PECAN – Programming Encoder Classification Analysis Network

Motivation

Programming language identification (PL-ID) is foundational for static analysis, indexing, developer tooling, and security triage. In real-world settings, however, PL-ID often operates under incomplete context with missing metadata and short code fragments.

Three challenges we focus on:

Missing metadata	Partial snippets	Non-repo contexts
• File extension/path unavailable or unreliable • Example: snippet with no .py	• Clipped code, logs, chat fragments • Example: function body without imports/context	• Analysis of code on unstructured data sources • Example: StackOverflow, logs, memory dumps

Contributions

- Unified benchmarking framework for metadata-free PL-ID across practical tools, encoder models, and LLM baselines.
- Reporting focus: fully fine-tuned **CodeBERT** [1] as the primary encoder baseline; other encoder families are contextual references.
- Reliability-aware LLM evaluation protocol (structured outputs + abstention/coverage metrics).
- Transparent reporting via both canonical (alias-aware) and strict raw-label metrics.
- Reproducible deterministic split artifacts and shared evaluation pipeline across all models.

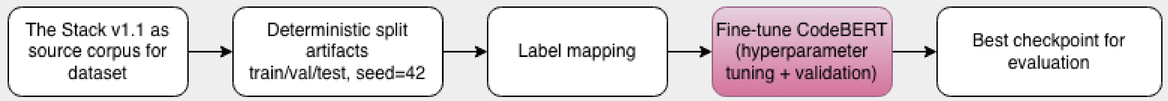
Experimental Setup

- Dataset Origin: The Stack v1.1
- **42.1M samples • 319 languages**
- Fixed-length byte-level snippets
- **68 / 12 / 20 train / val / test split (seed = 42)**

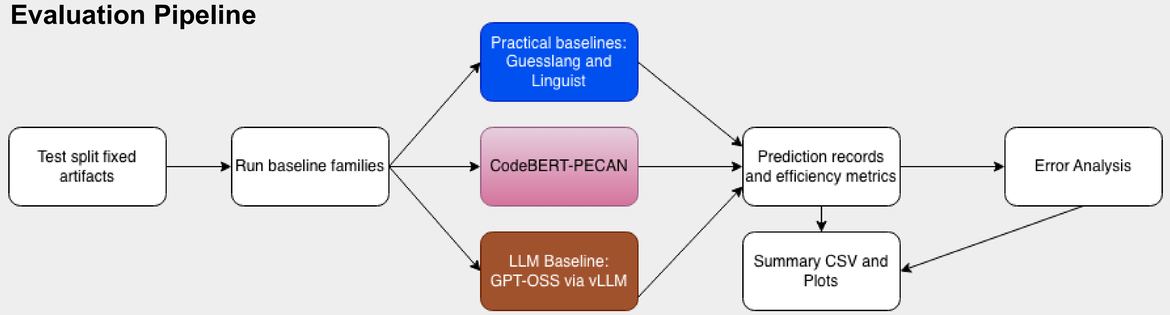
Baselines	Guesslang [2] • GitHub Linguist [3] • GPT-OSS via vLLM [4][5]
Metrics	Accuracy • F1 • Coverage • Latency
Errors	Confusion pairs • Failure breakdown

Methodology

Training Pipeline



Evaluation Pipeline



- **Unified evaluation** across practical tools, encoder models, and LLMs
- **Deterministic dataset splits** with fixed-length snippets
- **Dual evaluation**: canonical (alias-aware) vs raw label metrics
- **Reliability-aware evaluation**: coverage (answered rate) + abstention handling
- **Post-hoc error analysis**: confusions, per-language errors, and failure-setting breakdowns from saved prediction logs
- **CodeBERT training**: sweep-tuned hyperparameters with validation-based best-checkpoint selection

Results

Effectiveness

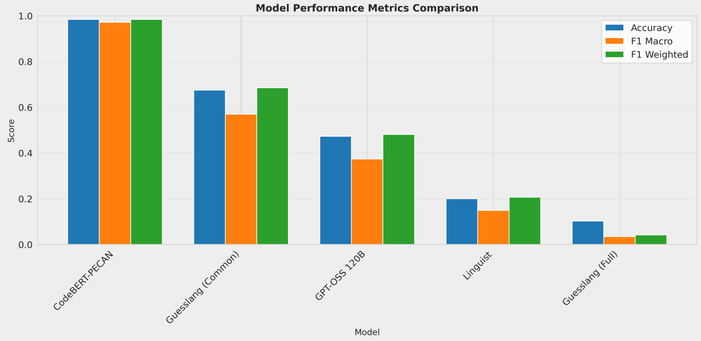


Figure 1 Model Performance Metrics Comparison
CodeBERT-PECAN achieves the strongest overall effectiveness (accuracy, macro-F1, weighted-F1) under the shared metadata-free evaluation protocol.

Efficiency Trade-off

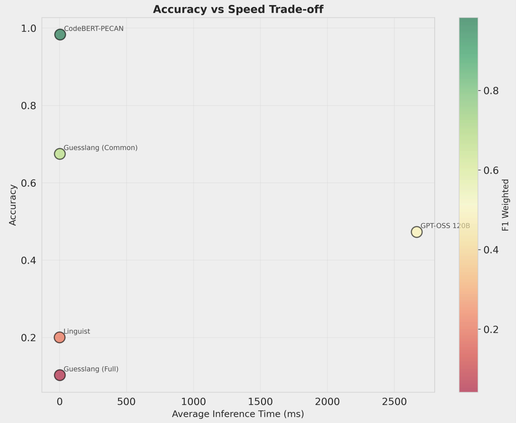


Figure 2 Accuracy vs Speed Trade-off
The quality-latency frontier shows CodeBERT-PECAN near the high-accuracy/low-latency region, while GPT-OSS is substantially slower despite moderate accuracy.

Error Analysis

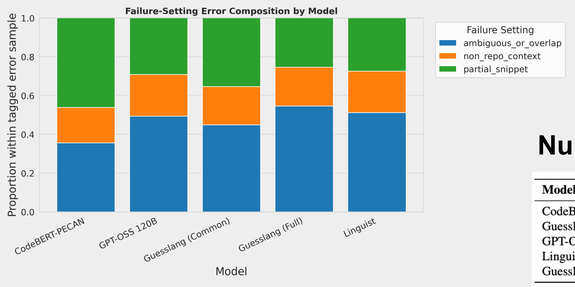


Figure 3 Failure-Setting Error Composition by Model
Error composition differs by model family across partial snippets, non-repository context, and ambiguous/overlap settings, highlighting where each approach is most brittle.

Numeric Summary

Model	Accuracy	F1 (Macro)	Inference (ms)	Unknown Rate	Answered Rate	Accuracy (Answered-Only)
CodeBERT-PECAN	0.9831	0.9711	4.89	0	1	0.9831
Guesslang (Common)	0.6746	0.5696	2.54	0.0005	0.9995	0.6749
GPT-OSS 120B	0.4727	0.3733	2667.48	0.0004	0.9996	0.4729
Linguist	0.2001	0.1492	0.85	0.0227	0.9773	0.2048
Guesslang (Full)	0.1025	0.0346	2.54	0.0002	0.9998	0.1025

Sorted by accuracy (descending)

Table 1 Condensed Model Comparison Table
Numeric summary of effectiveness, efficiency, and reliability/coverage metrics; rankings are consistent with the chart-based comparisons.

- Canonical vs raw deltas were small/absent for all models; conclusions unchanged.

Conclusion

- Metadata-free PL-ID should be evaluated as a multi-objective problem:
- predictive quality, output reliability, and runtime behavior
- A shared, reproducible pipeline enables fair comparison across
- practical tools, encoder models, and LLMs
- CodeBERT-PECAN achieves state-of-the-art accuracy for metadata-free programming language classification of code fragments

References

[1] Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., & Zhou, M. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. Proceedings of the Association for Computational Linguistics (ACL), 2020.
[2] Guesslang. Deep Learning for Source Code Language Detection. GitHub Repository. Available at: https://github.com/yoeo/guesslang
[3] GitHub. Linguist: Language Detection Library. GitHub Repository. Available at: https://github.com/github/linguist
[4] GPT-OSS. GPT-OSS: Open-Source Large Language Model. arXiv:2508.10925, 2025. Available at: https://arxiv.org/abs/2508.10925
[5] Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., & Stoica, I. Efficient Memory Management for Large Language Model Serving with PagedAttention. arXiv:2309.06180, 2023.

Authors


Ibrahim Alam
ialam2@lsu.edu


Jackson Descant
jdesca@lsu.edu


James Ghawaly
jghawaly@lsu.edu