**Literature Review: Automated Variable Renaming Using Large Language Models (LLMs)**

**1. Introduction**

Variable names significantly impact code comprehension, maintainability, and readability. However, developers frequently use inconsistent or uninformative variable names, making code harder to understand. Automated variable renaming aims to enhance identifier quality using **machine learning (ML) techniques**, including **statistical language models, deep learning models, and static code analysis**.

The paper **"Automated Variable Renaming: Are We There Yet?"** by Mastropaolo et al. [1] presents a **large-scale empirical study** on **data-driven variable renaming techniques**, evaluating their effectiveness in recommending meaningful identifiers. This review provides an in-depth discussion of the study’s methodologies, datasets, evaluation metrics, and key findings, alongside critical reflections on its contributions and limitations.

**2. Summary of the Reviewed Study**

**2.1 Research Goal**

The study investigates how well **state-of-the-art language models** can predict **meaningful variable names**, potentially **automating rename refactoring**. The authors evaluate three models:

1. **N-gram Cached Language Model** – A statistical model that predicts words based on preceding sequences [2].
2. **T5 Transformer Model** – A deep learning model trained to understand programming languages [3].
3. **CugLM (Transformer-based Code Completion Model)** – A neural model designed for predicting identifiers in source code [4].

These models are tested on datasets derived from **real-world software projects**, comparing their ability to **recommend variable names matching developers’ choices**.

**2.2 Methodology and Datasets**

The study constructs **three datasets** to evaluate the models:

* **Large-Scale Dataset:**
  + 1,221,193 Java method instances with local variables.
  + Extracted from **1,425 GitHub repositories** to train and test models.
  + projects on GitHub website (2008) by using the search tool by Dabic et al. (2021). This tool indexes all GitHub repositories written in 13 different languages and having at least 10 stars, providing a handy querying interface (SEART GitHub search 2021) to identify projects meeting specific selection criteria.
* **Reviewed Dataset:**
  + 457 Java methods with variable renaming that occurred **during code reviews**.
  + Ensures **higher-quality identifiers** that multiple developers agreed upon.
* **Developers’ Dataset:**
  + 442 instances of **variable renaming extracted from rename-refactoring commits**.
  + Represents **real-world refactoring** cases performed by developers.

Each model is trained on the large-scale dataset and evaluated on **all three datasets**.

**2.3 Evaluation Metrics**

The performance of the models is assessed using:

* **Prediction Accuracy:** Measures how often the predicted identifier matches the developer’s choice.
* **Confidence Score Analysis:** Examines how confident the models are in their predictions and correlates confidence with correctness.
* **McNemar’s Test & Odds Ratio (OR):** Used for **statistical comparison** of model performance.
* **Qualitative Analysis:** Manually inspects incorrect predictions to determine if they are still **meaningful** alternatives.

**2.4 Key Findings**

1. **CugLM significantly outperforms other models**, achieving **63.46% accuracy** on the large-scale dataset.
   * **T5 Transformer achieves 37.35%**, while the **N-gram model lags at 10.54%**.
2. **Performance drops on high-quality datasets** (reviewed & developers’ datasets), indicating **challenges in predicting meaningful names**.
3. **Confidence scores strongly correlate with correctness**, suggesting models can be used in **refactoring tools with confidence thresholds**.
4. **Some incorrect predictions are still useful**, offering **valid alternative names** (e.g., sum → totalAmount).

**3. Reflection and Research Insights**

**3.1 Contributions**

* This study is **one of the largest empirical evaluations** of **data-driven variable renaming**.
* It introduces **high-quality datasets** that enable **rigorous model evaluation**.
* Findings suggest that **LLMs like CugLM are ready for real-world integration** into **automated refactoring tools**.

**3.2 Limitations**

* **Bias toward frequent identifiers**: Models tend to predict **common names** rather than **domain-specific ones**.
* **Contextual understanding is limited**:
  + Models **struggle with longer, descriptive variable names**.
  + They often **fail to grasp the deeper semantics** of a variable in the codebase.
* **Over-reliance on training data**: Models perform **best on identifiers they have seen before**, limiting their generalization ability.

**3.3 Future Research Directions**

* **Enhancing semantic understanding**: Integrating **static code analysis** and **semantic representations** can improve predictions.
* **Expanding datasets**: Training on **multi-language repositories** and **industry-specific projects** can improve generalization.
* **Human-in-the-loop approaches**: Combining **LLM suggestions with developer feedback** can refine identifier recommendations.

**4. Conclusion**

The study provides **strong empirical evidence** that **LLMs can effectively support automated variable renaming**, with **CugLM achieving state-of-the-art performance**. However, challenges remain in **handling rare identifiers, improving context awareness, and ensuring trust in predictions**. Future research should focus on **enhancing semantic understanding** and **leveraging human-in-the-loop refinements** to build **practical rename-refactoring tools**.

**References**

[1] A. Mastropaolo, E. Aghajani, L. Pascarella, and G. Bavota, **“Automated Variable Renaming: Are We There Yet?”** *Empirical Software Engineering*, vol. 28, no. 45, 2023. DOI: [10.1007/s10664-022-10274-8](https://doi.org/10.1007/s10664-022-10274-8).

[2] V. Hellendoorn and P. Devanbu, **"Are Deep Neural Networks the Best Choice for Modeling Source Code?"** *Proceedings of the 2017 ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE)*, 2017, pp. 763–773.

[3] C. Raffel et al., **“Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”**, *Journal of Machine Learning Research*, vol. 21, pp. 1–67, 2020.

[4] C. Liu et al., **“Multi-Task Learning for Code Completion”**, *Proceedings of the 2020 ACM/IEEE International Conference on Automated Software Engineering (ASE)*, 2020, pp. 473–485.