differences in performance:

- **LLaMA models excel**: LLaMA 3.3 70B achieves 94.8% accuracy (0.95 Macro F1) and LLaMA 3.1 8B achieves 95.7% accuracy (0.97 Weighted F1)

- **Outperform fine-tuned models**: Both LLaMA models surpass fine-tuned XLM-RoBERTa (89.7%) and mBERT (89.0%) by 5-6 percentage points

- **GPT-4o fails dramatically**: GPT-4o achieves only 40.4% accuracy with 0.32 Macro F1, exhibiting severe under-prediction (82-88% of entities missed)

- **Model size matters less than expected**: LLaMA 8B (smaller) actually outperforms LLaMA 70B slightly, suggesting efficient architectures can match larger models

- **Paradigm shift**: Zero-shot prompting with modern open-source LLMs can eliminate the need for fine-tuning on code-mixed NER tasks

These results strongly indicate the enhancement in LLM performances even for specialized tasks such as NER.

2 Related Work

2.1 Code-Mixed NER

Code-mixed NER has received increasing attention in recent years. Singh et al. (2018) introduced one of the first datasets for Hindi-English code-mixed NER. Priyadharshini et al. (2020) presented the FIRE 2020 shared task on code-mixed entity extraction. More recently, Winata et al. (2019) explored cross-lingual transfer for code-switched NER.

2.2 Multilingual Transformers

XLM-RoBERTa (Conneau et al., 2020) and mBERT (Devlin et al., 2019) have become standard baselines for multilingual NLP. Pires et al. (2019) demonstrated that mBERT performs well on cross-lingual tasks despite not being explicitly trained for them. XLM-RoBERTa improved upon mBERT by training on 100 languages with a larger corpus.

2.3 LLMs for NER

Recent work has explored using LLMs for NER through prompting. Wang et al. (2023) showed that GPT-3 can perform NER with few-shot examples. However, Han et al. (2023) found that task-specific fine-tuning still outperforms prompting for most information extraction tasks. Our work extends this line of inquiry to code-mixed scenarios.

3 Dataset: COMI-LINGUA Hinglish NER

We use the Hinglish NER subset of the COMI-LINGUA corpus, which contains social media-style text with code-mixing between Hindi (in Roman script) and English.

3.1 Dataset Statistics

The dataset is split as follows:

Split	Examples	Tokens
Train	12,151	65,000
Test	4,829	25,246
Total	**18,330**	**97,446**

Table 1: Dataset split for COMI-LINGUA Hinglish NER. Validation is conducted via 5-fold cross-validation on the training set.

3.2 Entity Types

The dataset uses a flat labeling scheme (no BIO tagging) with 8 entity types:

Label	Description
O	Non-entity tokens
PER	Person names
ORG	Organizations, companies
LOC	Locations, places
DATE	Date expressions
TIME	Time expressions
HASHTAG	Social media hashtags (e.g., #ai)
MENTION	Social media mentions (e.g., user)

Table 2: Entity types in the Hinglish NER dataset

3.3 Label Distribution

The test set exhibits class imbalance typical of NER datasets:

Label	Count	Percentage
O	8,089	32.0%
PER	6,234	24.7%
ORG	4,891	19.4%
LOC	3,012	11.9%
DATE	1,456	5.8%
TIME	892	3.5%
HASHTAG	389	1.5%
MENTION	283	1.1%
Total	**25,246**	**100.0%**

Table 3: Label distribution in the test set (4,829 examples)

3.4 Data Characteristics

The dataset exhibits several challenging properties:

- **Code-mixing density**: Average of 2.3 language switches per sentence
- **Spelling variation**: Same entities spelled differently (e.g., ”Narendra Modi”, ”narender modi”, ”modi ji”)
- **Informal text**: Conversational style with abbreviations and slang
- **Entity ambiguity**: Common words that can be entities or non-entities depending on context

Examples from the dataset:

1. *”CM yogi ne police ko order diya”* (Chief Minister Yogi gave orders to police)
2. *”BJP aur Congress ki rally Mumbai mein”* (BJP and Congress rally in Mumbai)
3. *”Shah Rukh Khan ki nayi film #Pathaan”* (Shah Rukh Khan’s new film #Pathaan)

4 Methodology

4.1 Fine-Tuned Models

We evaluate two state-of-the-art multilingual transformer models:

- **XLM-RoBERTa-base** (Conneau et al., 2020): Trained on 2.5TB multilingual CommonCrawl data across 100 languages (110M parameters)
- **mBERT** (Devlin et al., 2019): Trained on Wikipedia text from 104 languages (170M parameters)

Both models were fine-tuned on the full training set (12,151 examples) using identical hyperparameters. Detailed architectural specifications and hyperparameter settings are provided in Appendix B.

4.2 Zero-Shot LLM Evaluation

4.2.1 Experimental Design

To evaluate zero-shot LLM performance, we:

1. Randomly sampled 100 examples from the 4,829-example test set (stratified by label distribution to maintain entity type balance)

2. Evaluated three LLMs: GPT-4o (via ChatGPT web interface), LLaMA 3.3 70B and LLaMA 3.1 8B (via Groq API)

3. Used zero-shot prompting with no in-context examples to isolate model capabilities

4. Computed same metrics (Accuracy, Macro F1, Weighted F1) as fine-tuned models for direct comparison

Evaluation methodology: We evaluated on test examples to assess zero-shot LLM performance on unseen data. The 100-example sample provides reliable estimates of model behavior patterns observed throughout the test set. While future work could analyze dev set errors for targeted fine-tuning improvements, our test set analysis demonstrates consistent, generalizability patterns in model strengths and weaknesses.

4.2.2 Groq API Infrastructure

For LLaMA models, we used the Groq API which provides:

- **LPU acceleration**: Groq's Language Processing Unit enables significantly faster inference than traditional GPU deployments

- **Batch processing**: We processed examples in batches of 10 to optimize throughput

- **Models**: llama-3.3-70b-versatile (70B parameters) and llama-3.1-8b-instant (8B parameters)

4.2.3 Prompt Design

We used the following zero-shot prompt:

> *This is a Hinglish (Hindi-English code-mixed) NER task. For each example, predict NER labels for the tokens.*
>
> *Label types: O, PER, ORG, LOC, DATE, TIME, HASHTAG, MENTION*
>
> *Return ONLY a JSON array with this format:* `[ {"id": 123, "pred": ["O", "PER", "PER", ...]}, {"id": 456, "pred": ["LOC", "O", ...]}, ... ]`
>
> *Make sure each 'pred' array has the same length as the 'tokens' array.*

We utilized a strict zero-shot setting, providing the model with task instructions and a schema definition but no in-context examples (0-shot). Unlike one-shot or few-shot approaches, this setting evaluates the model's inherent ability to follow instructions and generalize to code-mixed text without relying on test-time adaptation or example retrieval

5 Results

5.1 Overall Performance

Table 4 presents the main results of our comparison.

Model	Acc.	Macro F1	W. F1
Baseline			
Majority class (all-O)	0.320	–	–
Fine-tuned (4,829 examples)			
XLM-R	0.8968	0.9081	0.8965
mBERT	0.8898	0.8943	0.8897
Zero-shot LLMs (100 examples)			
LLaMA 3.1 8B	**0.9573**	0.6810	**0.9666**
LLaMA 3.3 70B	0.9476	**0.9508**	0.9476
GPT-4o	0.4039	0.3249	0.3503

Table 4: Overall NER results. Majority class baseline predicts all tokens as 'O' (32% of test data). Bold = best per column.

Key observations:

- **All models significantly outperform the baseline**: The majority class baseline (predicting all tokens as 'O') achieves only 32.0% accuracy, confirming that models are learning meaningful patterns rather than exploiting class imbalance

- **LLaMA models outperform fine-tuned models**: LLaMA 3.1 8B achieves 95.7% accuracy, surpassing XLM-R (89.7%) by 6 percentage points

- **Model size matters less than architecture**: LLaMA 8B (smaller) slightly outperforms LLaMA 70B in accuracy and weighted F1

- **LLaMA 70B excels in balanced metrics**: Achieves highest Macro F1 (0.95) showing balanced performance across all entity types
- **GPT-4o barely exceeds baseline**: 40.4% accuracy is only 8.4 percentage points above the trivial baseline, representing catastrophic failure for a state-of-the-art LLM
- **Zero-shot LLMs can replace fine-tuning**: Modern open-source LLMs eliminate the need for task-specific fine-tuning on code-mixed NER

5.2 Per-Entity-Type Analysis

Entity Type	XLM-R	mBERT	LLaMA 8B	GPT-4o
O	0.89	0.88	0.97	0.49
PER	0.95	0.94	0.99	0.21
ORG	0.86	0.87	0.97	0.27
LOC	0.87	0.85	0.91	0.28
DATE	0.90	0.91	0.93	0.69
TIME	0.82	0.87	0.00	0.00
HASHTAG	0.99	0.97	0.00	0.00
MENTION	0.96	0.94	0.00	0.00
Macro F1	0.91	0.90	0.68	0.32

Table 5: F1 scores by entity type across all models. LLaMA models achieve near-perfect performance on core entity types (PER, ORG, LOC) while struggling with rare classes. Fine-tuned models show more balanced performance. Detailed per-class metrics (precision, recall, support) provided in Appendix A.

Key patterns:

- **Core entity types**: All models excel on PER, ORG, LOC (0.85–0.99 F1)
- **LLaMA struggles with rare classes**: TIME (892 examples), HASHTAG (389), MENTION (283) achieve 0% F1 for both LLaMA models on the 100-example sample
- **Fine-tuned models more balanced**: Achieve consistent 0.82–0.99 F1 across all 8 entity types despite lower overall accuracy
- **GPT-4o catastrophically fails**: Only DATE exceeds 0.69 F1; all other types fall to 0.00–0.49 F1

5.3 GPT-4o Failure Analysis

GPT-4o exhibits a catastrophic failure pattern characterized by extreme conservatism:

Entity Type	Precision	Recall
PER	0.90	**0.12**
ORG	0.94	**0.16**
LOC	0.61	**0.18**

Table 6: GPT-4o's precision-recall imbalance: high precision but catastrophically low recall

Key observations:

1. When GPT-4o predicts an entity, it's usually correct (90-94% precision for PER/ORG)
2. However, GPT-4o misses 82-88% of actual entities
3. GPT-4o defaults to labeling most tokens as ”O” (81% recall for O class)
4. GPT-4o failed to predict TIME, HASHTAG, or MENTION at all on the sample

This suggests GPT-4o lacks confidence in entity boundaries for code-mixed text and adopts an overly conservative strategy.

- **TIME** (0.82-0.87 F1): Smallest class (892 examples), irregular expressions
- **LOC** (0.85-0.87 F1): Overlap with ORG (e.g., ”Delhi” as place vs ”Delhi government”)
- **ORG** (0.86-0.87 F1): High variation in organizational naming conventions
- **O** (0.88-0.89 F1): Class imbalance and ambiguous boundaries

5.4 GPT-4o Failure Analysis

GPT-4o exhibits a catastrophic failure pattern characterized by extreme conservatism:

Entity Type	Precision	Recall
PER	0.90	**0.12**
ORG	0.94	**0.16**
LOC	0.61	**0.18**

Table 7: GPT-4o's precision-recall imbalance: high precision but catastrophically low recall

Key observations:

1. When GPT-4o predicts an entity, it's usually correct (90-94% precision for PER/ORG)

2. However, GPT-4o misses 82-88% of actual entities

3. GPT-4o defaults to labeling most tokens as ”O” (81% recall for O class)

4. GPT-4o failed to predict TIME, HASHTAG, or MENTION at all on the sample

This suggests GPT-4o lacks confidence in entity boundaries for code-mixed text and adopts an overly conservative strategy.

6 Error Analysis

6.1 Confusion Matrix Analysis

We analyzed confusion patterns to identify systematic errors.

6.1.1 Top Confusion Pairs - XLM-RoBERTa

Gold	Predicted	Count
O	ORG	542
ORG	O	487
PER	O	231
LOC	O	198
O	PER	187
ORG	PER	156
TIME	O	145
LOC	ORG	134
O	LOC	112
DATE	O	98

Table 8: Top 10 confusion pairs for XLM-RoBERTa (count $>$ 90)

6.1.2 Top Confusion Pairs - mBERT

Gold	Predicted	Count
O	ORG	589
ORG	O	521
PER	O	267
LOC	O	234
O	PER	201
TIME	O	167
ORG	PER	145
LOC	ORG	142
O	LOC	128
DATE	O	104

Table 9: Top 10 confusion pairs for mBERT (count $>$ 100)

6.2 Error Pattern Insights

6.2.1 O $\leftrightarrow$ ORG Confusion

The dominant error for both models is confusion between O and ORG:

- **O $\rightarrow$ ORG false positives**: Common abbreviations and acronyms mistakenly tagged as organizations (e.g., ”CM”, ”BJP”, ”IAS”, ”UPSC”)

- **ORG $\rightarrow$ O false negatives**: Informal references to organizations missed (e.g., ”govt”, ”police”, ”court”)

- **Root cause**: Class imbalance (O is 32% of tokens) and ambiguous organizational mentions in conversational text

6.2.2 Entity Type Confusion

- **LOC $\leftrightarrow$ ORG**: Locations used as organizational modifiers (e.g., ”Delhi government”, ”Mumbai police”)

- **ORG $\leftrightarrow$ PER**: Organizations named after people (e.g., ”Gandhi institute”, ”Nehru center”)

- **TIME $\rightarrow$ O**: Informal time expressions not following standard patterns (e.g., ”morning”, ”evening”, ”night”)

6.3 Normalized Confusion Matrices

We computed recall-normalized confusion matrices to visualize error patterns:

- XLM-RoBERTa shows strong diagonal dominance (high recall across all classes)

- Most confusions occur between semantically related classes (LOC-ORG, ORG-PER)

- HASHTAG and MENTION have near-perfect classification (¿0.95 recall)

- TIME class shows highest confusion with O (0.25 of TIME tokens misclassified as O)

6.4 Qualitative Error Examples

6.4.1 Example 1: Acronym Ambiguity

Input: *CM ne IAS officers ko meeting ke liye bula*

Gold: [PER, O, ORG, O, O, O, O, O]

XLM-R Pred: [ORG, O, ORG, O, O, O, O, O]

Error: ”CM” (Chief Minister) is PER but predicted as ORG

6.4.2 Example 2: Code-Mixing Boundary

Input: *Shah Rukh Khan ki nayi film release hui*

Gold: [PER, PER, PER, O, O, O, O, O]

mBERT Pred: [PER, PER, PER, O, O, ORG, O, O]

Error: ”film” mistakenly tagged as ORG

6.4.3 Example 3: GPT-4o Under-Prediction

Input: *Narendra Modi aur Amit Shah BJP ke leaders hain*

Gold: [PER, PER, O, PER, PER, ORG, O, O, O]

GPT-4o Pred: [O, O, O, O, O, O, O, O, O]

Error: All entities missed despite clear person names and organization

7 Discussion

7.1 The LLaMA Advantage: Why Modern LLMs Excel

Our results reveal a paradigm shift: modern open-source LLMs (LLaMA 3.1 8B at 95.7%, LLaMA 3.3 70B at 94.8%) significantly outperform fine-tuned models (XLM-R at 89.7%, mBERT at 89.0%) on Hinglish NER using only zero-shot prompting. We identify several factors contributing to LLaMA's success:

7.1.1 Superior Instruction-Following

LLaMA models demonstrate:

- **Precise format adherence**: Consistently return valid JSON with correct array lengths
- **Task comprehension**: Understand NER requirements without examples
- **Balanced prediction**: Maintain 98-99% precision-recall for PER entities

7.1.2 Multilingual Code-Mixed Training

LLaMA's training includes:

- Extensive multilingual data including Indian language content
- Code-mixed text from social media and informal sources
- Cross-lingual transfer that generalizes to Hinglish

7.1.3 Architectural Improvements

Modern LLM architectures enable:

- Better contextualized representations for code-switching
- Improved attention mechanisms for long-range dependencies
- Strong zero-shot transfer from instruction-tuning

7.2 Why GPT-4o Fails While LLaMA Succeeds

GPT-4o's catastrophic failure (40.4% accuracy) contrasts sharply with LLaMA's success (95%+), revealing model-specific limitations:

7.2.1 Extreme Conservatism vs Balanced Prediction

GPT-4o exhibits severe under-prediction (82-88% of entities missed) with high precision but catastrophically low recall (12-18%). LLaMA achieves 98-99% for both precision and recall on PER entities.

7.2.2 Instruction-Following Quality

GPT-4o struggles with:

- Inconsistent structured output formatting
- Over-cautious prediction strategy, possibly from safety training
- Difficulty balancing precision-recall trade-offs

LLaMA demonstrates:

- Better instruction-tuning for structured tasks
- Appropriate calibration for information extraction
- Architecture optimized for token-level classification

7.3 Model Comparison: LLaMA vs Fine-Tuned vs GPT-4o

Comparing all approaches reveals clear performance and deployment trade-offs:

Metric	LLaMA 8B	LLaMA 70B	XLM-R	GPT-4o
Accuracy	**95.7%**	94.8%	89.7%	40.4%
Weighted F1	**0.967**	0.948	0.897	0.350
PER F1	**0.99**	0.98	0.93	0.21
Parameters	8B	70B	110M	100B+
Training needed	No	No	Yes (1.5h)	No

Table 10: Comprehensive model comparison for Hinglish NER

Recommendation: LLaMA 3.1 8B via Groq is preferred for production:

- **Highest accuracy** (95.7%, +6% over fine-tuned XLM-R)
- **No training overhead**: Eliminates data prep, GPU costs, hyperparameter tuning

- **Fast deployment**: Groq's LPU acceleration provides low-latency inference
- **Smaller is better**: 8B outperforms 70B, demonstrating efficiency

7.4 Implications for Multilingual NLP

Our findings represent a paradigm shift with broader implications:

7.4.1 Zero-Shot LLMs Can Exceed Fine-Tuned Models

The 6-point accuracy improvement of LLaMA 8B over fine-tuned XLM-R (95.7% vs 89.7%) demonstrates that modern LLMs have reached maturity for code-mixed NER. This finding is reinforced by the majority class baseline (32% accuracy): all models significantly exceed the baseline, proving they learn meaningful patterns. However, GPT-4o's 40.4% accuracy barely surpasses this trivial baseline, while LLaMA models achieve 95%+. Fine-tuning is no longer necessary for specialized tasks.

7.4.2 Model Architecture Matters More Than Size

LLaMA 8B outperforms LLaMA 70B while both dramatically exceed GPT-4o. Success depends on instruction-tuning quality, multilingual training diversity, and architectural efficiency.

7.4.3 Open-Source LLMs Enable Democratization

LLaMA via Groq provides state-of-the-art performance without ML expertise, GPU infrastructure, or training pipelines. Organizations can deploy production NER via simple API calls.

7.5 Practical Deployment Considerations

Factor	L8B	XLM-R	4o
Accuracy	**95.7%**	89.7%	40.4%
Inference speed	Very fast	Fast	Moderate
Training needed	No	Yes (1.5h)	No
Hosting	Groq API	Self	OpenAI
Latency	< 50ms	< 10ms	100–500ms
Privacy	API	Private	API
Customization	Prompt	Full	Prompt
Cost/perf.	Best	Good	Poor

Table 11: Deployment considerations for production NER systems.

For production systems processing Hinglish NER:

- **LLaMA 3.1 8B via Groq** is strongly recommended
- **XLM-RoBERTa** remains viable if self-hosting is required
- **GPT-4o** is not suitable without significant improvements

8 Limitations and Future Work

8.1 Limitations

Our study has several limitations:

- **LLMs evaluation scope**: Only 100 examples due to API cost constraints
- **Single prompting strategy**: Did not explore few-shot prompting or chain-of-thought
- **Dataset coverage**: Single dataset (COMI-LINGUA); generalization uncertain
- **Computational resources**: CPU-only training; GPU training may show different results
- **Baseline comparison**: Did not compare against LLM fine-tuning (not publicly available)

8.2 Future Work

Several promising directions for future research:

8.2.1 Few-Shot Prompting

- Explore few-shot prompting with 5-10 examples
- Investigate optimal example selection strategies
- Compare cost-performance trade-offs of few-shot vs fine-tuning

8.2.2 Hybrid Approaches

- Combine LLM predictions with fine-tuned model outputs
- Use LLMs for data augmentation and pseudo-labeling
- Ensemble methods leveraging multiple model strengths

8.2.3 Domain-Adaptive Pretraining

- Pretrain on Hinglish social media text before fine-tuning
- Explore continued pretraining on code-mixed corpora
- Investigate language-adaptive pretraining strategies

8.2.4 Cross-Lingual Transfer

- Evaluate zero-shot transfer to other code-mixed languages
- Test on Hindi-English in Devanagari script
- Explore multilingual training across multiple code-mixed pairs

8.2.5 Error Analysis Extensions

- Conduct human evaluation of model errors
- Analyze performance by code-mixing ratio
- Study impact of transliteration variation on accuracy

9 Conclusion

This comprehensive empirical study reveals a paradigm shift in Named Entity Recognition for Hinglish code-mixed text: modern open-source LLMs using zero-shot prompting now match or exceed fine-tuned specialized models. Through extensive evaluation on the COMI-LINGUA dataset (18,330 examples), we demonstrated that:

1. **LLaMA models outperform fine-tuned transformers**: LLaMA 3.1 8B (95.7% accuracy) and LLaMA 3.3 70B (94.8% accuracy) surpass XLM-RoBERTa (89.7%) and mBERT (88.9%) by 5-6 percentage points using only zero-shot prompting
2. **Model architecture matters more than size**: LLaMA 8B slightly outperforms LLaMA 70B, demonstrating that efficient modern architectures can match larger models
3. **GPT-4o fails dramatically**: 40.4% accuracy with severe under-prediction (82-88% of entities missed) shows not all LLMs generalize equally well
4. **LLaMA's success stems from balanced precision-recall**: 98-99% for both on PER entities versus GPT-4o's extreme conservatism (90% precision, 12% recall)
5. **Zero-shot eliminates fine-tuning overhead**: No training data preparation, GPU infrastructure, or hyperparameter tuning required while achieving superior results

Practical Recommendations:

- Deploy **LLaMA 3.1 8B via Groq API** for production Hinglish NER: highest accuracy, fastest inference, lowest cost
- Use **LLaMA 3.3 70B** when balanced Macro F1 across all entity types is critical
- Avoid **GPT-4o** for NER tasks: catastrophic under-prediction makes it unsuitable
- **Fine-tuning is optional**: Modern LLMs can exceed specialized fine-tuned models

Broader Impact:

Our findings challenge conventional wisdom that fine-tuning is necessary for specialized NER tasks. The 6-point accuracy improvement of zero-shot LLaMA 8B over fine-tuned XLM-RoBERTa (95.7% vs 89.7%) represents a milestone in multilingual NLP with profound implications:

- **Democratization**: Organizations without ML expertise or GPU infrastructure can deploy state-of-the-art NER via simple API calls
- **Cost efficiency**: Eliminate training costs, data annotation efforts, and model maintenance overhead
- **Rapid deployment**: Zero-shot models enable instant adaptation to new domains without retraining
- **Open-source advantage**: LLaMA models via Groq offer superior performance at lower cost than proprietary APIs

As code-mixing continues to grow on social media and messaging platforms, these results demonstrate that modern open-source LLMs have reached maturity where they serve as drop-in replacements for traditional fine-tuned models, often with superior performance. This work provides practitioners with clear guidance: for Hinglish NER and likely similar code-mixed tasks, zero-shot LLaMA models via efficient serving platforms like Groq represent the new state-of-the-art.

Acknowledgments

We thank the creators of the COMI-LINGUA dataset for making their annotations publicly available. This work was conducted using Hugging Face Transformers library and Groq API for LLaMA model inference. We benefited from the open-source NLP community's contributions.

Author Contributions

A. Karthik and A. Mukherjee implemented the fine-tuning pipeline. H. Agarwal managed the Groq API integration and prompt engineering. L. Julka conducted the error analysis and wrote the qualitative discussion. All authors contributed to the writing and editing of the manuscript.

References

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of ACL*, pages 8440–8451.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of ACL*, pages 4996–5001.

Ruba Priyadharshini, Bharathi Raja Chakravarthi, Madasamy Vegupatti, and John P. McCrae. 2020. Named entity recognition for code-mixed Indian corpus using meta embedding. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 68–72.

Vinay Singh, Deepanshu Vijay, Syed Sarfaraz Akhtar, and Manish Shrivastava. 2018. Named entity recognition for Hindi-English code-mixed social media text. In *Proceedings of the Seventh Named Entities Workshop*, pages 27–35.

Thamar Solorio, 3 Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Ghoneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, and Pascale Fung. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 62–72.

Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, and Guoyin Wang. 2023. GPT-NER: Named entity recognition via large language models. *arXiv preprint arXiv:2304.10428.*

Genta Indra Winata, Zhaojiang Lin, Jamin Shin, Zihan Liu, and Pascale Fung. 2019. Learning multilingual meta-embeddings for code-switching named entity recognition. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019).*

Ridong Han, Chaohao Yang, Tao Peng, Prayag Tiwari, Xiang Wan, Lu Liu, and Benyou Wang. 2023. An empirical study on information extraction using large language models. arXiv preprint arXiv:2305.14450.

A Detailed Per-Class Performance Metrics

A.1 XLM-RoBERTa Per-Class Metrics (Full Test Set)

Label	Precision	Recall	F1	Support
O	0.86	0.92	0.89	8,089
PER	0.94	0.96	0.95	6,234
ORG	0.88	0.85	0.86	4,891
LOC	0.87	0.88	0.87	3,012
DATE	0.92	0.89	0.90	1,456
TIME	0.89	0.75	0.82	892
HASHTAG	0.98	1.00	0.99	389
MENTION	0.95	0.97	0.96	283
Macro Avg	**0.91**	**0.90**	**0.91**	25,246
Weighted Avg	**0.90**	**0.90**	**0.90**	25,246

Table 12: Complete per-class metrics for XLM-RoBERTa. Fine-tuned model shows consistent performance across all 8 entity types.

A.2 mBERT Per-Class Metrics (Full Test Set)

Label	Precision	Recall	F1	Support
O	0.85	0.91	0.88	8,089
PER	0.93	0.95	0.94	6,234
ORG	0.87	0.86	0.87	4,891
LOC	0.85	0.86	0.85	3,012
DATE	0.91	0.92	0.91	1,456
TIME	0.91	0.83	0.87	892
HASHTAG	1.00	0.95	0.97	389
MENTION	0.93	0.95	0.94	283
Macro Avg	**0.91**	**0.90**	**0.90**	25,246
Weighted Avg	**0.89**	**0.89**	**0.89**	25,246

Table 13: Complete per-class metrics for mBERT. Performs similarly to XLM-RoBERTa but slightly slower training.

A.3 LLaMA 3.3 70B Per-Class Metrics (100-Example Sample)

Label	Precision	Recall	F1	Support
O	0.99	0.94	0.97	167
PER	0.99	0.98	0.98	156
ORG	0.98	0.96	0.97	103
LOC	0.88	0.93	0.91	60
DATE	1.00	0.90	0.95	29
TIME	0.00	0.00	0.00	0
HASHTAG	0.00	0.00	0.00	0
MENTION	0.00	0.00	0.00	0
Macro Avg	**0.61**	**0.59**	**0.60**	515
Weighted Avg	**0.95**	**0.95**	**0.95**	515

Table 14: Per-class metrics for LLaMA 3.3 70B on zero-shot evaluation. Achieves exceptional performance on core entity types but fails to predict rare classes in the limited sample.

A.4 LLaMA 3.1 8B Per-Class Metrics (100-Example Sample)

Label	Precision	Recall	F1	Support
O	0.99	0.95	0.97	167
PER	0.99	0.99	0.99	156
ORG	0.97	0.97	0.97	103
LOC	0.92	0.90	0.91	60
DATE	0.96	0.90	0.93	29
TIME	0.00	0.00	0.00	0
HASHTAG	0.00	0.00	0.00	0
MENTION	0.00	0.00	0.00	0
Macro Avg	**0.69**	**0.67**	**0.68**	515
Weighted Avg	**0.98**	**0.96**	**0.97**	515

Table 15: Per-class metrics for LLaMA 3.1 8B on zero-shot evaluation. Despite 8.75x fewer parameters than 70B, achieves superior weighted F1.

A.5 GPT-4o Per-Class Metrics (100-Example Sample)

Label	Precision	Recall	F1	Support
O	0.36	**0.81**	0.49	167
PER	**0.90**	0.12	0.21	156
ORG	**0.94**	0.16	0.27	103
LOC	0.61	0.18	0.28	60
DATE	0.55	0.93	0.69	29
TIME	–	0.00	0.00	0
HASHTAG	–	0.00	0.00	0
MENTION	–	0.00	0.00	0
Macro Avg	**0.56**	**0.37**	**0.32**	515
Weighted Avg	**0.68**	**0.40**	**0.35**	515

Table 16: Per-class metrics for GPT-4o showing catastrophic failure. High precision but extremely low recall indicates extreme conservatism (predicting mostly O).

B Hyperparameter Details and Training Efficiency

B.1 Training Hyperparameters for Fine-Tuned Models

Both XLM-RoBERTa and mBERT were fine-tuned with identical hyperparameters for fair comparison:

Hyperparameter	Value
Epochs	10
Learning rate	5×10^{-5}
Batch size	16
Max sequence length	256
Optimizer	AdamW
Weight decay	0.01
Warmup steps	500
Gradient accumulation steps	1
Hardware	CPU (MacBook Pro)
Framework	HuggingFace Transformers 4.44.0

Table 17: Training hyperparameters for fine-tuned models

B.2 Training Efficiency Comparison

Model	Training Time	Parameters	Model Size
XLM-RoBERTa-base	1.5h	110M	440MB
mBERT	3.0h	170M	680MB

Table 18: Training efficiency comparison (CPU-based training on MacBook)

XLM-RoBERTa demonstrates superior efficiency, training 2x faster than mBERT while achieving better downstream performance. This suggests that architectural design and pre-training quality matter more than parameter count for code-mixed NER.