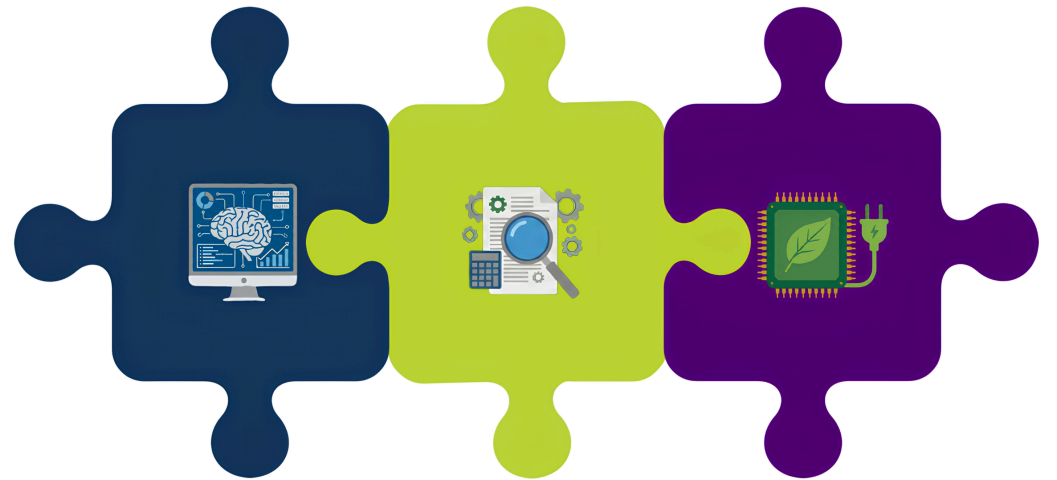


Applications of Large Language Models

Addressing Real-World Challenges



GIOVANNI PINNA

GIOVANNI PINNA | Applications of Large Language Models

Applications of Large Language Models

Addressing Real-World Challenges

Ph.D. Dissertation

by

GIOVANNI PINNA

Applied Data Science and Artificial Intelligence
University of Trieste, Italy

Giovanni Pinna

Application of Large Language Models

Supervisor: Prof. Luca Manzoni

Co-Supervisor: Prof. Andrea De Lorenzo

University of Trieste

Department of Mathematics, Informatics and Geosciences

Ph.D. Program in Applied Data Science & Artificial Intelligence

Academic Year 2024/2025

"Per aspera ad astra"

To those who taught me that every obstacle is an opportunity,
that knowledge grows only when it is shared,
and that true success is never a solitary achievement.

To all those who walked this path with me.

Contents

Abstract	i
Sommario	iii
1 Introduction	1
1.1 The Evaluation Crisis	2
1.2 Specialized Knowledge Democratization	3
1.3 Environmental Sustainability and Green AI	4
1.4 Research Questions	6
1.5 Contributions and Organization	8
2 Conceptual Framework	13
2.1 Transformers	14
2.2 Genetic Algorithms and Genetic Programming	16
2.3 LLM Technologies	20
2.4 Evaluation of Large Language Models	23
2.5 Conclusion	26
I Output Quality and Improvement	27
3 LLMs for legal texts	29
3.1 Introduction	31
3.2 Related Works	33
3.3 Methodology	33
3.4 Experimental Evaluation	36
3.5 Results	40
3.6 Discussion	45
3.7 Conclusions	47
4 LLMG1py	49
4.1 Introduction	51
4.2 Related Works	53
4.3 Methodology	55
4.4 Experimental Evaluation	60
4.5 Results	65

4.6	Discussion	69
4.7	Conclusions	74
4.8	Current and Future Impact	76
II	Metrics and Evaluation	81
5	Query Affinity Score	83
5.1	Introduction	85
5.2	Related Work	87
5.3	Methodology	90
5.4	Experimental Evaluation	94
5.5	Results	97
5.6	Discussion	106
5.7	Conclusions	111
III	Sustainability and Green AI	113
6	Green feature selection	115
6.1	Introduction	117
6.2	Related Work	119
6.3	Methodology	120
6.4	Experimental Evaluation	121
6.5	Results	122
6.6	Discussion	125
6.7	Conclusions	128
7	GA4GC	131
7.1	Introduction	133
7.2	Related Work	135
7.3	Methodology	137
7.4	Experimental Setup	138
7.5	Results	139
7.6	Discussion	144
7.7	Conclusions	148
IV	Conclusions	151
8	General discussion	153
8.1	Addressing the Research Questions	154
8.2	Implications and Broader Impact	164

8.3	Reflections on the Present and Future of LLMs	165
8.4	Limitations	167
8.5	Concluding Remarks	168
A	Appendix	171
A.1	Example Legal Materials from Chapter 3	171
B	Appendix	179
	Bibliography	187
	Publications	201
	Logo Description	203
	Curriculum Vitae	205
	Acknowledgements	207
	Acronyms	209
	Glossary	215

Abstract

Large Language Models (LLMs) have fundamentally transformed artificial intelligence, demonstrating unprecedented capabilities in text generation, code synthesis, and complex reasoning. Yet their rapid adoption has outpaced the development of adequate evaluation frameworks, exposing critical gaps in our ability to assess their reliability, measure their true utility, and deploy them sustainably. This thesis addresses these challenges through four interconnected research questions that span evaluation methodologies, knowledge democratization, reliability enhancement, and sustainable deployment.

The first contribution concerns the development of evaluation approaches that go beyond surface-level metrics. In the legal domain, we conduct a human-centered evaluation involving 75 participants with varying levels of legal expertise, comparing LLM-generated summaries of Italian Constitutional Court judgments against expert-crafted *massime* (authoritative legal summaries produced by specialized offices). The results reveal that GPT-4o achieves only 38% accuracy on comprehension questions compared to 45% for expert-crafted *massime*, with the latter reaching 58% among legally knowledgeable participants. A counterintuitive inverse expertise-error relationship emerges: participants with legal knowledge are *more* misled by GPT-4o summaries than non-experts, suggesting that the model's fluency masks inaccuracies that entrap knowledgeable readers. In the text-to-SQL domain, we propose the Query Affinity Score (QAS), a continuous metric that combines semantic similarity of query structures with execution result similarity, enabling granular assessment of partial correctness that binary metrics such as Exact Match and Execution Accuracy systematically miss. Evaluation across 11 models from 5 families on the BIRD benchmark demonstrates that QAS captures meaningful quality distinctions invisible to existing metrics.

ABSTRACT

The investigation of knowledge democratization uncovers a fundamental tension: while LLMs improve accessibility through text simplification and length reduction, they simultaneously introduce accuracy risks that are particularly dangerous in high-stakes domains. In the legal domain, GPT-4o’s comprehension gains stem primarily from brevity rather than genuine conceptual simplification, and the resulting summaries increase error rates. In code generation, even state-of-the-art models directly solve only approximately half of benchmark problems from the PSB2 suite, with systematic failures on edge cases and complex algorithmic reasoning that persist regardless of model scale. To address the resulting reliability gap, we develop a Genetic Improvement (GI) framework that combines LLM code generation with grammatical evolution using dynamically specialized grammars. This hybrid approach achieves statistically significant improvements in functional correctness across nearly all tested model-problem configurations, frequently yielding 10–50× increases in test pass rates, while self-correction—where LLMs attempt to fix their own output—proves largely ineffective.

Finally, we contribute multi-objective optimization frameworks for sustainable LLM deployment. In software maintenance, we present HotCat, a pipeline that employs NSGA-II for feature selection in bug taxonomy classification over 88 hotfix records across 17 categories, achieving 63% accuracy while reducing computational cost through selective feature inclusion. For coding agent optimization, we introduce GA4GC (Greener Agent for Greener Code), a framework that discovers non-dominated agent configurations through multi-objective optimization of hyperparameters including temperature, token limits, and timeout settings. Under a constrained evaluation budget, GA4GC achieves up to 37.7% runtime reduction while simultaneously improving correctness, with a 135-fold hypervolume improvement over default configurations. Collectively, these contributions establish an integrated framework for responsible LLM deployment that balances capability, reliability, and environmental sustainability.

Sommario

I Large Language Model (LLM) hanno trasformato radicalmente l'intelligenza artificiale, dimostrando capacità senza precedenti nella generazione di testo, nella sintesi di codice e nel ragionamento complesso. Tuttavia, la loro rapida adozione ha preceduto lo sviluppo di framework di valutazione adeguati, evidenziando lacune critiche nella capacità di verificarne l'affidabilità, misurarne la reale utilità e garantirne un impiego sostenibile. Questa tesi affronta tali sfide attraverso quattro domande di ricerca interconnesse che spaziano dalle metodologie di valutazione alla democratizzazione della conoscenza, dal miglioramento dell'affidabilità allo sviluppo sostenibile.

Il primo contributo riguarda lo sviluppo di approcci di valutazione che superano le metriche superficiali. Nel dominio giuridico, conduciamo una valutazione human-centered con 75 partecipanti con diversi livelli di competenza legale, confrontando riassunti generati da LLM di sentenze della Corte Costituzionale italiana con *massime* redatte da esperti (riassunti giuridici autorevoli prodotti da uffici specializzati). I risultati mostrano che GPT-4o raggiunge solo il 38% di accuratezza sulle domande di comprensione, contro il 45% delle *massime* redatte da esperti, che arrivano al 58% tra i partecipanti con conoscenze giuridiche. Emerge una controintuitiva relazione inversa tra competenza ed errore: i partecipanti con formazione giuridica vengono *maggiormente* ingannati dai riassunti di GPT-4o rispetto ai non esperti, suggerendo che la fluenza del modello maschera imprecisioni che intrappolano i lettori competenti. Nel dominio text-to-SQL, proponiamo il Query Affinity Score (QAS), una metrica continua che combina la similarità semantica delle strutture delle query con la similarità dei risultati di esecuzione, consentendo una valutazione granulare della correttezza parziale che le metriche binarie come Exact Match ed Execution Accuracy non riescono a catturare. La valutazione su 11 modelli di 5

SOMMARIO

famiglie sul benchmark BIRD dimostra che il QAS rivela distinzioni qualitative invisibili alle metriche esistenti.

L'indagine sulla democratizzazione della conoscenza rivela una tensione fondamentale: mentre gli LLM migliorano l'accessibilità attraverso la semplificazione e la riduzione della lunghezza dei testi, introducono contemporaneamente rischi di accuratezza particolarmente pericolosi in contesti ad alta criticità. Nel dominio legale, i miglioramenti di comprensione di GPT-4o derivano principalmente dalla brevità piuttosto che da una genuina semplificazione concettuale, e i riassunti risultanti aumentano i tassi di errore. Nella generazione di codice, anche i modelli più avanzati risolvono direttamente solo circa la metà dei problemi della suite PSB2, con fallimenti sistematici sui casi limite e sul ragionamento algoritmico complesso che persistono indipendentemente dalla scala del modello. Per colmare il conseguente gap di affidabilità, sviluppiamo un framework di Genetic Improvement (GI) che combina la generazione di codice tramite LLM con l'evoluzione grammaticale basata su grammatiche specializzate dinamicamente. Questo approccio ibrido ottiene miglioramenti statisticamente significativi nella correttezza funzionale su quasi tutte le configurazioni modello-problema testate, raggiungendo frequentemente incrementi di $10-50\times$ nei tassi di superamento dei test, mentre la self-correction—in cui gli LLM tentano di correggere il proprio output—si rivela sostanzialmente inefficace.

Infine, contribuiamo con framework di ottimizzazione multi-obiettivo per il deployment sostenibile degli LLM. Nella manutenzione software, presentiamo HotCat, una pipeline che impiega NSGA-II per la selezione delle feature nella classificazione tassonomica dei bug su 88 record hotfix distribuiti in 17 categorie, raggiungendo il 63% di accuratezza riducendo al contempo il costo computazionale attraverso l'inclusione selettiva delle feature. Per l'ottimizzazione degli agenti di coding, introduciamo GA4GC (Greener Agent for Greener Code), un framework che scopre configurazioni non dominate degli agenti attraverso l'ottimizzazione multi-obiettivo di iperparametri quali temperatura, limiti di token e timeout. Con un budget di valutazione limitato, GA4GC raggiunge una riduzione del runtime fino al 37.7% migliorando contemporaneamente la correttezza, con un incremento dell'hypervolume di 135 volte

rispetto alle configurazioni predefinite. Complessivamente, questi contributi stabiliscono un framework integrato per il deployment responsabile degli LLM che bilancia capacità, affidabilità e sostenibilità ambientale.

1

Introduction

The advent of Large Language Model (LLM) represents a major turning point in artificial intelligence (AI) and computer science. Within less than a decade, these models have evolved from experimental curiosities into sophisticated systems capable of performing tasks that, until recently, were considered the exclusive domain of human cognition [15, 3, 147]. The capacity to generate coherent prose, synthesize source code, engage in logical reasoning, and process multimodal inputs has fundamentally altered our understanding of what computational systems can achieve. This transformation extends far beyond incremental progress; it forces researchers to rethink basic assumptions about machine intelligence, evaluation methodologies, and the societal implications of automated reasoning systems.

The trajectory of LLM development has been characterized by rapid scaling in model size, training data, and computational resources. Models such as GPT-4 [3], Claude [5], and LLaMA [147] demonstrate capabilities that consistently exceed expectations derived from earlier generations of natural language processing (NLP) systems. These models exhibit emergent abilities—i.e., capabilities that appear qualitatively different from those present in smaller models—including few-shot and zero-shot learning, Chain of Thought (CoT) reasoning, and the ability to follow complex, multi-step instructions [155]. Such emergent phenomena suggest that scaling laws in deep learning (DL) may continue to yield surprises, raising both optimism about future capabilities and

concerns about our ability to adequately anticipate and evaluate these systems.

However, this remarkable progress has introduced a profound challenge. The evaluation frameworks, benchmarks, and metrics that guided NLP research for decades are no longer adequate for assessing systems of this complexity and capability. Traditional metrics were designed for narrowly defined tasks with clear ground truth. LLMs, by contrast, operate across a vast spectrum of tasks, often producing outputs that are semantically correct but syntactically diverse, or that exhibit subtle errors undetectable by surface-level comparison. This evaluation crisis threatens to undermine our ability to guide research, compare systems, and deploy LLMs responsibly in real-world applications.

1.1 The Evaluation Crisis

The inadequacy of existing evaluation methodologies manifests across multiple dimensions. Consider the domain of code generation, where LLMs have demonstrated impressive capabilities in synthesizing programs from natural language specifications [23]. Standard evaluation relies on test-driven metrics that assess whether generated code passes a predefined test suite. While functional correctness is a necessary criterion, binary pass/fail metrics fail to capture the nuanced quality of generated solutions. A program that handles core functionality but fails on edge cases receives the same evaluation as one that produces entirely incorrect output. Moreover, limited test coverage can create false positives, where code passes available tests but exhibits latent defects that surface only in production environments.

The text-to-SQL domain exemplifies similar challenges. Evaluation metrics such as Exact Match (EM) and Execution Accuracy (EX) have become standard benchmarks for assessing systems that translate natural language queries into structured database queries [162]. However, EM penalizes semantically equivalent queries that differ in syntactic details, such as table alias choices or clause ordering, while EX provides only binary feedback that obscures the degree of similarity between generated and reference queries. A query that correctly identifies relevant tables and columns but applies an incorrect filter condition receives the same negative evaluation as one that misinterprets

the entire request. This lack of granularity impedes understanding of model capabilities and limits the ability to identify specific areas for improvement.

Human evaluation, while conceptually more robust, introduces its own complexities. The assessment of LLM-generated summaries, translations, or creative outputs requires careful experimental design to ensure that evaluations measure genuine utility rather than surface-level fluency [20]. Furthermore, human evaluation is resource-intensive, limiting its applicability to large-scale studies or iterative model development.

Beyond task performance, the evaluation of LLMs must encompass broader considerations that extend to their operational characteristics and societal impacts. The computational resources required for training and inference have grown exponentially, raising questions about the environmental sustainability of current approaches. While LLMs may democratize access to specialized knowledge, they also carry risks: misuse, misinformation, and erosion of human expertise. A comprehensive evaluation framework must therefore adopt a multidimensional perspective that considers not only what LLMs can do, but how they do it and what consequences follow from their deployment.

1.2 Specialized Knowledge Democratization

One of the most promising aspects of LLM technology lies in its potential to democratize access to domains of knowledge traditionally reserved for specialized experts. Numerous professional fields—including law, medicine, software engineering, and scientific research—are characterized by complexity and specialized vocabulary that have historically created barriers to entry for non-experts. The capacity of LLMs to interpret, summarize, and generate content in these domains offers new opportunities for broadening participation and reducing information asymmetries [14].

The democratization potential manifests differently across domains, but common patterns emerge. Specialized texts often employ terminology, syntactic structures, and implicit assumptions that presuppose extensive background knowledge. Citizens seeking to understand legal rights, patients researching medical conditions, or analysts attempting to query databases all face similar

challenges: the information they need exists but remains locked behind linguistic and conceptual barriers. Traditional approaches to bridging these gaps rely on human intermediaries—lawyers, physicians, database administrators—whose expertise is expensive and cannot scale to meet growing demand [143].

LLMs offer the potential to automate aspects of this knowledge translation, transforming impenetrable texts into comprehensible explanations and enabling direct interaction with complex systems through natural language. The implications extend beyond convenience to central issues of equity and access. When specialized knowledge determines life outcomes—legal rights, health decisions, economic opportunities—barriers to understanding perpetuate existing inequalities. Technologies that genuinely democratize access could contribute to more equitable societies.

However, the promise of democratization must be tempered by recognition of current limitations and associated risks. LLMs, despite their impressive capabilities, remain imperfect. Generated summaries may contain inaccuracies that mislead users about critical information. Synthesized outputs may exhibit subtle errors that cause failures in consequential applications. The gap between apparent fluency and genuine correctness represents a critical concern: users may trust LLM outputs precisely because they appear authoritative, yet this appearance can mask underlying errors. Moreover, over-reliance on automated systems risks displacing human expertise entirely, potentially degrading the quality of specialized knowledge over time. These considerations underscore the importance of rigorous evaluation methodologies that can distinguish genuine utility from superficial fluency, and of hybrid approaches that combine AI efficiency with appropriate human oversight.

1.3 Environmental Sustainability and Green AI

The remarkable capabilities of LLMs come at a substantial computational cost. Training state-of-the-art models requires vast quantities of data, thousands of specialized processors, and weeks or months of continuous computation. Strubell et al. [142] demonstrated that training large transformer models can emit carbon dioxide equivalent to the lifetime emissions of multiple

automobiles. As models continue to scale, these environmental costs grow correspondingly, raising fundamental questions about the sustainability of current research trajectories.

The **green artificial intelligence (green AI)** community, pioneered by Schwartz et al. [137], advocates for developing **machine learning (ML)** approaches that prioritize computational efficiency alongside accuracy. This perspective argues that the research community has excessively focused on performance gains achieved through increased computation, while neglecting the costs associated with such gains. An evaluation framework that considers only task performance, without accounting for resource consumption, implicitly incentivizes inefficient approaches and contributes to unsustainable practices.

Inference costs compound the concerns raised by training. LLM-based applications that serve millions of users generate substantial cumulative computational loads. Agentic workflows, where LLMs operate as autonomous agents capable of iterative reasoning and tool utilization, amplify these costs through multiple sequential model invocations [6]. Recent analyses indicate that a single agent run on real-world **software engineering (SE)** tasks can consume over 100,000 tokens, translating to high monetary and environmental costs per task [27].

The environmental impact of AI systems intersects with broader concerns about climate change and resource sustainability. The International Energy Agency has documented the growing energy consumption of data centers, which host the infrastructure required for large-scale AI deployment [74]. Alignment with **green software engineering (GSE)** principles [84] and Net Zero targets established by international climate initiatives requires attention not only to what AI systems accomplish but to the resources they consume in the process.

Addressing these challenges requires multi-objective optimization approaches that balance performance against resource consumption. Feature selection techniques can reduce the dimensionality of inputs, decreasing both processing time and energy requirements while maintaining predictive accuracy. Configuration optimization can identify settings that achieve accept-

able performance with reduced computational overhead. These approaches represent practical strategies for sustainable AI deployment that complement broader efforts to improve algorithmic efficiency and hardware utilization.

1.4 Research Questions

This thesis addresses the multifaceted challenges outlined above through a series of interconnected research investigations. The overarching goal is to develop a comprehensive understanding of how to evaluate, improve, and sustainably deploy LLM-based systems across diverse application domains. Four primary research questions guide this work:

RQ1: *How can we develop more accurate and meaningful evaluation methodologies for LLM outputs that capture semantic correctness and practical utility beyond surface-level metrics?*

Traditional evaluation metrics fail to capture the nuanced quality of LLM outputs across domains ranging from natural language generation (NLG) to code synthesis. This research question investigates novel evaluation approaches that provide granular, informative assessments of system capabilities. We address this question through human-centered evaluation of legal summarization, where comprehension questions designed by domain experts measure genuine understanding rather than perceived quality, and through the development of continuous metrics for text-to-SQL evaluation that capture degrees of similarity rather than binary correctness.

RQ2: *To what extent can LLMs democratize access to specialized knowledge in the legal and software engineering domains, and what are the limitations and risks of current systems in these contexts?*

The potential for LLMs to make specialized knowledge accessible to non-experts could reshape how non-experts access specialized information. This research question examines the degree to which current systems can fulfill this promise while identifying limitations and risks that must

be addressed. We investigate this question through experiments on legal text simplification, comparing LLM-generated summaries to expert-crafted content and assessing comprehension across participants with varying expertise, and through analysis of code generation capabilities across varying problem complexities and programming languages.

RQ3: *What post-processing and hybrid approaches can enhance the reliability of LLM outputs in high-stakes applications?*

The gap between LLM capabilities and the reliability required for deployment in consequential domains necessitates techniques that improve output quality. This research question explores approaches that complement LLM generation with additional verification, refinement, or optimization steps. We address this question through the development of **genetic improvement (GI)** techniques for LLM-generated code, demonstrating how **evolutionary algorithm (EA)** can systematically enhance functional correctness without requiring model retraining.

RQ4: *How can we develop and deploy LLM-based systems while minimizing their environmental footprint and ensuring sustainable resource utilization?*

The computational costs of LLM training and inference raise key challenges about sustainability. This research question investigates approaches that balance task performance against resource consumption, contributing to the **green AI** field. We address this question through multi-objective optimization of feature selection for classification tasks and configuration optimization for coding agents, demonstrating practical strategies for reducing environmental impact while maintaining effectiveness.

These research questions are interconnected in important ways. Improved evaluation methodologies (RQ1) provide the foundation for assessing progress on democratization (RQ2), reliability enhancement (RQ3), and sustainability (RQ4). Understanding current limitations (RQ2) motivates the development of hybrid approaches (RQ3) and informs decisions about when LLM deployment

is appropriate. Sustainability considerations (RQ4) must be integrated throughout, ensuring that improvements in capability do not come at unacceptable environmental costs.

1.5 Contributions and Organization

This thesis makes contributions across multiple dimensions of LLM evaluation, application, and sustainable deployment. The primary contributions are organized around the research questions presented above:

EVALUATION METHODOLOGIES. We develop and validate novel approaches to evaluating LLM outputs that address the limitations of existing metrics. In the legal domain, we conduct a human-centered evaluation involving human participants with varying levels of legal knowledge, demonstrating that comprehension-based assessment provides more meaningful insights than automatic metrics. In the text-to-SQL domain, we propose the **Query Affinity Score (QAS)**, a continuous metric that combines semantic similarity of query structures with execution result similarity, providing granular feedback that binary metrics obscure.

DEMOCRATIZATION AND ACCESSIBILITY. We investigate the potential for LLMs to broaden access to specialized knowledge domains. Our work on legal summarization examines whether LLM-generated summaries can enhance comprehension for individuals without legal expertise compared to reading original judgments, and how AI-generated summaries compare to human-crafted summaries in accuracy and comprehensibility. We also examine code generation capabilities across varying problem complexities, identifying systematic limitations that persist regardless of model scale.

RELIABILITY ENHANCEMENT. We develop hybrid approaches that improve the reliability of LLM outputs for high-stakes applications. Our GI framework combines LLM generation with evolutionary optimization, using grammatical evolution with dynamically specialized grammars to refine generated code through

test-driven fitness evaluation. Experimental results demonstrate that the proposed method can improve in a statistically significant way the code generated by LLMs, with improvements frequently achieving $10\text{--}50\times$ increases in test pass rates.

SUSTAINABLE DEPLOYMENT. We contribute to the **green AI** agenda through multi-objective optimization approaches that balance performance against resource consumption. In software maintenance, we develop HotCat, a pipeline that employs NSGA-II to identify feature subsets for bug classification that balance predictive accuracy with computational efficiency. For coding agents, we introduce GA4GC, a framework that identifies non-dominated agent configurations through multi-objective optimization, achieving substantial runtime reductions while improving correctness under a constrained evaluation budget.

THESIS ORGANIZATION. The thesis is structured into three thematic parts, followed by a concluding discussion, as illustrated in Figure 1.1. Following this introduction, Chapter 2 provides foundational background on LLMs, transformer architectures, evaluation methodologies, and the application domains addressed in subsequent chapters; this chapter serves as a reference for the entire thesis. Throughout this thesis, we distinguish between thesis-level research questions (RQ1–RQ4, defined in this chapter) and chapter-specific research questions denoted as cXRQY, where X is the chapter number and Y the question number within that chapter.

Part I: Output Quality and Improvement examines how LLMs can be applied and enhanced for specialized tasks. Chapter 3 investigates LLM-based legal summarization, presenting a human-centered evaluation methodology and comparative analysis with expert-crafted summaries that addresses RQ1 and RQ2. Chapter 4 examines code generation capabilities and introduces our GI approach to enhancing functional correctness, contributing to RQ2 and RQ3.

Part II: Metrics and Evaluation focuses on developing novel evaluation frameworks. Chapter 5 introduces the QAS for text-to-SQL evaluation and presents experimental validation across multiple benchmarks, directly addressing RQ1.

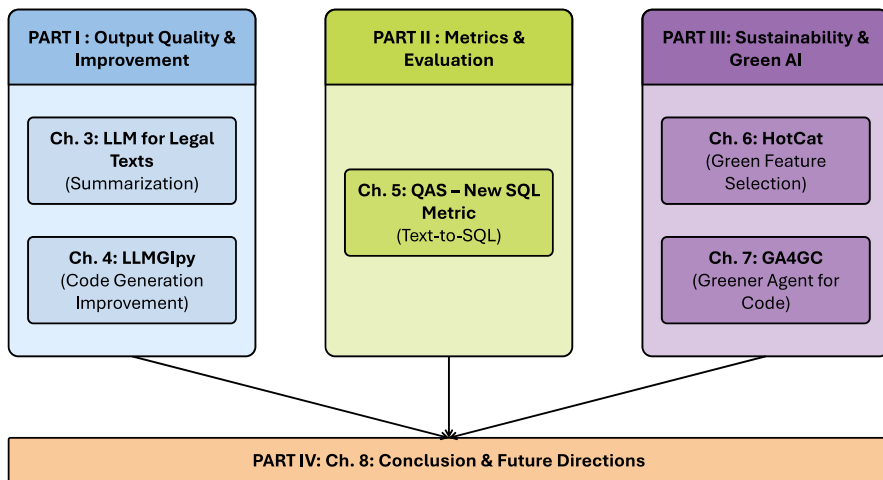


Figure 1.1: Thesis organization. The research is structured into three thematic parts addressing output quality improvement, evaluation metrics, and sustainability, followed by a concluding discussion that synthesizes the findings.

Part III: Sustainability and Green AI addresses the environmental implications of LLM deployment. Chapter 6 describes HotCat and our approach to green feature selection for software maintenance, while Chapter 7 presents the GA4GC framework for sustainable coding agent deployment. Both chapters contribute to RQ4.

Finally, **Part IV: Conclusions** provides a comprehensive synthesis: Chapter 8 integrates findings across all chapters, discusses broader implications for the field, and identifies promising directions for future research. Unlike the preceding thematic parts, this final part does not present new empirical contributions but rather reflects on the collective insights of the thesis.

2

Background

This chapter establishes the technical foundations necessary for understanding the research presented in this thesis. We begin by examining the Transformer architecture and the self-attention mechanisms that underpin modern LLMs. The discussion then introduces evolutionary computation techniques, specifically genetic algorithm (GA) and genetic programming (GP), which provide the optimization framework for our hybrid approaches. Subsequently, we survey the landscape of LLMs for code generation and text processing, discussing their capabilities and deployment considerations. Finally, we address the critical challenges of LLM evaluation, highlighting the limitations of current metrics and benchmarks that motivate the methodological contributions of this thesis.

2.1 Transformers

The Transformer architecture in Figure 2.1, introduced in “Attention is All You Need” [149], relies entirely on self-attention mechanisms to model dependencies between input and output sequences. Unlike sequential processing models, Transformers process the entire sequence in parallel, enabling efficient training and capturing long-range relationships regardless of the distance between elements.

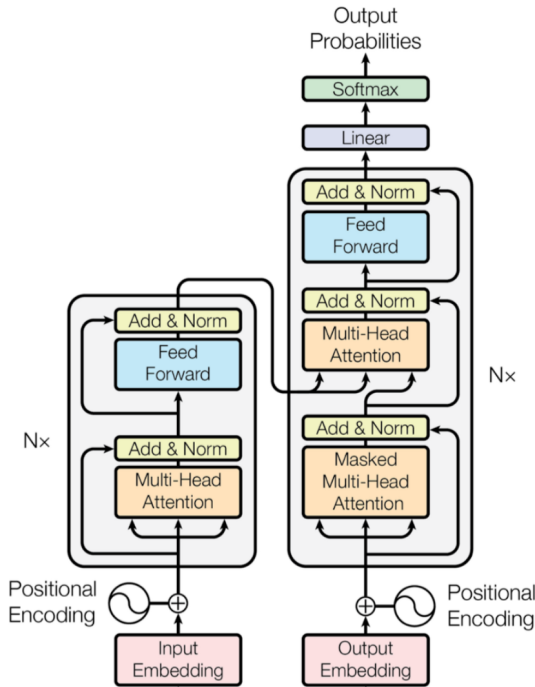


Figure 2.1: Transformer Architecture

The core of the Transformer is the attention mechanism, which maps a query and a set of key-value pairs to an output. Inputs X are projected into Query (Q), Key (K), and Value (V) matrices via learned linear transformations:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V \quad (2.1)$$

Scaled dot-product attention computes the relevance of keys to queries to weight the values:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.2)$$

The scaling factor $\sqrt{d_k}$ ensures gradient stability. To capture diverse relationships, Multi-Head Attention runs this process in parallel h times, projecting the concatenated results via an output matrix W^O .

Encoder

The *Encoder* component consists of stacked layers containing multi-head self-attention and feed-forward networks. It processes the input bidirectionally, allowing each token to attend to all other positions in the sequence simultaneously. This structure facilitates the creation of rich, context-aware representations and forms the backbone of models like BERT [40], which excel in understanding tasks.

Decoder

On the other hand, the *Decoder* generates the output sequence autoregressively. While it also utilizes self-attention, it incorporates a masking mechanism to ensure causality by preventing positions from attending to future tokens. The masking is applied as follows:

$$\text{Mask}(i, j) = \begin{cases} 0 & \text{if } i \geq j \\ -\infty & \text{otherwise} \end{cases}$$

This constraint ensures that predictions depend only on known outputs. Additionally, the decoder employs cross-attention to leverage the encoder's output. Decoder-only architectures, such as the GPT family [126], rely on this autoregressive property for generative tasks.

2.2 Genetic Algorithms and Genetic Programming

GAs represent a class of metaheuristic optimization techniques inspired by the principles of natural evolution [67]. These algorithms maintain populations of candidate solutions that evolve through selection, recombination, and mutation operations, progressively improving solution quality across generations [164].

The fundamental advantage of GAs lies in their ability to explore complex, multimodal search spaces without requiring gradient information or continuity assumptions [112]. This property makes GAs particularly suitable for problems where traditional optimization methods fail, including discrete optimization, **Multi-Objective Optimization (MOO)**, and problems with deceptive or rugged fitness landscapes.

However, GAs also exhibit limitations. The stochastic nature of evolutionary operators introduces variability in solution quality and convergence time. Additionally, GAs typically require numerous fitness evaluations, which becomes prohibitive for expensive objective functions [81]. The balance between exploration and exploitation remains a persistent challenge, with premature convergence threatening solution quality while excessive exploration impedes convergence.

The evolutionary cycle in GAs follows a structured process. A population comprises individuals, each representing a potential solution. The *genotype* is the internal, encoded representation of a candidate solution as manipulated by the evolutionary algorithm—for example, a binary string, a sequence of integers, or a tree structure. The genotype determines how genetic operators (crossover and mutation) act on solutions and how offspring inherit traits from parents. The *phenotype*, by contrast, is the decoded solution expressed in the problem domain, obtained by interpreting the genotype according to a mapping function. For instance, in grammatical evolution, the genotype is a sequence of integers, while the phenotype is the executable program obtained by mapping those integers to grammar production rules. Selection mechanisms, such as tournament selection [110] and *lexicase selection* [63], identify promising individuals for reproduction based on fitness values. Crossover operations combine genetic material from parent individuals to produce offspring, promoting the

exploration of new solution regions while preserving beneficial traits [144]. Mutation introduces random perturbations, maintaining genetic diversity and enabling escape from local optima [9]. The fitness function quantifies solution quality, guiding the evolutionary process toward optimal solutions.

2.2.1 Genetic Programming (GP)

GP extends GA to evolve computer programs rather than fixed-length solution vectors [85]. Each individual in a GP population represents an executable program, typically structured as a syntax tree, linear sequence of instructions, or grammatical derivation [125].

The fitness function in GP often evaluates program behavior against test cases, measuring correctness, efficiency, or other domain-specific criteria [62]. For instance, when evolving programs to solve algorithmic problems, fitness might count the number of passed test cases or incorporate execution time and memory usage [121].

Crossover in GP exchanges subtrees or code segments between parent programs, potentially transferring functional modules between solutions [114]. Mutation operations include subtree replacement, node mutation, and insertion/deletion of code segments, with operation selection influencing the balance between exploration and exploitation [132].

Grammar-Based Genetic Programming (GBGP)

Grammar-based genetic programming (GBGP) employs formal grammars, typically in Backus-Naur Form (BNF), to constrain the search space and ensure syntactic validity of evolved programs [109]. A BNF grammar defines production rules specifying valid program structures:

```
<expr> ::= <expr> <op> <expr> | <var> | <const>
<op> ::= + | - | * | /
<var> ::= x | y
<const> ::= 0 | 1 | 2 | ... | 9
```

2. CONCEPTUAL FRAMEWORK

This grammatical framework offers several advantages. Syntactic constraints eliminate invalid programs from the search space, improving search efficiency [115]. Domain knowledge can be incorporated through grammar design, biasing evolution toward promising solution structures [64]. Grammars facilitate the evolution of programs in arbitrary languages without modifying the underlying GA [44].

Grammatical evolution (GE), a prominent GBGP variant, separates genotype and phenotype through an integer-based encoding that maps to grammar productions [114]. This separation enables the application of standard genetic operators while maintaining syntactic validity, combining the flexibility of variable-length representations with the simplicity of linear genomes [134].

2.2.2 Genetic Improvement (GI)

GI represents a specialized application of GP focused on improving existing software rather than evolving programs from scratch [120]. This approach leverages the observation that existing code, even if imperfect, contains valuable information about the problem domain and solution structure.

GI operates by generating variants of existing programs through localized modifications, including statement insertion, deletion, replacement, and reordering [88]. The search process targets specific improvement objectives such as bug fixing [90], performance optimization [87], energy consumption reduction [16], or functionality enhancement [59].

Recent advances integrate GI with LLMs, addressing the semantic incorrectness often present in LLM-generated code [121, 122]. By treating LLM output as initial solutions for evolutionary refinement, these hybrid approaches combine the broad knowledge of Language Model (LM) with the targeted optimization capabilities of EA. This synergy proves particularly effective when LLMs produce structurally sound but functionally flawed code, enabling GP to correct logical errors while preserving the overall program structure [122].

The fitness landscape in GI typically exhibits high locality, where small code changes to an existing program produce proportional fitness changes, facilitating gradient-like optimization through evolutionary search [89]. This property

distinguishes GI from general GP: while evolving programs from scratch typically produces more rugged fitness landscapes due to the absence of structural guidance, GI benefits from the initial program as a starting point, combined with test suites for fitness evaluation, making it particularly suitable for automated program repair and optimization.

2.2.3 Multi-Objective Optimization

Many real-world optimization problems involve multiple conflicting objectives that cannot be simultaneously optimized. MOO addresses this challenge by seeking Pareto-optimal solutions—configurations where no objective can be improved without degrading another [34]. The set of all Pareto-optimal solutions forms the *Pareto front*, providing decision-makers with a range of trade-off options rather than a single solution. In practice, optimization algorithms such as NSGA-II approximate the Pareto front by identifying *non-dominated* solutions within the evaluated population. These solutions are non-dominated with respect to the observed set, but may not be truly Pareto-optimal in the global sense, particularly when the number of evaluations is limited.

NSGA-II (Non-dominated Sorting Genetic Algorithm II) [37] has become the standard algorithm for MOO in SE due to its computational efficiency ($O(MN^2)$ complexity, where M is the number of objectives and N is the population size) and effective elitism mechanism. The algorithm employs non-dominated sorting to rank solutions and crowding distance to maintain diversity along the Pareto front. These properties make NSGA-II particularly suitable for problems with expensive fitness evaluations, such as those involving LLM inference or code execution.

MOO has found extensive application in SE contexts [58], including test case generation, software refactoring, and resource allocation. Recent work has extended these applications to AI systems, demonstrating that MOO can effectively balance competing concerns such as model accuracy, computational efficiency, and environmental sustainability [137]. The green AI movement advocates for treating computational efficiency as a primary evaluation metric alongside task performance, motivating the application of MOO to discover

configurations that achieve favorable trade-offs between effectiveness and resource consumption.

The environmental impact of AI has emerged as a pressing concern within the research community. Strubell et al. [142] demonstrated that training large Transformer models can emit carbon dioxide equivalent to the lifetime emissions of multiple automobiles, with estimates reaching 284 tonnes of CO₂ for certain architectures. More recent analyses indicate that inference, rather than training, increasingly dominates energy consumption for deployed systems, with a single year of inference potentially consuming over 25× the energy required for training [157]. These findings underscore the importance of configuration optimization and efficient deployment strategies for LLM-based systems.

2.3 LLM Technologies

LLMs act as Deep Neural Network (DNN) trained on massive text corpora to predict subsequent tokens in a sequence, learning statistical patterns and semantic relationships. A *token* is the fundamental unit of text processing in LLMs: rather than operating on individual characters or whole words, these models segment text into subword units using tokenization algorithms such as Byte Pair Encoding (BPE) [138]. Common words may correspond to single tokens, while rare or complex words are decomposed into multiple tokens. For example, the word “understanding” might be tokenized as “under” + “stand” + “ing”. This subword tokenization balances vocabulary size with the ability to represent any text. Formally, an LLM models the conditional probability distribution $P(x_t \mid x_1, x_2, \dots, x_{t-1})$. Through self-supervised learning on billions of tokens, these models develop emergent capabilities such as reasoning and generalization without explicit task-specific training [15].

Since the release of GPT-1, the field has evolved rapidly from relatively small models to massive systems with trillions of parameters like GPT-4. While proprietary models initially dominated, the landscape has broadened with open-weight families such as Meta’s LLaMA, Mistral, and Google’s Gemma. These families offer various sizes and specializations, effectively democratiz-

ing access to high-performance models and demonstrating that careful data curation can sometimes rival pure scale [147, 77].

2.3.1 Training and Generation Objectives

While the underlying architecture relies on the Transformer (discussed in Section 2.1), the core training objective is Causal Language Modeling. The model minimizes the negative log-likelihood of the next token given the context:

$$\mathcal{L} = - \sum_{t=1}^T \log P(x_t | x_{<t}; \theta) \quad (2.3)$$

Text generation is performed autoregressively. So, at each step, the model produces logits that are transformed into probabilities via softmax. Strategies like *Temperature Sampling* modulate this distribution: lower temperatures make the output distribution more peaked, favoring the highest-probability tokens and producing more deterministic and repetitive outputs, while higher temperatures flatten the distribution, increasing diversity and creativity at the cost of coherence [69]. It is important to note that lower temperatures do not guarantee factual correctness—they merely reduce output variability.

2.3.2 Optimization and Adaptation

Efficiently deploying these models requires managing significant computational resources. Key parameters include the *context window*, which has grown from a few thousand to millions of tokens, allowing the processing of entire codebases. To reduce memory footprints, optimization techniques like *quantization* are employed, compressing weights from 16-bit floating-point to 8-bit or even 4-bit integers with minimal loss in accuracy [39]. For adapting pre-trained models to specific tasks, *Parameter-Efficient Fine-Tuning* (PEFT) methods like *Low-Rank Adaptation (LoRA)* are preferred. LoRA freezes the pre-trained weights W_0 and injects trainable low-rank matrices ($W = W_0 + BA$), reducing trainable parameters by orders of magnitude [70].

2.3.3 Advanced Prompting

Beyond Fine-Tuning (FT), model performance can be enhanced at inference time through In-Context Learning (ICL). By providing examples within the prompt (few-shot learning), models can adapt to specific patterns without weight updates. For complex tasks, techniques like CoT prompting encourage the model to generate intermediate reasoning steps before the final answer, significantly improving performance in logic and coding problems [156].

2.3.4 AI Agents for Code Generation

The evolution of LLMs has led to the emergence of AI Agents. In this thesis, we use this term specifically to refer to autonomous software systems that leverage an LLM as their core reasoning engine, integrated with external tools and feedback loops, to accomplish multi-step tasks with minimal human intervention [159]. Unlike passive models, agents incorporate a reasoning core—the LLM—integrated with memory systems (short-term context and long-term vector storage) and external tools such as code interpreters, file system access, and web search [159]. In the context of SE, these agents can plan multi-step solutions, generate code, execute it to check for errors, and iteratively refine their implementation based on feedback (Self-Reflection). This paradigm shifts the role of the LLM from a coding assistant to an autonomous developer capable of handling repository-level tasks.

In practice, several AI coding agents have reached widespread deployment. Notable examples include GitHub Copilot [173], OpenAI Codex [23], Devin by Cognition AI, Cursor, and Claude Code by Anthropic. These agents differ substantially in their operational modality: some function as interactive code-completion assistants embedded in integrated development environments (*e.g.*, Copilot, Cursor), while others operate as fully autonomous agents capable of creating and submitting complete pull requests without human intervention (*e.g.*, Devin, Claude Code, OpenAI Codex). Li et al. [94] characterize this transition as “Software Engineering 3.0,” defined by autonomous AI teammates capable of completing complex development activities with minimal human supervision.

Recent empirical studies have begun to systematically evaluate these agents

in real-world settings. Pinna et al. [124] analyzed 7,156 pull requests across five AI coding agents, finding that task type is a dominant factor influencing acceptance rates—with a 29 percentage-point gap between task categories that substantially exceeds inter-agent variance. Their task-stratified analysis revealed that no single agent outperforms all others across all task types: OpenAI Codex achieves consistently high acceptance rates, while Claude Code leads in documentation tasks and Cursor excels in bug-fixing. Complementarily, Gong et al. [52] investigated the reliability of AI-generated pull request descriptions by analyzing 23,247 agentic PRs, finding that 1.7% exhibited high message-code inconsistency (PR-MCI)—where descriptions misalign with actual code changes. Their analysis revealed that high-MCI pull requests had 51.7% lower acceptance rates and took 3.5× longer to merge, highlighting that even when the generated code is acceptable, unreliable descriptions can undermine trust and review efficiency.

These findings underscore both the promise and the current limitations of AI coding agents: while they can significantly accelerate development workflows, their effectiveness varies considerably across task types, and the trustworthiness of their outputs—including both code and natural-language descriptions—remains an open challenge. This thesis contributes to this research direction particularly in Chapters 6 and 7, where we investigate the sustainable deployment and optimization of such agents.

2.4 Evaluation of Large Language Models

Evaluating LLMs presents unique difficulties compared to traditional software testing due to the stochastic nature of text generation. Natural language exhibits a one-to-many mapping where multiple valid outputs exist for a single input, and the semantic-syntactic gap makes it difficult to verify if a response conveys the correct meaning despite differing surface forms. Effective benchmarking aims to balance functional correctness with generalization and computational efficiency. However, a critical challenge is test insufficiency—limited test cases in standard benchmarks often fail to catch edge cases, leading to overestimated performance metrics. While traditional

metrics exist, semantic approaches like $BERT_{score}$ [168] leverage pre-trained contextual embeddings to measure similarity, offering better correlation with human judgment than simple n-gram matching, though they still struggle to guarantee functional equivalence in code generation [104].

2.4.1 Coding Benchmarks

To assess code generation, specific benchmarks have been established. HumanEval provides 164 hand-crafted Python problems with accompanying test suites and function signatures [23]. The benchmark's $pass@k$ metric measures the probability that at least one of k generated samples passes all tests, providing insight into both accuracy and consistency. However, HumanEval is limited by low test coverage, which can mask semantic errors. To address this, HumanEval+ expands test suites significantly, revealing that models often fail on edge cases previously ignored and altering model rankings based on robustness [104].

For more complex scenarios requiring complete programs with complex control flow, the PSB2 benchmark suite [62] offers 25 diverse programming problems spanning string manipulation, numerical computation, list processing, and logical reasoning. Each problem includes comprehensive test suites with 1000 training cases and 1000 testing cases, enabling robust assessment of generalization beyond training data. Originally designed for GP, PSB2 has become essential for robust LLM testing due to its extensive coverage and emphasis on algorithmic challenges that single-shot generation struggles to address.

Additional benchmarks like MBPP, APPS, and CodeContests focus on varying difficulty levels ranging from basic scripting to competitive programming, aiming to evaluate different aspects of algorithmic reasoning [8, 66].

2.4.2 Challenges in Code Evaluation

Evaluating generated code involves distinct hurdles beyond simple text matching. Insufficient test coverage allows incorrect code to pass, while increasing test quantity often drastically drops scores, demonstrating high sensitivity to

test suite quality. Standard metrics like statement or branch coverage are often inadequate due to the combinatorial explosion of execution paths in complex logic. Furthermore, code might pass tests but fail user intent due to ambiguous natural language requirements, or lack non-functional quality attributes like readability and algorithmic efficiency. Finally, rigorous evaluation incurs high computational costs due to the need for sandboxed execution of thousands of samples to calculate reliable pass rates.

2.4.3 SQL Query Generation Evaluation

SQL generation introduces the challenge of declarative equivalence, where syntactically different queries yield identical results. The *Spider* benchmark addresses this by testing generalization across disjoint domains, ensuring models do not simply memorize schema structures [162]. Evaluation typically relies on *EX*, which compares the output of queries run on actual databases, providing a functional correctness check. Alternatively, *EM* compares string structure but is often too strict against valid syntactic variations. Recent iterations like *Spider 2.0* extend this to enterprise-level complexity and dialect diversity.

2.4.4 Optimization and GI

To mitigate LLM limitations, GI can be integrated to iteratively refine generated code using evolutionary algorithms and test cases, discovering correct solutions that simple re-prompting misses by exploring the solution space systematically [121]. Similarly, techniques like MOO apply algorithms like NSGA-II to find non-dominated configurations for AI agents, balancing correctness, cost, and runtime efficiency [51]. These optimization strategies extend to specific tasks like feature selection for bug taxonomy, demonstrating that evolutionary methods can significantly enhance the reliability and efficiency of LLM-based systems without requiring larger models [33].

2.4.5 Cross-Cutting Considerations

Broader evaluation must address data contamination, where models memorize training data, requiring strict decontamination protocols such as time-based splits. While human evaluation remains the gold standard for subjective qualities like maintainability and documentation, it is unscalable and costly. Consequently, the field is moving towards multi-dimensional frameworks that can assess functional correctness, runtime efficiency, and code quality simultaneously, reflecting real-world deployment requirements where the LLM's output must be not only correct but also efficient.

2.5 Conclusion

This chapter has established the theoretical foundations underlying the integration of DL, transformer architectures, and evolutionary computation in automated SE. The progression from traditional DL through transformers to GP illustrates a trajectory toward increasingly sophisticated approaches to program synthesis and improvement. The limitations of sequential architectures in DL motivated the development of transformers, which achieve superior parallelization and long-range dependency modeling through attention mechanisms. Similarly, the semantic incorrectness often exhibited by transformer-based LM motivates the application of GI techniques, creating a complementary relationship between neural and evolutionary approaches.

The synthesis of these paradigms—leveraging LLMs for initial code generation followed by evolutionary refinement—represents a promising direction for automated programming systems [121]. This hybrid approach combines the broad knowledge encoded in LM with the targeted optimization capabilities of genetic algorithms, addressing the respective limitations of each methodology.

I

Output Quality and Improvement

3

Evaluating LLMs for Legal Text Summarization

This chapter investigates the potential of LLMs to democratize access to legal knowledge by evaluating their effectiveness in summarizing complex legal judgments. We present a comprehensive human-centered evaluation involving 75 participants with varying levels of legal expertise, comparing LLM-generated summaries against expert-crafted **Massime** and original court judgments. This investigation directly addresses the thesis research questions RQ1 (evaluation methodologies) and RQ2 (democratization of specialized knowledge) within the legal domain.

Authors: *Giovanni Pinna*, Davide Tugnoli, Mara Bartole, Luca Manzoni, Andrea De Lorenzo

Title: From Courts to Comprehension: Can LLMs Make Judgments More Accessible?

Published in: 2024 IEEE/WIC International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT). IEEE, 2024. p. 297-304.

<https://ieeexplore.ieee.org/document/10973499>

3.1 Introduction

Legal regulations shape everyday life in democratic societies, governing social, economic, and political interactions. However, a critical paradox emerges, legal texts are meant to govern and guide public behavior, and their inherent complexity renders them largely inaccessible to those without specialized legal training [175]. This accessibility gap represents a significant barrier to legal literacy and citizen empowerment, potentially undermining the democratic principle that ignorance of the law is no excuse. This challenge is closely tied to the inherent complexity of legal language, which arises from the use of highly specialized terminology, complex syntactic constructions, extensive internal cross-referencing, and the need for a level of precision that often prioritizes legal accuracy over communicative clarity [2]. Traditional approaches to bridging this comprehension gap have relied heavily on human expertise—legal professionals who can interpret and explain complex judgments to lay audiences. In the Italian legal system, this function is partially served by *massime* (singular: *massima*), concise summaries that distill the essential legal principles from court decisions. However, the creation of **Massime** requires substantial human resources and specialized expertise, making it an expensive and time-consuming process that cannot scale to meet the growing volume of legal documentation.

Recent advances in NLP, particularly the emergence of LLM, present a promising avenue for democratizing access to legal information. These models have demonstrated remarkable capabilities in understanding and generating human-like text across various domains [147, 119]. The potential application of LLMs to legal text summarization could change how citizens interact with legal documents, transforming impenetrable texts into comprehensible summaries.

This chapter contributes to the thesis research questions **RQ1** on evaluation methodologies and **RQ2** on the democratization of specialized knowledge by examining the viability and effectiveness of LLM-based legal summarization through a set of chapter-specific research questions. These questions collectively investigate how such models can be assessed, how reliably they perform in the legal domain, and to what extent they can support broader access to complex legal information.

c3RQ1: To what extent can LLM-generated summaries of legal judgments enhance comprehension for individuals without legal expertise compared to reading original judgments?

c3RQ2: How do AI-generated summaries compare to human-crafted **Massime** in terms

3. LLMs FOR LEGAL TEXTS

of accuracy, comprehensibility, and potential for misrepresentation of legal content?

c3RQ3: What are the trade-offs between different LLM architectures (open-source versus proprietary, fine-tuned versus general-purpose) in generating accessible legal summaries?

3

These chapter-specific research questions directly address the thesis-level questions: c3RQ1 and c3RQ2 contribute to RQ2 by empirically assessing the democratization potential and risks of LLMs in the legal domain, while all three questions contribute to RQ1 by developing and applying human-centered evaluation methodologies that go beyond surface-level metrics.

These research questions are critical because they collectively address the core challenges of legal accessibility and implementation. By examining whether c3RQ1 can be answered affirmatively, the study directly engages with the fundamental objective of determining whether LLMs are capable of genuinely improving the comprehensibility of legal texts, a prerequisite for their meaningful use in democratic societies. At the same time, investigating c3RQ2 sheds light on the quality-cost trade-off between automated and human-generated summaries, an issue with far-reaching consequences for how resources are allocated within legal systems. Finally, the focus of c3RQ3 on practical implementation highlights the need to balance performance with accessibility when choosing among different model architectures, ensuring that technical advances can be translated into usable and scalable solutions.

Our approach to answering these questions employs a rigorous human-centered evaluation methodology. Rather than relying solely on automatic metrics, which have demonstrated significant limitations in capturing semantic accuracy and comprehensibility [20], we conduct an extensive evaluation involving 75 participants with varying levels of legal knowledge. Participants are presented with legal texts—original judgments, *Massime*, and AI-generated summaries—and assessed through comprehension questions designed by legal experts. This extrinsic evaluation method measures not the surface quality of summaries but their practical utility in facilitating understanding.

The evaluation of LLMs presents unique challenges that underscore the importance of human assessment. Traditional automatic metrics such as ROUGE [100], BLEU [118], and METEOR [10] primarily measure surface-level similarity between generated and reference texts. However, these metrics fail to capture critical aspects of legal summarization: factual accuracy, preservation of legal nuance, and genuine comprehensibility for non-experts. A summary might achieve high ROUGE scores while containing factual er-

rors or legal misrepresentations that could mislead readers. Conversely, a summary with lower automatic scores might effectively convey the essential legal principles in accessible language.

The significance of this work extends beyond technical advancement in NLP. Successfully democratizing access to legal information could transform citizen engagement with the legal system, enabling individuals to better understand their rights and obligations. This has profound implications for access to justice, legal literacy, and the functioning of democratic institutions. Moreover, our findings contribute to the broader discourse on responsible AI deployment in high-stakes domains, where the consequences of errors or misrepresentations can significantly impact individuals' lives.

3.2 Related Works

The intersection of AI and legal text processing has evolved from early theoretical frameworks [17] through traditional extractive and abstractive methods [101, 163] to the current paradigm dominated by LLMs [147]. While recent literature explores the application of these models in various legal tasks—ranging from contract review [108] to the summarization of court rulings in different jurisdictions [2, 38]—critical limitations persist regarding hallucination risks and the inadequacy of automatic metrics for assessing legal accuracy. Distinct from prior studies that rely primarily on automatic scoring or expert-centric validation, this work addresses the specific challenges of the Italian civil law context by comparing open-source and proprietary models through a human-centered evaluation focused on layperson comprehension.

3.3 Methodology

Our methodology employs a comprehensive evaluation framework designed to assess the effectiveness of LLMs in generating comprehensible legal summaries. The approach centers on human evaluation through comprehension assessment, providing direct insights into the practical utility of different summarization methods.

3.3.1 Evaluation Framework Design

The core of our methodology involves a questionnaire-based assessment where participants interact with four distinct text types without prior knowledge of their categories.

3. LLMS FOR LEGAL TEXTS

Each participant receives: (i) one complete original judgment, (ii) one expert-crafted **Massima**, (iii) one summary generated by an open-source LLM, and (iv) one summary generated by GPT-4o. This blind presentation prevents bias based on preconceptions about text sources. For each text, participants answer three carefully constructed multiple-choice questions, each offering five response options including an explicit "I don't know" choice. This option proves crucial for distinguishing between confident incorrect responses and acknowledged uncertainty—a distinction particularly important when evaluating potential misinformation risks in legal contexts. The questions undergo rigorous development by legal experts who access only the original judgments, deliberately avoiding exposure to **Massime** or generated summaries to prevent inadvertent bias toward particular summary types. This methodological choice ensures questions assess comprehension of essential legal content rather than artifacts of specific summarization approaches.

3

3.3.2 Addressing Inherent Evaluation Challenges

Our approach acknowledges fundamental limitations in summary evaluation. Model-generated summaries may omit critical information present in original texts due to compression algorithms or context window limitations. **Massime**, despite their expert craftsmanship, focus on extracting specific legal principles rather than providing comprehensive case summaries. A **Massima** encapsulates the principle of law established by a judgment, offering legal professionals a concise view of applicable legal principles. Consequently, **Massime** may not contain all information necessary to answer every comprehension question. This limitation transforms into a methodological strength: by including questions potentially unanswerable from summaries alone, we assess not only what information summaries successfully convey but also what they omit. This approach reveals the completeness and reliability of different summarization methods.

3.3.3 Selection Rationale for **Massime** as Benchmark

Despite being designed for audiences that differ from our study population, **Massime** serve as an appropriate benchmark for several compelling reasons. They constitute the current gold standard for legal summarization within the Italian legal system, as they are produced by specialized offices with substantial legal expertise. Their widespread availability further makes them suitable for large-scale comparison, enabling systematic evaluation across a broad set of cases. Moreover, as human-generated summaries, they of-

for a meaningful baseline against which the performance of AI-based approaches can be assessed, even if they are not perfectly aligned with the accessibility objectives that guide this study. The creation of **Massime** requires extensive analysis to extract relevant legal principles—a resource-intensive process that highlights the potential value of automated alternatives. By comparing LLM outputs against these expert summaries, we establish whether AI can approach or exceed human performance while potentially serving broader audiences.

3.3.4 Participant Selection and Stratification

Our study design incorporates 75 voluntary participants stratified by legal knowledge. Approximately 25% report some legal background, while 75% identify as having no legal expertise. This distribution reflects our primary target audience—individuals without specialized legal training—while maintaining sufficient legal expertise representation to identify differential effects across knowledge levels. We acknowledge that additional participant characteristics, such as prior familiarity with LLMs and general technical literacy, were not collected in the initial survey and represent a limitation of the current study design. Prior exposure to LLM-generated text could potentially influence participants' trust in and critical evaluation of AI-generated summaries, and future work should incorporate such background variables to enable more nuanced analysis of how technological familiarity interacts with comprehension outcomes. The stratification enables crucial analyses: examining whether legal knowledge moderates the effectiveness of different summary types, identifying which approaches best serve non-expert audiences, and understanding how summary characteristics interact with reader background to influence comprehension outcomes.

3.3.5 Text Presentation and Randomization

To minimize order effects and ensure robust results, we implement complete randomization of text presentation. Each participant encounters the four text types in random order, with each text corresponding to a different judgment to maintain question independence. This design prevents learning effects where exposure to one text might influence comprehension of subsequent related content. Participants retain access to texts while answering questions, simulating realistic usage scenarios where readers can reference documents during comprehension tasks. This approach measures not memory but the ability to extract and understand information from available texts—the primary

3. LLMS FOR LEGAL TEXTS

use case for legal document accessibility tools.

3.3.6 Web-Based Data Collection Platform

We developed a custom web application to facilitate standardized data collection. The platform begins with an introductory guide explaining the study purpose and procedures, followed by an initial survey collecting demographic information, educational background, and self-assessed legal knowledge. The interface presents texts and questions sequentially, allowing participants to skip questions if needed—a feature that provides valuable data about text accessibility. When participants abandon question sets entirely, it signals potential issues with text length, complexity, or clarity that mere incorrect answers might not reveal.

3.3.7 Temporal and Content Considerations

From our corpus of 10000 judgment-*Massima* pairs, we carefully selected 25 pairs spanning 2000 to 2023, ensuring temporal representation while avoiding obsolete legal terminology that might confound comprehension assessment. This temporal distribution captures evolution in legal language and concepts while maintaining relevance for contemporary readers. The selection process prioritized judgments addressing diverse legal topics, preventing domain-specific biases that might favor particular summarization approaches. Constitutional Court decisions, with their broad societal implications and varied subject matter, provide ideal test cases for assessing summarization effectiveness across legal domains. Concrete examples of the materials used in the evaluation—including an excerpt of an original judgment, the corresponding expert-crafted *massima*, LLM-generated summaries, and sample comprehension questions—are provided in Appendix A.

3.4 Experimental Evaluation

The experimental phase implements a systematic evaluation of multiple LLM architectures and FT strategies, carefully designed to identify optimal approaches for legal text summarization.

Our experimental design incorporates diverse model architectures to comprehensively assess the current landscape of LLM capabilities. We selected five models spanning different design philosophies, parameter scales, and deployment contexts:

Gemma-2B-it* and **Gemma-7B-it[†]** [47] (Google’s instruction-tuned variants with 2 and 7 billion parameters respectively), **LLaMA-2-7B-chat** [148][‡] (L2, Meta’s 7-billion parameter chat-optimized model), **LLaMantino-7B** [11][§] (a 7-billion parameter Italian-adapted variant of LLaMA 2), and **GPT-4o[¶]** (OpenAI’s proprietary multimodal model accessed via API, with an estimated scale of hundreds of billions of parameters, though the exact architecture is undisclosed). The selection criteria prioritized: (i) open-source models deployable on consumer hardware, ensuring accessibility for resource-constrained environments; (ii) models with demonstrated multilingual capabilities, particularly for Italian text processing; (iii) architectural diversity to identify design patterns conducive to legal summarization; and (iv) inclusion of state-of-the-art proprietary models (GPT-4o) as performance benchmarks.

3.4.1 Fine-Tuning Methodology

Recognizing that open-source models lack specialized training for Italian legal text, we implemented domain-specific FT using *Massime* as training data. The FT process utilized 9,500 judgment-*Massima* pairs (95% of our corpus), reserving 500 pairs for evaluation. Training proceeded on the Leonardo supercomputer[¶] using four NVIDIA A100 GPUs with 64GB VRAM each. We fine-tuned each model for three epochs using the Paged AdamW optimizer [107] with learning rate 0.0002 and weight decay 0.03. These hyperparameters balanced training efficiency with model stability, preventing overfitting while enabling meaningful adaptation to legal language patterns. The open-source models range from 2 to 7 billion parameters, enabling deployment on consumer-grade hardware with a single GPU. Fine-tuning each model on the Leonardo supercomputer required approximately 4–6 hours of wall-clock time using the four A100 GPUs. For GPT-4o, no fine-tuning was performed; the model was accessed via OpenAI’s API at the time of the experiment (mid-2024). The API cost for generating summaries of all 25 test judgments with GPT-4o was negligible (under \$5 USD at the then-current pricing), making it an accessible option for practitioners. In contrast, the open-source models, once fine-tuned, can be deployed at zero marginal cost on consumer hardware, representing a significant advantage for resource-constrained legal institutions.

*<https://huggingface.co/google/gemma-2b-it>

[†]<https://huggingface.co/google/gemma-7b-it>

[‡]<https://huggingface.co/meta-llama/Llama-2-7b-chat-hf>

[§]<https://huggingface.co/swap-uniba/LLaMantino-2-7b-hf-UltraChat-ITA>

[¶]Version of 13 May 2024

[¶]<https://www.hpc.cineca.it/systems/hardware/leonardo/>

3. LLMs FOR LEGAL TEXTS

Model-specific instruction templates, detailed in Table 3.1 of the original paper, ensure optimal instruction following. For LLaMA-based models, we employed Italian-translated system prompts, observing improved instruction adherence and Italian language generation compared to English prompts. Gemma models received specialized legal expert prompts emphasizing their role in legal text analysis and summarization.

3

Table 3.1: Instruction tags used for FT and inference of LLMs. Each model family requires specific formatting to ensure proper instruction following.

Model	Prompt Template
Gemma	<code><start_of_turn>user\n {request} <end_of_turn>\n<start_of_turn>model\n</code>
LLaMA 2	<code><s>[INST] <<SYS>>\n {System Prompt} \n<</SYS>>\n\n {User Message} [/INST]</code>
LLaMantino	<code><s>[INST] <<SYS>>\n {System Prompt} \n<</SYS>>\n\n {User Message} [/INST]</code>

A critical experimental consideration involves managing varying context window limitations across models. LLaMA 2's 4096-token window proved insufficient for 21 of 25 test judgments, necessitating truncation that potentially omits crucial information. Gemma's 8192-token window accommodated more content but still required truncation for 10 judgments. GPT-4o's 128000-token window eliminated truncation concerns, processing all judgments completely. This disparity introduces an important experimental variable: the relationship between context window size and summarization quality. Truncation may force models to summarize incomplete information, potentially affecting comprehension outcomes and highlighting a fundamental limitation of current open-source models for legal applications.

3.4.2 Qualitative Model Assessment

Initial qualitative evaluation revealed significant performance variations across models. Gemma models (both 2B and 7B variants) demonstrated poor Italian language handling, frequently generating responses containing HTML tags or failing to address queries appropriately in both base and FT versions. This limitation disqualified them from further evaluation despite their architectural advantages. Base LLaMA 2 consistently generated English responses despite Italian prompting, indicating insufficient multilingual train-

ing. However, fine-tuned LLaMA 2 (which from now on we will call FT-LLaMA 2) showed marked improvement in Italian legal text generation, producing coherent summaries that warranted inclusion in human evaluation. LLaMantino, while demonstrating comparable performance to FT-LLaMA 2, occasionally included system prompt artifacts in outputs, compromising response quality. Based on these observations, FT-LLaMA 2 emerged as the optimal open-source representative for comparative evaluation against GPT-4o.

Summary generation employed carefully crafted prompts optimized through iterative testing. For FT-LLaMA 2 and LLaMantino, we utilized: "Rispondi in italiano e fai il riassunto del seguente testo" (Respond in Italian and summarize the following text). This direct instruction minimizes ambiguity while maintaining focus on the summarization task. Gemma models received more elaborate prompts positioning them as legal experts: "Sei un modello esperto di leggi, sentenze e ordinanze italiane. Di seguito è riportata una sentenza o una legge italiana, il tuo compito è quello di riassumerla" (You are an expert model in Italian laws, judgments, and ordinances. Below is an Italian judgment or law, your task is to summarize it). GPT-4o received similar instructions through its API, maintaining consistency in task specification while leveraging its superior context handling and language capabilities.

3.4.3 Human Evaluation

The human evaluation component represents the cornerstone of our experimental methodology, directly measuring comprehension outcomes rather than surface-level text characteristics. The evaluation protocol implements several key design decisions that enhance validity and reliability. Participants interact with our custom web application, which tracks not only response accuracy but also engagement metrics, including reading time and question completion rates. These additional metrics provide insights into text accessibility beyond mere comprehension accuracy. The web platform records timestamps for each interaction, enabling analysis of reading speeds and response times across different text types. Extended reading times or high skip rates signal accessibility barriers that correct answer percentages alone might not reveal.

Our analysis employs rigorous statistical methods to identify significant differences across experimental conditions. The Chi-squared test [30] evaluates overall response pattern differences between participants with and without legal knowledge (p -value = 4×10^{-5} , indicating highly significant differences). Pairwise comparisons between text

types utilize the Chi-squared test with Bonferroni-Holm correction [68] to control for multiple comparisons. This conservative approach ensures identified differences represent genuine effects rather than statistical artifacts. The analysis examines multiple outcome dimensions: correct answer percentages, incorrect answer rates (excluding "I don't know" responses), uncertainty rates ("I don't know" responses), and question skip rates. This multifaceted approach reveals nuanced differences in how various summary types support comprehension.

Throughout the experimental process, we implemented multiple quality assurance measures. Legal experts validated comprehension questions for clarity and relevance. Technical validation ensured proper text display and response recording. Pilot testing with a small participant subset identified and resolved interface issues before full deployment. The emphasis on human evaluation addresses fundamental limitations of automatic metrics in legal contexts. Human evaluation directly measures these dimensions through comprehension assessment, providing ecologically valid insights into summary utility. This approach acknowledges that a legally accurate but incomprehensible summary fails our fundamental goal of democratizing legal information access. An example set of comprehension questions posed to participants, along with the corresponding answer options and correct answers, is available in Appendix A.1**.

3.5 Results

Our experimental evaluation yields comprehensive insights into the effectiveness of different summarization approaches across multiple evaluation dimensions.

3.5.1 Correct Answer Analysis

Figure 3.1 demonstrates clear performance hierarchies across text types. *Massime* achieve the highest overall correct answer rate at 45%, rising to 58% for participants with legal knowledge—the only condition exceeding 50% accuracy. This 17% advantage for legally knowledgeable participants represents the largest expertise effect observed across all conditions.

GPT-4o summaries achieve 38% overall accuracy, with marginal improvements for participants with legal knowledge (39%) versus without (38%). This minimal expertise effect suggests GPT-4o generates summaries accessible across knowledge levels, though

** All the questions and the web application used for data collection are also available in the online repository.

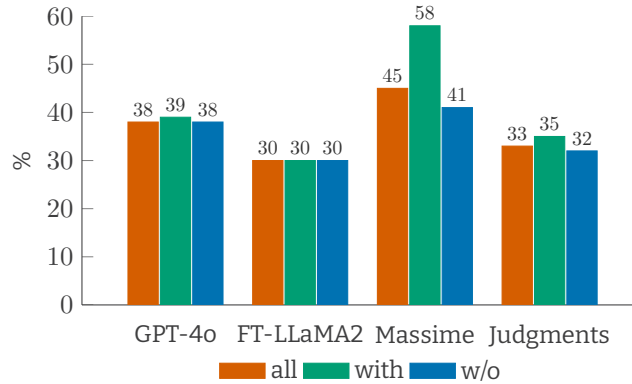


Figure 3.1: Percentage of correct answers for each type of text for all participants (*all*), for participants with legal knowledge (*with*), and for people without legal knowledge (*w/o*)

at lower absolute accuracy than *Massime*. Original judgments yield 33% accuracy overall, with similar performance for participants with (35%) and without (32%) legal knowledge. This 3% difference indicates that raw legal texts remain challenging regardless of background knowledge, suggesting that complexity extends beyond mere terminology to fundamental structural and conceptual barriers. The summaries of the model FT-LLaMA 2 perform poorest at 30% overall accuracy, showing no meaningful difference between participants with and without legal knowledge. This uniformly poor performance indicates fundamental limitations in the model's summarization capabilities.

Skip rates in Figure 3.3 correlate strongly with text length. Original judgments show the highest skip rate at 24% overall, rising to 32% for participants without legal knowledge. Summaries from FT-LLaMA 2 exhibit similarly high skip rates despite shorter length, suggesting that poor quality discourages engagement regardless of brevity.

3.5.2 Length and Engagement Analysis

Text length analysis in Figure 3.4 reveals dramatic variations across categories. Original judgments average over 10000 words, while *massime* and GPT-4o summaries typically remain below 1000 words, an order of magnitude difference. The FT-LLaMA 2 summaries occupy an intermediate position, typically ranging from 1000 to 3000 words.

Reading time analysis in Figure 3.5 demonstrates the practical implications of length differences. *Massime* and GPT-4o summaries show similar reading time distributions, typically under 10 minutes. Original judgments and FT-LLaMA 2 summaries require

3. LLMS FOR LEGAL TEXTS

3

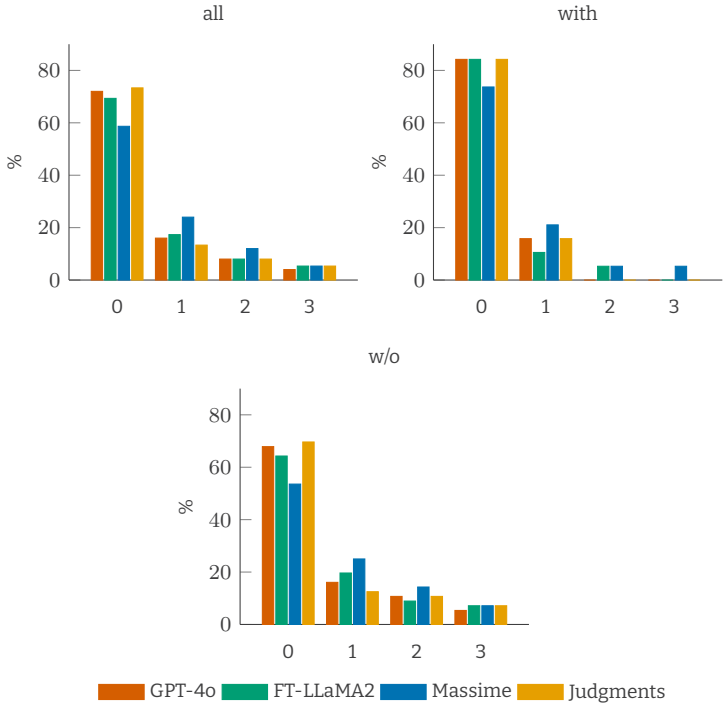


Figure 3.2: Distribution of the number of I don't know answers according to the type of text for all participants (top left), participants with preexisting legal knowledge (top right), and participants without legal knowledge (bottom).

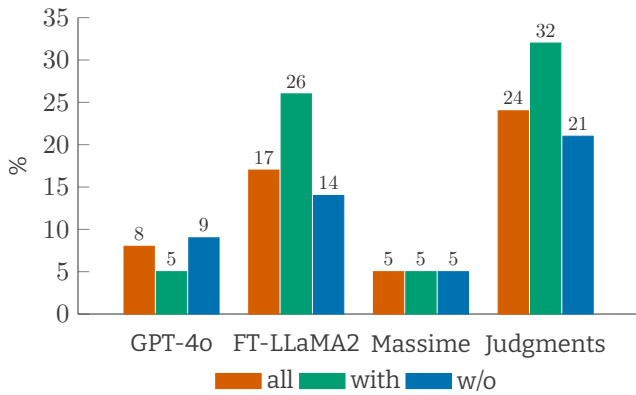


Figure 3.3: Percentage of skipped answers for each type of text for all participants (all), for participants with legal knowledge (with), and for people without legal knowledge (w/o)

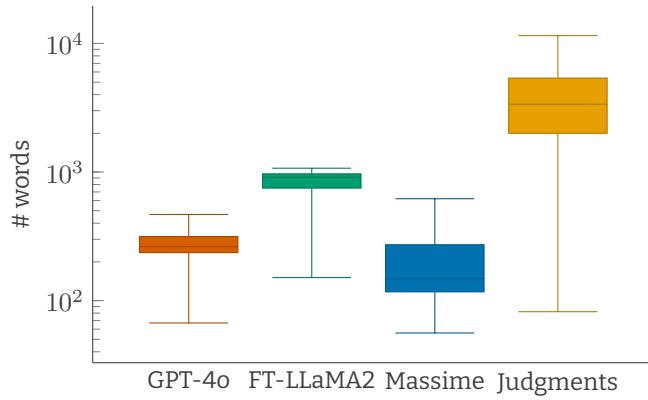


Figure 3.4: Boxplots of the length (in number of words) of the texts for each category. The boxes show the 25-th and 75-th percentiles, while the whiskers denote the minimum and maximum. Notice that the vertical axis is in *logarithmic* scale.

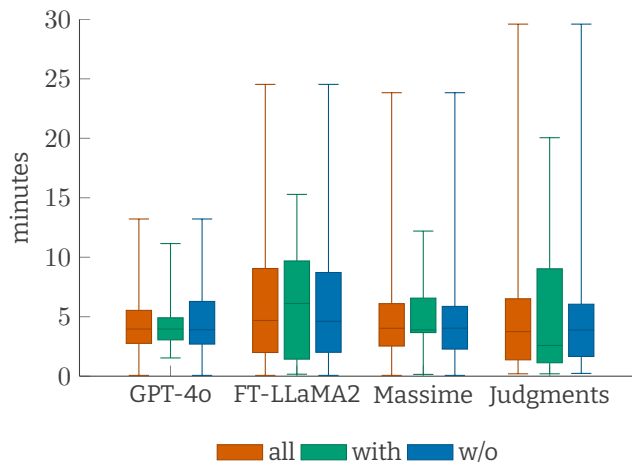


Figure 3.5: Boxplots showing the reading and response time for each type of text and the different categories of participants. The boxes show the 25-th and 75-th percentiles, while the whiskers denote the minimum and maximum.

substantially longer engagement, often exceeding 20 minutes. Importantly, participants with legal knowledge invest more time reading **Massime**, suggesting deeper engagement with these expert-crafted summaries. The correlation between text length and skip rate reveals a critical threshold: texts exceeding approximately 3000 words experience dramatically increased abandonment rates. This finding has important implications for summary design, suggesting that effectiveness requires not just accuracy but also appropriate conciseness.

3.5.3 Statistical Significance Analysis

Chi-squared testing with Bonferroni-Holm correction identifies statistically significant differences between text type pairs. For participants with legal knowledge, significant differences emerge between: GPT-4o summaries and original judgments ($p = 0.011$), **Massime** and original judgments ($p = 0.011$), and **Massime** and FT-LLaMA 2 summaries ($p = 0.011$). For participants without legal knowledge, the same pairs show significant differences with slightly varied p-values: GPT-4o summaries versus original judgments ($p = 0.038$), **Massime** versus original judgments ($p = 0.001$), and **Massime** versus FT-LLaMA 2 summaries ($p = 0.038$).

The comparison between GPT-4o and FT-LLaMA 2 summaries approaches but does not reach significance for legally knowledgeable participants ($p = 0.054$), suggesting these approaches may produce similarly problematic outcomes through different mechanisms: GPT-4o through potential misinformation, FT-LLaMA 2 through inadequate information extraction.

Synthesizing across metrics reveals nuanced performance profiles. **Massime** excel in accuracy and minimize errors but generate higher uncertainty, reflecting their focused extraction of legal principles. Their brevity and expert craftsmanship produce the best overall comprehension outcomes, particularly for legally knowledgeable readers. GPT-4o summaries offer accessibility through brevity and generate moderate comprehension rates. However, their high error rates raise serious concerns about reliability in legal contexts where misinformation carries significant consequences. The model appears to prioritize fluency and conciseness over factual accuracy. Original judgments and FT-LLaMA 2 summaries both suffer from excessive length that discourages engagement. While original judgments maintain accuracy when read completely, their complexity and length create insurmountable barriers for most readers. FT-LLaMA 2 combines the worst aspects: insufficient quality improvement over original texts while maintaining prohibitive length.

3.6 Discussion

The evaluation reveals a fundamental tension in LLM-based legal summarization: accessibility and accuracy work against each other.

GPT-4o improves comprehension for laypeople by 6 percentage points over original judgments—a statistically significant gain. However, this improvement stems from length reduction, not conceptual simplification. Critically, this accessibility comes at the cost of accuracy: GPT-4o summaries inadvertently increase the rate of incorrect answers, suggesting the model relies on oversimplification that misrepresents legal nuances. Conversely, smaller open-source models like FT-LLaMA 2 fail to enhance comprehension entirely, often performing worse than the original texts due to limited context windows and insufficient reasoning capacity, resulting in verbose and truncated outputs.

c3RQ1 Answer

GPT-4o achieves a modest 6 percentage point improvement in comprehension for laypeople compared to original judgments, primarily through length reduction rather than conceptual simplification. However, this accessibility gain comes at the cost of increased error rates, indicating that the model relies on oversimplification that misrepresents legal nuances. Open-source models like FT-LLaMA 2 fail to enhance comprehension entirely due to context window limitations.

Expert-crafted **Massime** outperform all AI alternatives. Their superiority—45–58% accuracy versus 38% for GPT-4o—reflects deeper legal understanding that current models cannot replicate. While GPT-4o offers speed and drastic cost reductions, it exhibits a concerning accuracy-accessibility trade-off: its summaries are fluent but prone to “hallucinations” leading to high error rates among legally knowledgeable participants who may be misled by plausible-sounding inaccuracies. This poses significant risks for legal applications where misinformation can have serious consequences, suggesting that current AI models cannot yet replace expert reliability despite their efficiency.

c3RQ2 Answer

Human-crafted summaries significantly outperform AI-generated summaries, achieving 45–58% accuracy compared to 38% for GPT-4o. While GPT-4o offers speed and cost reductions, it exhibits a concerning accuracy-accessibility trade-off: fluent summaries prone to hallucinations lead to higher error rates among legally knowledgeable participants (inverse expertise-error relationship). Current AI models cannot replace expert reliability in high-stakes legal contexts.

Proprietary models like GPT-4o dominate open-source alternatives. Two factors explain this gap: larger parameter counts and context windows of 128k tokens that avoid truncation of lengthy legal documents. Open-source alternatives, despite FT on Italian legal texts, struggle with structural limitations and are unable to balance completeness with conciseness. The results suggest a “threshold effect” where models below a certain capacity (e.g., 7B parameters) are inadequate for complex legal reasoning regardless of training data. Consequently, the answer to item c3RQ3 is that the deployment decisions must weigh the superior performance of proprietary models against the transparency and privacy benefits of open-source architectures.

c3RQ3 Answer

Proprietary models (GPT-4o) hold a decisive advantage due to extensive parameters and large context windows (~128k tokens) preventing information truncation. Open-source alternatives struggle with structural limitations despite fine-tuning on Italian legal texts. A “threshold effect” emerges where models below ~7B parameters are inadequate for complex legal reasoning regardless of training data. Deployment decisions must weigh superior proprietary performance against open-source transparency and privacy benefits.

Analysis revealed counter-intuitive patterns, notably an inverse relationship between legal expertise and error detection in GPT-4o summaries, suggesting that high fluency masks subtle inaccuracies that entrap even knowledgeable readers. Furthermore, the data indicate a bimodal distribution in comprehension, users either understand well or not at all, and a refusal to engage with poor-quality summaries (like those from FT-LLaMA 2) even when short. This implies that brevity is insufficient without a minimum threshold of clarity and accuracy.

3.6.1 Limitations and Future Directions

We acknowledge limitations regarding our focus on Italian Constitutional Court judgments, which may not fully generalize to other civil law domains or common law systems, and potential selection biases in our participant pool. Future research should expand to diverse legal document types, cross-jurisdictional comparisons, and longitudinal studies on information retention. Technically, efforts must focus on developing legal-specific architectures with better context management and fact-checking mechanisms, moving toward hybrid approaches that combine AI scalability with necessary human oversight.

3.7 Conclusions

This study confirms that while LLMs represent a promising avenue for democratizing legal information, they currently fall short of the “gold standard” set by human-expert summaries (**Massime**) in terms of accuracy and reliability. Our results establish a clear hierarchy: human experts excel at balancing precision with accessibility; GPT-4o offers a scalable but risky alternative, prone to generating plausible but inaccurate content that poses significant dangers in legal contexts; and open-source models, such as FT-LLaMA 2, remain inadequate due to architectural limitations such as context window constraints, failing to provide practical utility despite fine-tuning efforts.

Beyond technical benchmarks, this research highlights critical methodological and societal implications. We demonstrate that human-centered evaluation is essential in high-stakes domains, as automatic metrics fail to detect subtle hallucinations that can mislead even knowledgeable users. Future progress lies not in mere parameter scaling, but in the development of hybrid systems combining AI efficiency with human oversight and purpose-built legal architectures. Ultimately, achieving genuine access to justice requires prioritizing human understanding over algorithmic speed, ensuring that the deployment of AI in the legal sphere supports rather than compromises the rule of law.

4

Improving LLM Code Generation by Leveraging Genetic Improvement

This chapter presents a hybrid neuro-evolutionary approach that enhances the functional correctness of LLM-generated code through GI. By combining the generative capabilities of LLMs with GE using dynamically specialized grammars and test-driven fitness evaluation, we demonstrate that evolutionary refinement is able to improve in a statistically significant way the code generated by LLMs across diverse programming problems. This investigation directly addresses the thesis research questions RQ2 (democratization of specialized knowledge in the SE domain) and RQ3 (post-processing and hybrid approaches for reliability enhancement), establishing that systematic evolutionary search can compensate for the semantic limitations inherent in current code generation systems.

4. LLMGIPY

4

Authors: *Giovanni Pinna*, Damiano Ravalico, Luigi Rovito, Luca Manzoni, Andrea De Lorenzo

Title: Enhancing large language models-based code generation by leveraging genetic improvement.

Published in: European Conference on Genetic Programming (Part of EvoStar). Cham: Springer Nature Switzerland, 2024. p. 108-124.

https://link.springer.com/chapter/10.1007/978-3-031-56957-9_7

Authors: *Giovanni Pinna*, Damiano Ravalico, Luigi Rovito, Luca Manzoni, Andrea De Lorenzo

Title: Exploring the Effect of Genetic Improvement for Large Language Models-Generated Code

Published in: SN Computer Science, 2025, 6.7: 1-21.

<https://link.springer.com/article/10.1007/s42979-025-04281-x>

Authors: *Giovanni Pinna*, Damiano Ravalico, Luigi Rovito, Luca Manzoni, Andrea De Lorenzo

Title: Improving LLM-Generated Code via Genetic Improvement: A Summary of Recent Advances

Published in: Ital-IA, Workshop on Generative AI: Generative AI and Education

<https://ceur-ws.org/Vol-4121/>

4.1 Introduction

The automatic synthesis of source code from natural language specifications represents a fundamental challenge in computer science, with implications spanning SE, AI, and computational linguistics. This chapter addresses a critical limitation in contemporary code generation: while LLMs have demonstrated remarkable capabilities in producing syntactically valid code from textual descriptions, they frequently fail to generate semantically correct solutions that satisfy functional requirements.

The problem we address is the persistent inaccuracy in LLM-generated code, particularly when confronted with complex programming tasks or ambiguous specifications. Despite significant advances in neural architectures and training methodologies, even state-of-the-art models such as GPT-4, Claude, and LLaMA struggle to produce functionally correct code consistently [23]. This limitation is especially pronounced for programming languages underrepresented in training corpora and for problems requiring precise handling of edge cases. The resolution of this problem carries substantial implications for **software development (SD)** productivity. Accurate automatic code generation would fundamentally transform how software is created, enabling developers to focus on high-level design while delegating implementation details to automated systems. Furthermore, reliable code synthesis would democratize programming, making software creation accessible to domain experts without extensive programming expertise.

This chapter contributes to the thesis research questions RQ2 (democratization of specialized knowledge in the SE domain) and RQ3 (post-processing and hybrid approaches for reliability enhancement). Our research addresses the following chapter-specific questions:

- c4RQ1:** Do current LLMs provide sufficient accuracy for automatic code generation across varying problem complexities, or do systematic limitations persist regardless of model scale?
- c4RQ2:** Can evolutionary computation techniques, specifically GI, enhance the functional correctness of LLM-generated code in a statistically significant manner?
- c4RQ3:** How does the effectiveness of GI-based enhancement vary across different LLM architectures and problem domains?

These chapter-specific research questions directly address the thesis-level questions: c4RQ1 contributes to RQ2 by empirically assessing the current limitations of LLMs in democratizing software development, while c4RQ2 and c4RQ3 contribute to RQ3 by developing and evaluating hybrid approaches that enhance LLM reliability.

4. LLMGIPY

We propose a novel approach that combines the generative capabilities of LLMs with the optimization power of EA [121]. Our method employs GE with dynamically specialized grammars to iteratively refine LLM-generated code using test-driven fitness evaluation. This hybrid approach leverages the complementary strengths of neural and evolutionary paradigms: LLMs provide reasonable initial solutions and domain-relevant components, while GI systematically explores the solution space to achieve functional correctness.

The core insight underlying our approach is that even when LLMs fail to produce correct code, they often generate solutions with correct high-level structure but flawed implementation details. These partial solutions contain valuable information, including relevant library imports, appropriate function signatures, and domain-specific constants, that can seed an evolutionary search process. By extracting this information to specialize a grammar dynamically, we constrain the search space to regions likely to contain correct solutions, enabling efficient refinement through GI.

The evaluation of LLMs for code generation presents unique methodological challenges. Traditional metrics often focus exclusively on syntactic validity or EM accuracy, failing to capture semantic correctness or functional behavior. Moreover, limited test coverage can create false positives where code passes available tests but fails on edge cases. Our approach addresses these evaluation challenges through comprehensive test suites and fitness functions that assess functional correctness rather than syntactic similarity.

Historical context reveals that early LLMs exhibited poor performance in code generation tasks, even for popular languages like Python with extensive representation in training data. Models with fewer than 50 billion parameters continue to struggle with code synthesis, producing outputs with logical errors, incorrect control flow, or improper handling of boundary conditions [23]. While recent models have shown improvement, they remain vulnerable to generating plausible but incorrect solutions, particularly for languages with limited training representation.

Our GI approach presents several notable advantages. The grammar-based methodology is inherently transferable to any programming language for which a BNF grammar can be specified, enabling broad applicability. It is especially effective for underrepresented languages where LLMs tend to perform poorly. Empirical results further demonstrate that GI can deliver substantial gains in code correctness, improving in a statistically significant way the code generated by LLMs across diverse problem domains.

The significance of this work extends beyond immediate technical contributions. As LLMs become integral to modern development workflows, ensuring their reliability becomes paramount. Our approach provides a principled post-processing mechanism that

enhances code quality without requiring model retraining or architectural modifications. This compatibility with existing LLM infrastructure facilitates immediate practical deployment while advancing theoretical understanding of hybrid neuro-evolutionary systems.

Building upon the foundational methodology presented in this chapter [121], we subsequently developed an extended study [123] that introduces refined selection strategies and an enhanced fitness function. This extension demonstrates that methodological improvements to the evolutionary process can yield additional performance gains, further validating the hybrid neuro-evolutionary paradigm.

4.2 Related Works

This section contextualizes our approach within the landscape of LLM-based code generation and evolutionary program synthesis. Building upon the foundational concepts established in Chapter 2, we focus here on domain-specific developments and the integration of neural and evolutionary paradigms.

4.2.1 LLMs for Code Generation

The application of LLMs to code generation has emerged as a transformative development in automated software synthesis. Leveraging the Transformer architectures and training methodologies discussed in Sections 2.1 and 2.3, contemporary code-generation models undergo pre-training on vast corpora containing billions of tokens from open-source repositories, technical documentation, and programming forums.

Models such as Codex [23], which powers GitHub Copilot, underwent training on 159GB of Python code from public repositories. Open-source alternatives like CodeLLaMA [133] and StarCoder [98] leverage curated datasets spanning multiple programming languages, with particular emphasis on languages with substantial open-source presence. These models employ specialized tokenization schemes that preserve code structure and semantics, enabling them to learn syntactic patterns, programming idioms, and problem-solving strategies across multiple languages and paradigms.

Despite these advances, a fundamental challenge persists: LLMs frequently generate code that is syntactically valid but semantically incorrect, particularly for complex algorithmic problems or languages underrepresented in training data. This limitation motivates the hybrid approaches developed in this chapter.

4.2.2 Evaluation Benchmarks

The assessment of LLM-generated code necessitates sophisticated evaluation frameworks, as discussed in Section 2.4.1. We selected the PSB2 benchmark [62] for our evaluation due to its comprehensive test coverage (1000 training and 1000 testing cases per problem) and its emphasis on complete programs with complex control flow. Unlike benchmarks focused on isolated functions, PSB2 problems require algorithmic reasoning across multiple computational domains, making them particularly challenging for direct LLM synthesis and thus ideal for assessing the effectiveness of post-hoc improvement techniques.

The $pass@k$ metric, which measures the probability that at least one of k generated samples passes all tests, provides insight into both accuracy and consistency. However, as noted in Section 2.4.2, limited test coverage can mask semantic errors, motivating the use of benchmarks with extensive test suites.

4.2.3 Evolutionary Methods with LLMs

The GP and GI foundations established in Section 2.2 provide the theoretical basis for our hybrid approach. While traditional GP evolves programs from scratch, GI operates on the principle that most required functionality already exists within an initial program, requiring only targeted modifications to achieve desired behavior. This assumption aligns naturally with LLM-generated code, which often contains correct structure but flawed implementation details.

Our work innovatively combines LLM generation with **grammar-based genetic improvement (GBGI)**, leveraging complementary strengths of neural and evolutionary paradigms [121]. Unlike previous approaches that treat these technologies independently, we establish a pipeline where LLM outputs seed evolutionary search, while grammar specialization incorporates LLM-extracted domain knowledge. This synthesis addresses limitations of both approaches: LLMs' tendency toward plausible but incorrect solutions, and GI's challenge in generating initial functionality from scratch.

The integration introduces several technical innovations. Dynamic grammar specialization extracts constants, function names, and library imports from LLM outputs, incorporating domain-specific elements that would be difficult to evolve independently. Test-driven fitness evaluation provides objective correctness measures that guide evolutionary search toward functional solutions. The approach's language-agnostic design, requiring only BNF grammar specification as described in Section 2.2.1, ensures broad

applicability across programming paradigms.

Recent concurrent work has explored related directions. Liventsev et al. [106] investigated fully autonomous programming with LLMs, while Chen et al. [22] explored evolutionary approaches to prompt optimization for code generation. Tao et al. [145] proposed grammar-obeying program synthesis combining LLMs with many-objective genetic programming. Our approach differs by focusing on post-hoc improvement of LLM-generated code through specialized grammars derived from the model's own output, rather than modifying the generation process itself. At a broader level, recent surveys have further mapped the growing synergies between evolutionary computation and LLMs. Stein and Bäck [141] proposed LLaMEA, a framework that uses LLMs to automatically generate metaheuristic optimization algorithms, demonstrating that LLMs can serve as algorithm designers within an evolutionary loop. A comprehensive survey by Chauhan et al. [21] catalogs the diverse methods, synergies, and applications at the intersection of evolutionary computation and LLMs, situating our work within this rapidly expanding landscape.

4.3 Methodology

4.3.1 Overview

Our methodology establishes a systematic pipeline for evolutionary refinement of LLM-generated code [121]. The approach consists of three principal phases: extraction and encoding of LLM outputs, dynamic grammar specialization based on problem context, and evolutionary optimization using GE. Each phase addresses specific challenges in bridging the gap between LLM generation and functional correctness. Figure 4.1 presents a high-level overview of the proposed workflow.

The proposed method requires only a limited number of LLM queries, and the evolutionary process can efficiently run on commodity hardware by leveraging parallel execution. The methodology is cost-effective as it primarily depends on running an EA, which requires only CPU resources and has an adjustable computational budget based on hardware capacity. Furthermore, the approach is independent of both the programming language and the specific LLM employed, provided a suitable grammar is available for the target language.

4. LLMGIPY

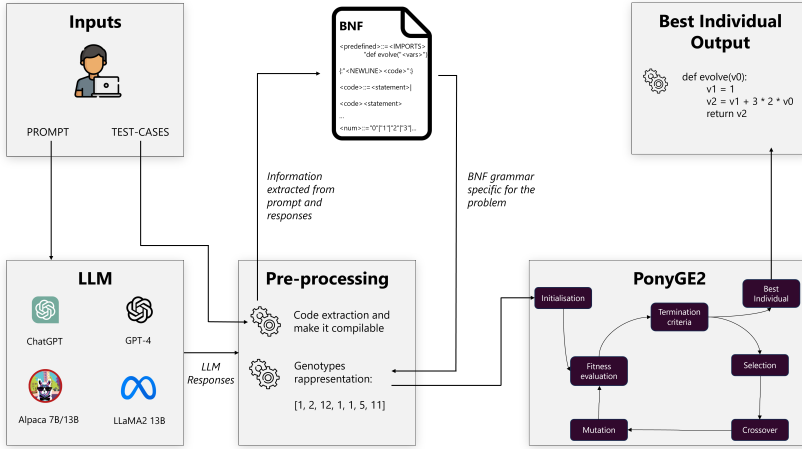


Figure 4.1: High-level workflow of the proposed approach. The user provides a problem description and test cases. The LLM generates initial code, which is extracted and used to specialize a grammar. GE then performs iterative refinement guided by test case evaluation.

4.3.2 Extraction and Preprocessing of LLM Output

The initial phase transforms raw LLM outputs into representations suitable for evolutionary manipulation. When presented with a textual problem description, the LLM generates a response containing both explanatory text and embedded code. This heterogeneous output necessitates sophisticated extraction procedures to isolate executable content. We employ delimiter-based extraction using specialized tokens that demarcate code boundaries within the textual response. Regular expression patterns identify and remove comments, as these elements, while potentially informative for human readers, are superfluous for execution and complicate grammatical analysis. The extraction process must accommodate various formatting conventions, including markdown code blocks, indentation-based structures, and inline code fragments.

Syntactic validation represents a critical preprocessing step, as LLM outputs frequently contain syntax errors that would prevent successful parsing. We implement an iterative correction algorithm that progressively removes problematic lines until achieving syntactic validity.

This algorithm reduces the code to an empty function if no valid solution can be recovered, ensuring a valid starting point for evolutionary refinement. While potentially removing functional code, this approach guarantees syntactic validity for subsequent

Algorithm 1 Syntactic Correction Algorithm

Require: Code string C **Ensure:** Syntactically valid code C'

```

1:  $C' \leftarrow C$ 
2: while not syntactically_valid( $C'$ ) do
3:    $line \leftarrow$  find_first_error_line( $C'$ )
4:   if  $line = \text{null}$  then
5:      $C' \leftarrow$  empty_function()
6:   break
7:    $C' \leftarrow$  remove_line( $C', line$ )
8: return  $C'$ 

```

processing.

Following syntactic correction, we construct an **Abstract Syntax Tree (AST)** representation. The **AST** captures program structure while abstracting implementation details, facilitating meaningful genetic operations. Variable and function names undergo systematic renaming to generic identifiers (v_0, v_1, \dots, v_n), eliminating naming dependencies and reducing the grammar's terminal vocabulary. This normalization process ensures that evolutionary search focuses on algorithmic structure rather than identifier selection.

4.3.3 Dynamic Grammar Specialization

The effectiveness of GE critically depends on grammar design. Overly general grammars create vast search spaces that impede convergence, while restrictive grammars may exclude valid solutions. Our approach dynamically specializes a base grammar using problem-specific information extracted from both the problem description and LLM output. The base grammar encompasses fundamental language constructs including control structures, operators, and standard library functions. For Python, this includes:

```

<program> ::= <imports> "def evolve(" <params> ")" <body>
<imports> ::= <import> <imports> | <imports>
<body> ::= <stmt> | <stmt> <body>
<stmt> ::= <assign> | <if_stmt> | <while_stmt> |
          <for_stmt> | <return_stmt>
<expr> ::= <binary_op> | <unary_op> | <literal> |
          <var> | <func_call>

```

4. LLMGIPY

This foundation provides syntactic structure while remaining agnostic to problem-specific elements. Dynamic specialization augments the base grammar with domain-relevant terminals extracted through multiple mechanisms:

Constant Extraction: Numeric and string literals appearing in the problem description often represent critical values for the solution. We employ pattern matching to identify potential constants, including numerical values, string patterns, and special characters. These constants are added as terminal productions, increasing their probability of inclusion in evolved solutions.

Library and Function Mining: The LLM output provides valuable hints about relevant libraries and functions. We parse import statements and function calls, adding these as specialized productions. This mechanism leverages the LLM's training on vast code corpora to identify domain-appropriate tools without manual specification.

Keyword Extraction: We apply KeyBERT [54], a keyword extraction algorithm based on BERT embeddings, to identify semantically important terms from the problem description. These keywords are incorporated as string terminals, potentially serving as variable names, dictionary keys, or string operations.

The specialization process maintains probability weights for productions, with extracted elements receiving higher weights to encourage their use while preserving grammar completeness. This probabilistic approach balances exploitation of problem-specific knowledge with exploration of alternative solutions.

4.3.4 Evolutionary Optimization Process

The evolutionary phase employs GE to refine the LLM-generated code toward functional correctness. The population initialization strategy leverages the LLM output as a seed, creating genetic diversity through replication with minor variations rather than random generation. This approach exploits the assumption that LLM outputs, while imperfect, contain valuable partial solutions.

Fitness Function

The fitness function quantifies solution quality based on test case performance. Given a test suite $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ where x_i represents inputs and y_i represents expected outputs, we define the basic fitness F as the number of passed test cases:

$$F(P) = \sum_{i=1}^n \mathbb{1}[P(x_i) = y_i] \quad (4.1)$$

where $\mathbb{1}[\cdot]$ is the indicator function. This fitness function counts the number of test cases for which the program P produces the correct output, providing a straightforward measure of functional correctness. Higher fitness values indicate better solutions, with the maximum value n representing perfect correctness across all test cases.

Selection and Genetic Operators

Selection employs tournament selection with adaptive pressure, balancing exploitation of promising solutions with maintenance of genetic diversity. The tournament size is set to 50, providing strong selective pressure that accelerates identification of promising solutions while maintaining sufficient diversity for continued exploration. Tournament selection operates by randomly selecting a fixed number of individuals from the population and choosing the one with the highest fitness as a parent for reproduction.

Crossover operations exchange subtrees between parent ASTs, preserving syntactic validity while combining solution components. We implement depth-aware crossover that preferentially selects crossover points at similar tree depths, reducing the likelihood of creating severely unbalanced offspring.

Mutation operates at the AST level, replacing individual nodes or subtrees with grammatically compatible alternatives. These localized modifications target semantically relevant structures such as conditionals, function calls, and loop constructs. The mutation rate is calibrated to maintain adequate exploratory capability while preserving beneficial genetic material discovered through evolution.

The evolutionary process proceeds through generational replacement with elitism, preserving the best solutions while exploring variations. Elitism ensures that the highest-quality solutions discovered during evolution are never lost due to stochastic variation.

4.3.5 Methodological Extensions

Building upon the foundational approach described above, we developed an extended methodology [123] that introduces three key improvements to enhance the evolutionary process:

Lexicase Selection

We replaced tournament selection with *lexicase selection* [63] combined with a down-sampling strategy. Lexicase selection evaluates individuals sequentially on randomly ordered test cases, selecting solutions that perform well across diverse test case samples. This approach promotes population diversity and reduces the risk of premature convergence to suboptimal solutions. The down-sampling strategy evaluates individuals on only a small fraction of test cases during selection, enhancing computational efficiency while capturing generalization capabilities.

4

Granular Fitness Function (F_E)

We defined an enhanced fitness function F_E that accounts for partial correctness, enabling finer-grained discrimination between solutions:

$$F_E(P) = \sum_{i=1}^n \begin{cases} 1 & \text{if } P(x_i) = y_i \\ 1 - \min\left(1, \frac{|P(x_i) - y_i|}{\max(|y_i|, \epsilon)}\right) & \text{otherwise} \end{cases} \quad (4.2)$$

This function assigns partial credit based on how close the output is to the expected value, enabling the evolutionary process to distinguish between solutions with the same number of passed test cases but different error magnitudes.

Updated Model Evaluation

Due to the rapid evolution of LLMs, the extended study evaluated more recent models including CodeLlama-7B, LLaMA-3-8B, ChatGPT, and GPT-4, ensuring that findings remain relevant in the current AI landscape.

4.4 Experimental Evaluation

4.4.1 Benchmark and Problem Selection

Our experimental evaluation employs the PSB2 benchmark suite [62], whose characteristics are detailed in Section 2.4.1. We selected PSB2 for this investigation because its comprehensive test coverage (1000 training and 1000 testing cases per problem) enables reliable fitness evaluation during evolution, and its emphasis on complete programs with complex control flow makes it particularly challenging for direct

LLM synthesis—precisely the scenario where post-hoc improvement techniques can demonstrate value. The 25 problems span multiple difficulty levels and programming paradigms, facilitating comparison with related work in program synthesis.

4.4.2 Model Selection

We evaluate five LLMs spanning different architectures and parameter scales:

- **Alpaca-7B (A7)***: A LLaMA-based model fine-tuned on instruction-following data, representing smaller open-source alternatives with 7 billion parameters.
- **Alpaca-13B (A13)†**: Larger variant with 13 billion parameters and improved capacity for complex reasoning.
- **LLaMA-2-13B (L2)‡**: Updated architecture with enhanced training data and optimization techniques.
- **ChatGPT (CG)**: GPT-3.5-based model with extensive fine-tuning for conversational interactions and code generation.
- **GPT-4 (G4)**: State-of-the-art model representing current best performance in code generation tasks.

This model selection enables analysis across multiple dimensions: open versus proprietary models, parameter scale effects, and architectural variations. The inclusion of both smaller open-source models and larger proprietary systems allows assessment of GI's effectiveness across the capability spectrum.

4.4.3 Experimental Protocol

Each problem is presented to the LLM using a standardized prompt structure:

Write a single Python function to solve the following problem and include any necessary imports:

{Problem Description}

The problem descriptions are sourced directly from the PSB2 specifications, ensuring consistency and avoiding prompt engineering biases. We generate 10 independent

*<https://huggingface.co/circulus/alpaca-7b>

†<https://huggingface.co/chavinlo/alpaca-13b>

‡<https://huggingface.co/meta-llama/Llama-2-13b-chat-hf>

solutions per problem-model combination, capturing the stochastic nature of LLM generation and providing multiple seeds for evolutionary refinement.

4.4.4 Evolutionary Parameters

The GI process employs evolutionary parameters determined through preliminary experimentation to balance effectiveness and computational feasibility, building upon the GE framework introduced in Section 2.2.1. A population of 1000 individuals provides a reasonable trade-off between computational cost and the preservation of genetic diversity, while the evolutionary process runs for 100 generations, which proved sufficient to reach convergence in most problem instances.

Structural complexity is managed by initializing the maximum tree depth at 15 and progressively extending it to 30 over the course of evolution, allowing solutions to grow in expressiveness as the search progresses. Genomes are limited to a maximum length of 200 codons without wrap-around, ensuring controlled growth and maintaining interpretability of the evolved programs. Selection is performed via tournament selection with a tournament size of 50, introducing strong selective pressure that accelerates the identification of promising candidates. Genetic variation is further shaped through a crossover rate of 0.8, which promotes substantial recombination between parent solutions, and a mutation rate of 0.2, which sustains exploratory capacity throughout the search. Collectively, these parameters reflect a deliberate balance between exploration and exploitation, with strong selection driving rapid improvement while mutation helps preserve genetic diversity.

4.4.5 Baseline Comparisons

To assess the effectiveness of the GI enhancement, we compare it against two baseline approaches:

Direct LLM Generation (\mathcal{L}): Evaluates the unmodified output produced by the language model on the test cases. This baseline reflects the standard practice in LLM-based code generation and establishes the initial quality of model outputs.

LLM Self-Correction (\mathcal{L}^+): Employs an iterative refinement procedure in which the LLM is provided with feedback on failing test cases and attempts to correct its own output accordingly. This baseline tests whether LLMs can effectively debug their own outputs given explicit error feedback.

For the self-correction baseline, we implement the following iterative protocol:

1. If the code times out, the LLM is prompted: “Your code is too slow. Please rewrite it and make it correct and more efficient.”
2. If the code contains syntax errors, the LLM is prompted: “Your code contains syntax errors. Please rewrite it and fix all the syntax errors.”
3. If the code produces incorrect outputs, 10 pairs of input-output examples from the failing test cases are provided with the prompt: “Your code is incorrect. Please rewrite it.”

This process continues for up to 10 iterations or until no improvement occurs. We conduct 2 self-correction runs per problem-model combination and retain the best solution across iterations based on training set performance.

4.4.6 Evaluation Metrics

We evaluate solution quality using metrics aligned with the evaluation framework discussed in Section 2.4.2:

Test Pass Rate: The primary indicator of correctness, capturing the proportion of test cases for which the produced outputs are accurate. We report performance on both the training set (used during evolution) and a held-out test set to assess generalization.

Convergence Speed: Measured as the number of generations required to reach maximum fitness, providing insight into the efficiency of the evolutionary search.

Statistical Significance: Established through the Wilcoxon-Mann-Whitney test for pairwise comparisons and the Kruskal-Wallis test for multiple-group analyses, applying the Holm-Bonferroni correction to account for multiple testing. All statistical analyses adopt a significance level of $\alpha = 0.05$.

4.4.7 Grammatical Constraints

The experimental grammar encompasses a substantial subset of Python, excluding certain advanced features to maintain manageable search spaces:

- Lambda expressions and nested function definitions
- Class definitions and object-oriented constructs
- Generator expressions and `async/await` syntax
- Non-standard library imports beyond those identified during specialization

4. LLMGIPY

These constraints balance expressiveness with tractability, covering the vast majority of algorithmic solutions while avoiding combinatorial explosion in the search space. The grammar’s design ensures that any solution expressible within the constraints can be discovered through evolutionary search given sufficient computational resources.

4

4.4.8 Computational Resources

Experiments were executed on virtual machines running Ubuntu 20.04, with 16 dedicated cores on Intel Xeon Gold 6140 CPUs and 128 GB of RAM. On average, a single repetition of an evolutionary process consisting of 100 generations with a population size of 200 completes in less than 30 minutes, including fitness evaluations on both training and test sets. This computational efficiency enables comprehensive experimentation across multiple models and problems within reasonable time constraints. Our Python implementation is publicly available at <https://github.com/dravalico/LLMGIPy>.

4.4.9 Extended Experimental Setup

In the extended study [123], the experimental configuration was adjusted to evaluate the methodological enhancements outlined in Section 4.3.5. Notable changes include an updated selection of LLM models—CodeLlama-7B (CL7), LLaMA-3-8B (L3), ChatGPT (CG), and GPT-4 (G4)—to ensure alignment with current state-of-the-art capabilities. The population size was reduced to 200 individuals to improve computational efficiency, while diversity was maintained through the adoption of *lexicase selection*. This selection strategy replaced traditional tournament selection by evaluating candidates on randomly ordered subsets of test cases. Additionally, the granular fitness function F_E was introduced and compared against the basic fitness F , allowing assessment of the impact of partial correctness scoring.

The original configuration is referred to as I_T^F , characterized by tournament selection and basic fitness, whereas the extended configuration, $I_L^{F_E}$, combines *lexicase selection* with the granular fitness function. An ablation study was performed, systematically isolating each component to quantify its contribution to overall performance improvements.

4.5 Results

4.5.1 Overall Performance Comparison

Table 4.1 presents the median test pass rates across all problem-model combinations. The results demonstrate consistent improvement through GI enhancement, with statistically significant gains for nearly all configurations where the initial LLM output was imperfect.

Table 4.1: Median number (scaled in $[0, 1]$) of passed test cases for each problem P and LLM. We highlight in bold the methods that are statistically significantly better for the same problem and LLM. Problems that are directly solved from \mathcal{L} are indicated with \checkmark . Problems that are not parsed for the GI are indicated with np (not parsed). With a 0 we denote problems that do not pass any test cases.

Problem	A7			A13			L2			CG			G4		
	\mathcal{L}	\mathcal{L}^+	GI	\mathcal{L}	\mathcal{L}^+	GI	\mathcal{L}	\mathcal{L}^+	GI	\mathcal{L}	\mathcal{L}^+	GI	\mathcal{L}	\mathcal{L}^+	GI
BS	0.01	0	0.58	0	0	0.20	0.13	0.00	0.20	\checkmark	—	—	\checkmark	—	—
BB	0	0	np	0	0	np	0	0	np	0.00	0.00	0.06	0.00	0.04	0.06
BW	0	0	np	0	0	0.03	0	0	0.06	0.00	0.00	0.03	0.04	0.00	0.04
CC	0	0	0.31	0	0	np	0	0	0.31	\checkmark	—	—	\checkmark	—	—
CS	0	0	np	0	0	0	0	0	0	0	0	0	0	0	0
CV	0	0	np	0	0	0	0	0	np	0	0	0	0	0	0
DG	0	0	np	0	0	np	0	0.00	0.00	0.07	0.00	0.17	0.15	0.00	0.17
FP	0	0	np	0	0	np	0	0	0	0	0	0	0	0	0
FB	0	0	np	0.94	0	0.94	\checkmark	—	—	\checkmark	—	—	\checkmark	—	—
FC	0	0	np	0	0	np	0.18	1.00	1.00	\checkmark	—	—	\checkmark	—	—
GD	0	0	np	0	0	np	0	0	0.59	\checkmark	—	—	\checkmark	—	—
IS	0	0	np	0.16	0	0.67	\checkmark	—	—	\checkmark	—	—	\checkmark	—	—
LD	0	0	0.09	0.05	0	0.09	0	0	0.09	\checkmark	—	—	\checkmark	—	—
LH	0	0	0.04	0	0	0.06	0	0	0.04	0.52	0.00	1.00	0.00	0	0.08
MM	0	0	np	0	0	0	0	0	np	0	0	0	0	0	0
MC	0.50	0	np	0	0	0.02	\checkmark	—	—	\checkmark	—	—	\checkmark	—	—
PD	0	0	0.13	0	0	0.13	0	0.01	0.19	\checkmark	—	—	\checkmark	—	—
SL	0	0	0.00	0	0	0.00	0.00	0	0.00	0.00	0	0.01	0.00	0	0.01
SD	0	0	np	0	0	np	0.05	0	np	0.05	0.00	0.06	0.04	0.00	0.09
SB	0.50	0	0.78	0	0	np	0.49	0.00	0.55	0.00	0.42	np	0	0.67	0.50
SW	0	0	0.34	0	0	0.34	0.56	0.00	0.56	\checkmark	—	—	\checkmark	—	—
SQ	0	0	0.01	0	0	0.01	0.00	0	0.01	\checkmark	—	—	\checkmark	—	—
SC	0	0	np	0	0	0.05	0.04	0	0.05	\checkmark	—	—	\checkmark	—	—
TW	0.30	0	np	0.30	0	np	0.99	0	np	\checkmark	—	—	\checkmark	—	—
VD	0	0	np	0	0	np	0	0	np	0.96	0.00	1.00	0.95	0.00	0.95

4.5.2 Model-Specific Performance Analysis

The results reveal distinct performance patterns across model scales. Smaller models (A7, A13, L2) exhibit weak initial performance, typically yielding median pass rates below 0.1 across most problems. Nevertheless, GI delivers substantial improvements, frequently achieving 10–50 \times increases in test pass rates. The high proportion of parsing

4. LLMGIPY

failures (“np”) observed for A7 highlights the limited quality of its initial outputs, which often contain structural errors that fall outside the expressiveness of our grammar.

In contrast, larger models (CG, G4) directly solve 11–13 problems without any enhancement, indicating markedly stronger baseline capabilities. For the remaining problems where large models produce imperfect solutions, GI still proves beneficial. Notable examples include LH(0.52 \rightarrow 1.00 for CG) and VD(0.96 \rightarrow 1.00 for CG), demonstrating that even near-correct solutions can benefit from evolutionary refinement.

4

4.5.3 Self-Correction Baseline Analysis

The self-correction baseline (\mathcal{L}^+) proves largely ineffective across all models. Only two instances show meaningful improvement: L2on FCachieves perfect correction (0.18 \rightarrow 1.00), and G4on SBreaches 0.67 accuracy. The general failure of self-correction suggests that LLMs struggle to identify and rectify logical errors in their own outputs, even when provided with explicit test failure information.

This limitation appears fundamental to the autoregressive generation paradigm, where models commit to solution strategies early in generation and cannot effectively backtrack when those strategies prove flawed. The contrast with GI’s success highlights the value of systematic search over self-reflective refinement.

4.5.4 Evolutionary Dynamics

Figure 4.2 illustrates fitness evolution across generations for representative problems. Two recurring evolutionary patterns emerge from our analysis:

The first pattern, *gradual improvement*, is characteristic of smaller models. These models generally display steady fitness gains across generations, indicating that their initial outputs, although weak, contain partially correct structures that can be incrementally refined through evolutionary search. Tasks such as BSand SWillustrate this pattern, showing continuous progress as evolution proceeds.

The second pattern, *bimodal distribution*, is predominantly observed in larger models, whose outputs tend to be either correct from the outset or fundamentally flawed. In cases where the initial solution is incorrect, evolutionary improvement remains minimal, suggesting that high-quality initial solutions induce local optima that limit the effectiveness of further search.

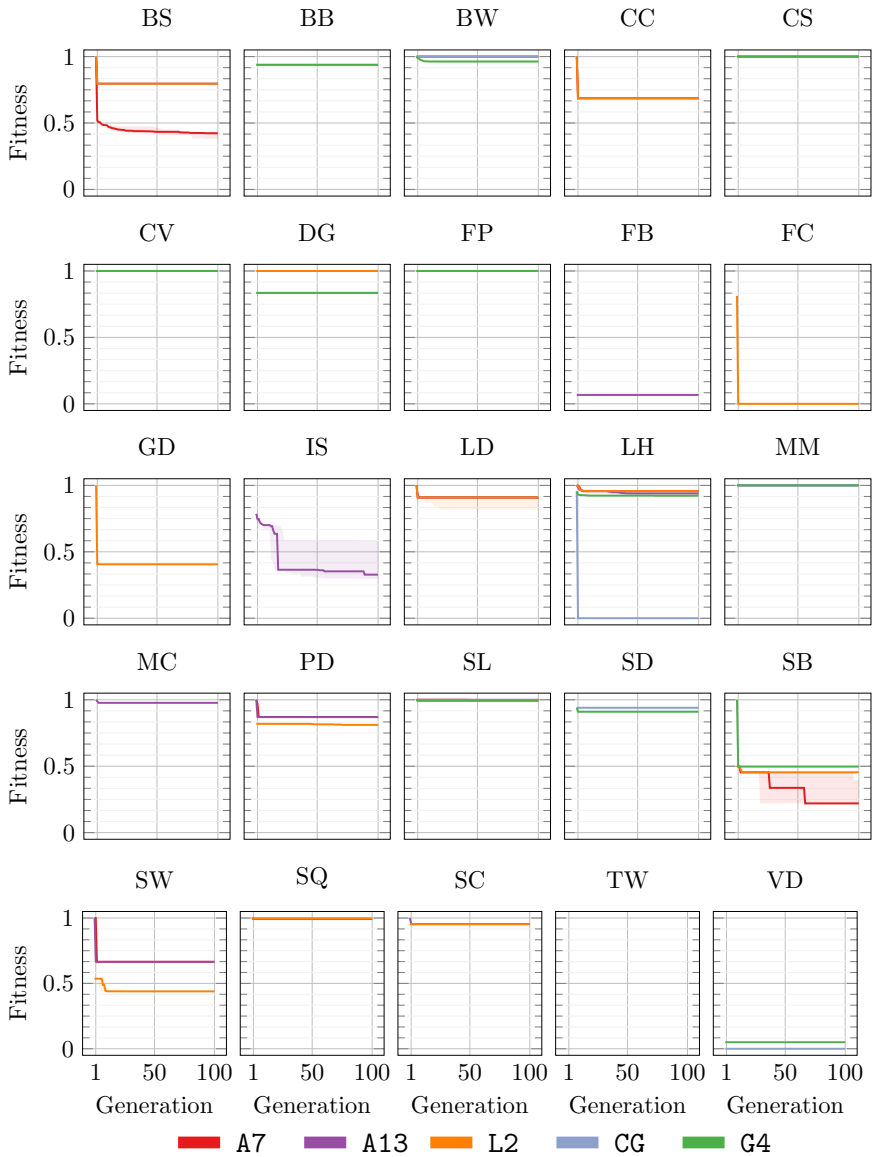


Figure 4.2: Trend of the fitness (scaled in $[0, 1]$) across the generations of GI. Each line represents the median fitness across all runs. Shaded area represents the inter-quartile range. We omit the corresponding lines where no GI is performed.

4.5.5 Statistical Significance Analysis

We perform comprehensive statistical testing to validate the significance of observed improvements. The Kruskal-Wallis test confirms significant differences among approaches ($p < 0.001$) for all problem-model combinations with imperfect initial solutions. Pairwise Wilcoxon-Mann-Whitney tests with Holm-Bonferroni correction establish that GI significantly outperforms both \mathcal{L} and \mathcal{L}^+ baselines in 89% of applicable cases.

The remaining 11% of cases without significant improvement correspond to problems where:

1. The search space is too complex relative to available fitness signal (e.g., MM)
2. The initial solution is too distant from correct implementations (e.g., problems with “np” designation)
3. Test coverage is insufficient to guide meaningful evolution (e.g., SL with only binary outcomes)

4.5.6 Impact of Grammar Specialization

An examination of the specialized grammars underscores the importance of domain-specific augmentation. The problems in which GI achieves the largest performance gains consistently coincide with cases where the LLM output provides extractable constants, operators, or functions directly relevant to the task.

Representative examples include:

- **LHAlgorithm:** The presence of modulo operations and digit-manipulation routines in the LLM output enables rapid convergence toward a correct checksum implementation.
- **SC:** Identifying string-translation utilities and character-mapping constructs from the model's output allows evolution to assemble the appropriate encryption logic.
- **VD:** Detection of mathematical functions such as `sqrt` and `pow` from the model's imports guides the search toward the correct distance formula.

In contrast, tasks showing limited improvement tend to lack such recoverable domain knowledge, forcing the evolutionary process to operate solely with the base grammar and thereby constraining its effectiveness. This observation validates the core hypothesis underlying our approach: that LLM outputs contain valuable structural and domain knowledge even when semantically incorrect.

4.5.7 Extended Results with Improved Methodology

The extended study [123] evaluated the methodological improvements on the same PSB2 benchmark with updated LLMs. Table 4.2 summarizes the comparative performance between the original approach (I_T^F) and the extended configuration (I_L^{FE}).

Table 4.2: Comparison of original (I_T^F) vs. extended (I_L^{FE}) methodology. Values indicate number of problem-model combinations where each approach achieved statistically significant better performance.

Comparison	I_L^{FE} wins	I_T^F wins
Overall	11	1
Smaller models (CL7, L3)	9	0
Larger models (CG, G4)	2	1

The extended configuration I_L^{FE} outperformed the original I_T^F in 11 problem-model combinations, while I_T^F prevailed in only 1 case. Notably, 9 of the 11 improvements occurred with smaller LLMs (CodeLLama-7B-Instruct (CL7)[§] and LLaMA-3-8B (L3-8B)[¶]), where there is more room for evolutionary correction. The ablation study confirmed that both *lexicase selection* and the F_E function contribute independently to these improvements, with their combination yielding the strongest overall performance.

Overall, the extended study demonstrated that for problems where the LLM alone does not produce a fully correct solution, GI significantly improves the initial LLM-generated code in 50% to 75% of cases across the tested models [123].

4.6 Discussion

4.6.1 Research Question Analysis

RQ4.1: Limitations of Current LLMs

Our results show that current LLMs, regardless of scale, cannot reliably generate correct code for diverse programming tasks. Even state-of-the-art models like GPT-4 directly solve only 52% (13 out of 25) of PSB2 problems, with the remainder producing partially or completely incorrect solutions. Smaller models perform substantially worse, with suc-

[§]<https://huggingface.co/codellama/CodeLlama-7b-Instruct-hf>

[¶]<https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct>

4. LLMGIPY

cess rates below 20%, indicating that model scale alone does not guarantee functional correctness.

The failure modes vary systematically by model size. Smaller models frequently produce syntactically invalid or semantically meaningless code, suggesting insufficient understanding of programming constructs and language semantics. Larger models generate plausible but subtly incorrect solutions, handling common cases while failing on edge conditions or complex logical requirements. This pattern indicates that increasing model size shifts error types from syntactic to semantic without eliminating fundamental limitations in code generation accuracy.

The self-correction experiments provide additional evidence of LLM limitations. The inability to debug their own outputs, even with explicit test feedback, suggests that error correction requires capabilities beyond current architectures. Models appear to lack the systematic reasoning necessary to trace execution, identify logical flaws, and implement corrective modifications. This finding aligns with broader observations about LLM limitations in tasks requiring multi-step logical reasoning and self-reflection.

c4RQ1 Answer

Current LLMs cannot reliably generate correct code regardless of scale. Even GPT-4 directly solves only 52% (13/25) of PSB2 problems; smaller models achieve <20% success rates. Failure modes vary systematically: smaller models produce syntactically invalid code, while larger models generate plausible but subtly incorrect solutions. Self-correction experiments confirm that LLMs lack the systematic reasoning required for debugging, even with explicit test feedback.

RQ4.2: Effectiveness of Genetic Improvement

GI demonstrates statistically significant enhancement of LLM-generated code across nearly all tested configurations. The statistically significant enhancement observed across nearly all tested configurations validates the hypothesis that evolutionary search can address the semantic incorrectness that LLMs struggle to resolve independently. This improvement matters because the approach requires no modification to the underlying LLM and operates entirely as a post-processing mechanism.

The success of GI can be attributed to the interaction of several complementary factors. Its search process systematically explores variations that preserve syntactic validity while altering semantic behavior, allowing correct solutions to emerge through

incremental refinement rather than single-shot generation. At the same time, test-driven fitness offers an objective and external source of guidance, compensating for the lack of explicit feedback that LLMs face during autoregressive generation. In addition, the population-based nature of the approach maintains diversity among candidate solutions, preventing premature commitment to suboptimal paths—a limitation that arises in autoregressive generation when early token choices strongly constrain subsequent possibilities.

Grammar specialization proves crucial to GI effectiveness. By incorporating problem-specific elements from both the natural language description and LLM output, the search space becomes tractable while retaining necessary expressiveness. This dynamic adaptation represents a key methodological innovation, as manually crafted grammars would be impractical for real-world deployment where problems vary widely in domain and structure.

c4RQ2 Answer

GI demonstrates statistically significant enhancement ($p < 0.001$) across nearly all tested configurations where the initial LLM output was imperfect. Success factors include: systematic exploration preserving syntactic validity, test-driven fitness providing objective guidance, and population-based diversity preventing premature convergence. Dynamic grammar specialization incorporating problem-specific elements from LLM outputs proves crucial, making the search space tractable while retaining expressiveness.

RQ4.3: Variation Across Models and Domains

The effectiveness of GI varies systematically with both model quality and problem characteristics. Smaller models benefit more from enhancement, as their outputs typically contain correct high-level structure, but flawed implementation details—errors well-suited to evolutionary refinement. The partial correctness of smaller model outputs provides a foundation upon which GI can build through incremental improvements.

Larger models' outputs exhibit a bimodal distribution: either correct from the outset or fundamentally wrong, offering less opportunity for incremental improvement. When large models fail, they often fail in ways that require substantial structural changes rather than localized corrections, placing the correct solution outside the reachable neighborhood of the initial code through typical evolutionary operations.

4. LLMGIPY

Problem domain significantly influences enhancement success. String manipulation and numerical computation problems show consistent improvement, likely due to well-defined correctness criteria and modular solution structure that admits incremental refinement. Conversely, complex algorithmic problems with interdependent logic prove resistant to evolutionary enhancement, as single mutations rarely transition between fundamentally different algorithmic approaches.

The correlation between grammar extraction success and GI performance highlights the importance of domain knowledge transfer. Problems where LLMs identify relevant libraries and functions see greater improvement, as the specialized grammar provides building blocks for correct solutions. This suggests that hybrid approaches leveraging LLM knowledge extraction with evolutionary search offer synergistic benefits beyond either technique in isolation.

4

c4RQ3 Answer

GI effectiveness varies systematically with model quality and problem domain. Smaller models benefit more from enhancement as their outputs contain correct structure but flawed details—errors suited to evolutionary refinement. Larger models exhibit bimodal outputs (correct or fundamentally wrong), offering less opportunity for incremental improvement. String manipulation and numerical computation show consistent gains; complex algorithmic problems with interdependent logic prove more resistant. Grammar extraction success correlates strongly with GI performance.

4.6.2 Comparison with Alternative Approaches

GI offers several advantages compared to alternative code improvement strategies. Reprompting the LLM may generate different solutions, but these can still be incorrect, as the model lacks insight into why its previous attempts failed. FT requires large datasets and significant computational resources, making it unsuitable for real-time code correction or deployment on problems not represented in training data. Rule-based approaches often fail to generalize across diverse problems, requiring extensive manual engineering for each new domain.

In contrast, GI uses evolutionary principles to iteratively refine solutions, allowing corrections to evolve based on user-provided test cases without requiring changes to the LLM itself. The approach is computationally efficient, running on commodity hardware,

and requires only black-box access to the model's outputs rather than internal weights or gradients.

4.6.3 Limitations and Threats to Validity

Several limitations constrain the interpretation and generalization of our findings. The fitness function's reliance on test suites introduces a risk of overfitting: although evaluation is performed on held-out test sets, both training and testing data are drawn from the same underlying distribution. As a result, real-world tasks may expose unseen error modes or edge cases not represented in PSB2, potentially diminishing the effectiveness of GI in deployment scenarios. Another constraint concerns grammar coverage. The restricted Python subset employed in this study excludes constructs such as lambda expressions, generators, and advanced object-oriented patterns. This limitation narrows the expressiveness of the search space and may preclude the discovery of more concise or efficient solutions that depend on these language features.

Computational cost further affects the feasibility of the approach. Comprehensive experimentation for each problem-model combination requires hundreds of CPU-hours due to the need for repeated fitness evaluations. Although individual evolutionary runs typically complete within thirty minutes, scaling the methodology to larger problems or broader hyperparameter searches remains challenging, particularly in resource-constrained environments. The choice of benchmark also limits generalizability. While PSB2 offers a diverse set of tasks, it represents a relatively narrow class of algorithmic problems whose scale, structure, and specification differ substantially from real-world SD contexts characterized by complex dependencies, external libraries, and underspecified requirements. Finally, outcomes are sensitive to model selection. The evaluation relies on specific LLM versions that may not reflect the latest state-of-the-art capabilities. Given the rapid evolution of model families, future systems may exhibit different error profiles or levels of responsiveness to evolutionary enhancement, making periodic re-evaluation necessary as new models emerge.

4.6.4 Analysis of Extended Methodology

The extended study [123] provides additional insights into the factors influencing GI effectiveness. The superior performance of *lexicase selection* over tournament selection can be attributed to its ability to maintain population diversity by selecting individuals

that excel on different subsets of test cases. This diversity preservation prevents premature convergence and enables exploration of a broader solution space.

The granular fitness function F_E contributes by providing more informative gradient signals during evolution. When two solutions pass the same number of test cases, F_E can distinguish between them based on how close their incorrect outputs are to expected values. This finer discrimination helps guide the search toward solutions that are “almost correct” and may need only minor adjustments.

Notably, the improvements from the extended methodology are most pronounced for smaller LLMs, where there is greater room for evolutionary correction. For larger models that already produce near-correct solutions, the marginal benefit of improved selection and fitness strategies diminishes, as the initial code quality is already high.

4.7 Conclusions

This chapter has presented a comprehensive investigation into enhancing LLM code generation through GI [121]. Our approach successfully addresses the persistent challenge of semantic incorrectness in LLM-generated code by combining neural generation with evolutionary refinement. The core contribution lies in demonstrating that evolutionary computation and neural language models can work synergistically, with each approach compensating for the other’s limitations.

The empirical evaluation across 25 problems and 5 models demonstrates that GI is able to improve in a statistically significant way the code generated by LLMs, with some cases achieving perfect correctness from completely incorrect initial solutions. These improvements are statistically significant ($p < 0.001$) and consistent across different model architectures and problem domains. The success of our approach stems from the synergistic combination of LLM knowledge extraction and systematic evolutionary search, where LLMs provide reasonable initial structures and domain-relevant components while GI refines semantic correctness through test-driven optimization.

The superiority of GI over self-correction approaches reveals fundamental limitations in current LLM architectures. While LLMs excel at pattern recognition and syntactic generation, they lack the systematic reasoning required for debugging and logical refinement. Evolutionary approaches complement these capabilities by providing objective fitness evaluation and exploratory search mechanisms that escape local optima in solution space. This complementarity suggests that hybrid neuro-evolutionary systems represent a promising direction for reliable code generation.

Our methodological contributions extend beyond immediate technical achievements. The dynamic grammar specialization mechanism represents a methodological advance in evolutionary program synthesis. By automatically extracting and incorporating problem-specific knowledge from LLM outputs, we eliminate the need for manual grammar engineering while maintaining search efficiency. This automation is essential for practical deployment where domain experts cannot be expected to craft specialized grammars for each problem. The grammar-based approach ensures transferability across programming languages, requiring only BNF specification for new language support. This generality is particularly valuable for underrepresented languages where LLMs exhibit weaker performance due to limited training data. As programming languages continue to proliferate and evolve, approaches that gracefully handle linguistic diversity become increasingly important for real-world applicability.

4.7.1 Summary of Extended Work

The extended study [123] validated and refined the foundational methodology by introducing a set of interrelated improvements to the evolutionary framework. In place of tournament selection, *lexicase selection* with down-sampling was adopted, fostering greater population diversity and improved generalization by evaluating candidates across varied test case orderings. At the same time, the introduction of a more granular fitness function, (I_L^{FE}) , allowed for finer discrimination between candidate solutions by accounting for partial correctness, thereby providing richer and more informative gradient signals throughout the evolutionary process. The evaluation protocol was also updated to include more recent LLM architectures—CL7, L3-8B, CG, and G4—ensuring that the results remained relevant with respect to the current state of the art.

As a result of these refinements, the extended configuration (I_L^{FE}) outperformed the original approach (I_T^F) in 11 out of 12 comparable cases, with nine of these improvements observed on smaller models, where evolutionary correction exerts a stronger influence. Taken together, these findings demonstrate that targeted methodological enhancements to the evolutionary process lead to substantial performance gains, providing further empirical support for the effectiveness of the hybrid neuro-evolutionary paradigm.

4.7.2 Implications and Future Directions

The implications for SE practice are substantial. As LLMs become integrated into development workflows through tools like GitHub Copilot, ensuring code correctness becomes paramount. Our approach provides a principled post-processing mechanism that enhances reliability without requiring model retraining or architectural modifications. This compatibility with existing infrastructure facilitates immediate adoption while research continues on more fundamental improvements to generative models.

Future research directions include extending the approach to additional programming languages, exploring multi-objective optimization considering factors beyond correctness (efficiency, readability, security), and investigating the co-evolution of prompts alongside code. The integration of formal verification techniques could provide stronger correctness guarantees, while incremental learning approaches might enable the system to improve over time based on accumulated experience. Additionally, analyzing the minimum number of test cases needed to achieve meaningful improvement would provide insights into the practical effort required from users deploying this approach.

4.8 Current and Future Impact

The importance of reliable code generation extends far beyond traditional SD. Modern AI systems increasingly rely on dynamic code generation for complex reasoning and tool use, making the correctness of generated code critical for system reliability and safety. This section examines the broader implications of our work for contemporary AI applications and future technological developments.

4.8.1 AI Agents and Dynamic Code Generation

Modern AI agents use LLMs to generate code on-the-fly rather than executing pre-written programs. When an agent needs to query a database or call an API, it writes the necessary code at runtime. This makes code correctness a runtime concern, not just a development-time one. This fundamental shift makes code generation quality a critical bottleneck in agent capabilities.

Tool use in AI systems exemplifies this dependency. Modern agents interact with external services, databases, and APIs through dynamically generated code. When an agent needs to analyze data, it might generate SQL queries or Python scripts executed in sand-

boxed environments. The correctness of these generated programs directly determines whether the agent accomplishes its intended task or produces erroneous results that cascade through subsequent operations.

Consider a financial analysis agent tasked with portfolio optimization. The agent might generate code to fetch market data, calculate risk metrics, and optimize asset allocation. Each component requires the correct implementation of mathematical formulas, proper handling of edge cases, and appropriate error management. A subtle error in volatility calculation or correlation matrix construction could lead to catastrophic investment decisions. Our GI approach provides a safety net, validating and correcting generated code before execution.

4.8.2 Multi-Step Reasoning and Error Propagation

The challenge intensifies with multi-step reasoning tasks where agents decompose complex problems into programmatic subtasks. Each step might involve generating code for data transformation, statistical analysis, or logical inference. Errors compound through the pipeline, making end-to-end correctness increasingly unlikely without systematic validation and correction mechanisms.

Database interaction presents another critical domain. Agents frequently generate SQL queries to extract insights from structured data. These queries must handle null values, aggregate functions, joins, and subqueries correctly. A malformed query might return plausible but incorrect results, leading to flawed conclusions that users trust due to the agent's authoritative presentation. Our test-driven enhancement approach can validate query correctness against known data patterns and expected results, providing an additional layer of verification before results are presented to users.

4.8.3 Implications for AI Safety and Reliability

The implications for AI safety and reliability are profound. As agents assume greater autonomy and responsibility, ensuring the correctness of their generated code becomes a fundamental safety requirement. Financial transactions, medical diagnoses, and critical infrastructure control all depend on accurate program execution. The ability to automatically enhance and validate generated code provides essential safeguards against potentially harmful errors that could arise from incorrect implementations.

Furthermore, the evaluation of AI agents requires accurate assessment of their code generation capabilities. Benchmarks that merely check syntactic validity or superficial

4. LLMGIPY

correctness fail to capture the nuanced requirements of real-world applications. Our comprehensive evaluation framework, combining multiple test cases with semantic correctness criteria, provides more meaningful assessment of agent capabilities and limitations than simple pass/fail metrics.

4.8.4 Economic and Societal Impact

4

The economic impact of improved code generation extends across industries. SD costs could decrease substantially if AI agents reliably generate correct code from specifications. Domain experts without programming expertise could create custom applications by describing requirements in natural language, democratizing software creation beyond professional developers. Scientific research could accelerate through automated implementation of computational models and analysis pipelines, reducing the barrier between theoretical formulation and empirical validation.

Educational applications also benefit significantly. Students learning programming could receive immediate, correct implementations of algorithmic concepts, with our enhancement approach ensuring pedagogical accuracy. Automated tutoring systems could generate practice problems and solutions tailored to individual learning progress, with confidence that provided code exemplifies best practices rather than propagating subtle errors.

4.8.5 Trajectory Toward Reliable Automated Programming

As we advance toward more capable AI systems, code generation will likely remain a fundamental capability. The ability to translate abstract reasoning into executable programs represents a bridge between thought and action, enabling AI systems to manipulate their environment and extend their capabilities. Ensuring this bridge's reliability through approaches like ours is essential for beneficial AI development.

The convergence of LLMs, evolutionary algorithms, and formal methods suggests a future where code generation achieves unprecedented reliability. Our work contributes to this trajectory by demonstrating that hybrid approaches can overcome individual technique limitations. As models grow more capable and evolutionary methods become more sophisticated, we anticipate systems that generate not just correct but optimal code, considering multiple objectives including performance, security, and maintainability.

The methodology presented in this chapter scales naturally with improvements in

both constituent technologies. Better LLMs provide higher-quality initial solutions with more useful domain knowledge to extract for grammar specialization. More efficient EAs enable exploration of larger search spaces within practical time constraints. Enhanced fitness functions incorporating formal specifications could provide stronger correctness guarantees beyond test-based validation.

In conclusion, improving code generation transcends academic interest to become a critical enabler of AI's beneficial impact on society. Our GI approach represents a significant step toward reliable automated programming, with implications spanning from immediate practical applications in developer tools to long-term AI safety considerations. As AI agents become ubiquitous in professional and consumer applications, ensuring their code generation capabilities will be essential for realizing the technology's transformative potential while mitigating associated risks.

II

Metrics and Evaluation

5

New SQL metric for a more precise evaluation of LLM on text-to-SQL tasks

This chapter develops a novel evaluation metric for text-to-SQL systems that addresses the fundamental limitations of existing binary metrics. The **QAS** combines semantic similarity of query structures with execution result similarity to provide granular, continuous assessments of query quality. Evaluation across 11 models from 5 families demonstrates that **QAS** captures partial correctness that binary metrics like **EX** systematically miss, enabling more nuanced model comparison. This investigation directly addresses the thesis research question **RQ1** (evaluation methodologies) by developing metrics that capture semantic correctness and practical utility beyond surface-level comparison.

5. QUERY AFFINITY SCORE

5

Authors: Giovanni Pinna, Yuriy Perezhohin, Luca Manzoni, Mauro Castelli, Andrea De Lorenzo,

Original title: Redefining text-to-SQL metrics by incorporating semantic and structural similarity,

Published in: Scientific Reports, 2025, 15:1: 22357

<https://www.nature.com/articles/s41598-025-04890-9>

5.1 Introduction

NLP has witnessed remarkable advancements in recent years, particularly in the domain of generating Structured Query Language (SQL) queries from textual descriptions, a task commonly referred to as text-to-SQL. This progress has been primarily driven by the growing demand to democratize database interactions, enabling non-technical users to retrieve information from databases using natural language rather than requiring expertise in structured query languages [4, 117]. The implications of this technological advancement are profound: in business intelligence, accurate SQL generation facilitates data-driven decision-making by allowing users to query complex datasets without deep technical knowledge [172]; in data analysis, the ability to extract insights from large datasets through natural language requests significantly improves efficiency and accessibility [55]; and in customer support, natural language interfaces enable staff to query databases directly, thereby improving response times and customer satisfaction. The emergence of LLM has further accelerated progress in this field, offering unprecedented capabilities in understanding and generating SQL queries from natural language prompts [15, 3]. These models, through their ICL capabilities discussed in Section 2.3, can perform prediction tasks without requiring extensive task-specific training data, thereby reducing the resource-intensive process of model FT [15]. However, to establish trust in this technology and facilitate its adoption in real-world applications, it is essential to evaluate these systems comprehensively and accurately. Only through rigorous evaluation can we understand the current capabilities and limitations of text-to-SQL systems and determine the extent to which we can rely on them for critical applications.

The evaluation of text-to-SQL systems presents a fundamental challenge that extends beyond merely examining model outputs. A comprehensive assessment must also look closely at the metrics employed to evaluate these outputs. If the metrics used to assess model capabilities are overly simplistic or approximate, the resulting evaluations may lead to overestimated or underestimated conclusions regarding model performance. This concern is particularly relevant for text-to-SQL evaluation, where traditional metrics have proven to be inadequate in capturing the nuances of query similarity.

As introduced in Section 2.4.3, current evaluation approaches predominantly rely on the binary metrics **EM** and **EX**. While **EM** assesses syntactic identity between generated and reference queries, it fundamentally ignores that multiple syntactically different SQL queries can correctly respond to the same natural language prompt. For instance,

5. QUERY AFFINITY SCORE

the queries `SELECT * FROM users WHERE age > 18` and `SELECT * FROM users WHERE age >= 19` may be semantically equivalent for integer ages but would receive an EM score of 0. Similarly, a query that uses different table aliases, alternative join syntax, or a different ordering of clauses may produce identical results while being marked as incorrect by EM.

EX addresses the syntactic rigidity of EM by comparing execution results rather than textual representation. However, it still returns a binary outcome that fails to quantify the *degree* of difference between result tables when queries do not produce identical outputs. A query that returns 99% of the correct rows receives the same EX score (0) as one that returns completely unrelated data.

The binary nature of these metrics captures only a partial picture of reality. In the text-to-SQL domain, there exist numerous gradations between a completely correct query and a completely incorrect one. A generated query might correctly identify the relevant tables and columns but apply an incorrect filter condition, or it might return the correct data but in an incorrect order. These subtle differences are meaningful for understanding model capabilities and guiding improvement efforts, yet they are entirely obscured by binary evaluation approaches.

The lack of granularity in current metrics has several important implications for practical applications. Metrics that do not differentiate between errors of varying severity treat a missing condition or structural flaw the same as a minor syntax issue, limiting developers' ability to prioritize and address the most impactful problems. Without metrics that provide detailed feedback, it is difficult to identify where models struggle and how to adjust training data or refine architectures effectively. Furthermore, text-to-SQL tasks often involve interpreting subtle user intentions in natural language, and non-granular metrics are unable to assess whether systems capture nuances such as implicit constraints or contextual relationships between tables.

These considerations motivate the main contribution of this chapter, which is the development of a novel metric that captures both the semantic similarity of SQL query structures and the equivalence of their execution results. This metric, which we term the QAS, provides a continuous value that indicates precisely how similar a generated query is to a reference query, considering both its structural composition and the table it produces upon execution.

This chapter contributes to the thesis research question RQ1 (evaluation methodologies) by developing metrics that capture semantic correctness and practical utility beyond surface-level comparison. The chapter-specific research questions that guide this investigation are as follows:

- c5RQ1:** Can a continuous, non-binary metric provide more informative assessments of SQL query similarity compared to existing binary approaches?
- c5RQ2:** How do semantic similarity and execution result similarity contribute individually and jointly to the overall assessment of query quality?
- c5RQ3:** Can the proposed metric effectively differentiate between text-to-SQL models and provide insights into their relative strengths and weaknesses?
- c5RQ4:** To what extent can the metric identify specific types of errors in generated queries, such as missing aggregation operators or incorrect ordering clauses?

These chapter-specific research questions directly address the thesis-level question RQ1: by demonstrating that continuous metrics provide richer information than binary ones (c5RQ1), understanding the individual contributions of different similarity dimensions (c5RQ2), and establishing practical utility for model comparison and error diagnosis (c5RQ3, c5RQ4), we advance the development of meaningful evaluation methodologies for LLM outputs.

The approach proposed in this chapter addresses these questions by combining two complementary similarity measures. The semantic similarity component employs embedding models specifically designed for code to capture the structural and intentional similarity between queries. The table similarity component uses an edit distance-based algorithm to quantify the degree of difference between execution results. These components are combined through a weighted aggregation to produce the final QAS, which provides a comprehensive assessment of query similarity.

The remainder of this chapter is organized as follows. Section 5.2 provides an overview of related work in text-to-SQL systems, benchmarks, and evaluation metrics. Section 5.3 presents the proposed methodology in detail, including the semantic similarity, table similarity, and QAS computation. Section 5.4 describes the experimental setup and evaluation framework. Section 5.5 presents the experimental results. Section 5.6 discusses the findings and addresses the research questions. Finally, Section 5.7 concludes the chapter and outlines directions for future work.

5.2 Related Work

This section contextualizes the proposed metric within the evolution of text-to-SQL systems and their evaluation methodologies. Building upon the foundational concepts of LLM capabilities and evaluation challenges established in Chapter 2, we focus here on

5. QUERY AFFINITY SCORE

domain-specific developments in SQL generation and the limitations of existing metrics that motivate our contribution.

5.2.1 Text-to-SQL Models: Evolution and Categories

Enabling non-technical users to interact effectively with databases has been a longstanding challenge in the field of NLP [117, 82]. The evolution of text-to-SQL models can be traced through three main categories of approaches: rule-based methods, FT methods, and ICL methods.

The earliest models for the text-to-SQL task were primarily rule-based, relying on predefined templates and rules to generate SQL queries [165, 135]. While these models demonstrated reasonable performance on specific domains, they presented significant limitations in terms of adaptability, requiring extensive modification of templates and rules for new domains.

The limitations of rule-based approaches motivated a transition toward sequence-to-sequence models leveraging Long Short-Term Memorys (LSTMs) networks [56] and Convolutional Neural Networks (CNNs) [25]. To address the structural complexity of databases, researchers subsequently explored graph-based representations using Graph Neural Networks (GNNs) to model database schemas [151, 18]. Concurrently, significant advances were achieved through FT pre-trained LM based on the T5 architecture [127, 136, 95]. More recent fine-tuned models, such as DAILSQL [46] and CodeS [96], have pushed the boundaries further, though FT methods remain heavily dependent on large amounts of high-quality labeled data.

The most recent paradigm shift has been driven by the advent of LLM [15, 3, 42, 48, 1, 77, 78]. Leveraging the ICL capabilities discussed in Section 2.3, these models can infer task requirements and generate appropriate responses based solely on provided context, without explicit FT [111, 102, 128, 156, 103]. The C3 SQL model [41] exemplifies this capability, demonstrating the effectiveness of ICL for generating zero-shot SQL queries with high accuracy.

5.2.2 Benchmarks for Text-to-SQL Evaluation

The development of high-quality benchmarks has been fundamental to achieving the impressive results of modern text-to-SQL models. Early benchmarks in this field were primarily single-domain datasets [165, 31, 76], which limited their utility for assessing model generalizability. More recent benchmarks, such as WikiSQL [170] and

Spider [162], are cross-domain, offering broader tests of model capabilities across diverse database schemas.

The *Spider* benchmark [162] has become one of the most widely used evaluation resources in text-to-SQL research, introducing the **EM** and **EX** metrics discussed in Section 2.4.3. However, despite its widespread adoption, *Spider* and similar benchmarks have been criticized for focusing primarily on database schemas rather than data values [60, 152, 92, 91], which can limit their alignment with real-world applications where data content plays a crucial role.

To address these gaps, researchers have introduced advanced benchmarks including **BIRD** [97], **UNITE** [86], and **ScienceBenchmark** [169]. The **BIRD** benchmark [97] is particularly significant as one of the first benchmarks explicitly designed for realistic scenarios, with queries and databases drawn from real-case applications. **BIRD** includes large-scale databases, complex queries requiring reasoning over multiple tables, and scenarios that demand external knowledge beyond the database content. Notably, **BIRD** also introduced the **Valid Efficiency Score (VES)**, which evaluates the efficiency of SQL queries based on their execution time, adding another dimension to performance assessment.

5.2.3 Limitations of Existing Evaluation Metrics

As established in Section 2.4.3, the evaluation of text-to-SQL models has historically relied on binary metrics with fundamental limitations. **EM** overlooks that multiple syntactically different queries can correctly respond to the same natural language prompt, making it overly conservative. **EX** captures semantic equivalence more accurately but cannot quantify partial correctness—a query returning 95% of correct data is treated identically to one returning entirely incorrect results.

Beyond these task-specific metrics, researchers have employed general-purpose semantic similarity measures. $BERT_{score}$ [167] has been used to evaluate semantic similarity between queries by leveraging contextual embeddings [166]. While $BERT_{score}$ captures semantic intent through language-based representations, it is not specifically tailored for SQL. As a result, it may fail to capture critical structural, semantic, and logical aspects unique to SQL queries, such as the significance of specific SQL keywords, the role of subqueries, or the equivalence of different query formulations.

These limitations have significant implications for the advancement of text-to-SQL research. Without metrics that provide fine-grained assessments, researchers cannot precisely identify where models struggle or how close models are to producing correct outputs. This hinders targeted improvement efforts and may lead to misleading conclu-

sions about model capabilities. The scientific community has thus needed new metrics that address these shortcomings, particularly in the era of LLMs where models can produce a diverse range of valid query formulations. The QAS proposed in this chapter directly addresses these limitations by providing continuous, granular assessments that capture both semantic structure and functional correctness.

5.3 Methodology

5

This section presents the proposed metric for evaluating SQL query similarity. The metric integrates two complementary perspectives: semantic similarity, which assesses the structural and intentional resemblance between queries, and table similarity, which quantifies the correspondence between execution results. These components are combined into a unified QAS that provides a continuous, granular assessment of query quality. Figure 5.1 illustrates the high-level pipeline of the proposed approach. Natural language requests are processed by a text-to-SQL model to generate SQL queries. These generated queries are then compared against reference queries from the BIRD benchmark using the proposed metric. The comparison proceeds in three stages, as shown in Figure 5.1: semantic similarity computation (box 1), table similarity computation (box 2), and weighted combination into the final QAS (box 3).

5.3.1 Semantic Similarity (S_C)

The semantic similarity component aims to determine whether different queries serve the same purpose and whether their structures are inherently similar. This is crucial because two syntactically different queries may be attempting to retrieve the same information from a database, and an effective metric should recognize this equivalence.

Existing approaches for assessing semantic similarity include computing the cosine similarity between embeddings or using methods like $BERT_{score}$ [167]. However, these general-purpose methods are not tailored for code and programming languages, making them less effective at assessing the similarity of syntactically different but semantically equivalent SQL queries. SQL has specific structural elements, including keywords (SELECT, FROM, WHERE, JOIN), operators, table and column references, and logical expressions, that general-purpose language models may not adequately capture.

To address this limitation, we employ embedding models specifically designed for code, including SQL queries. These models, built on the same architectural principles as BERT [40], are fine-tuned for programming languages. This specialization enables them

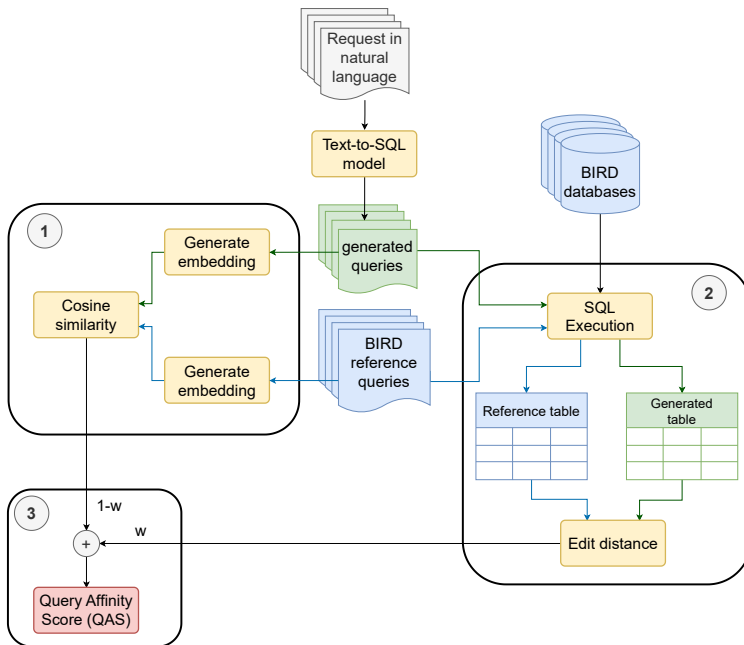


Figure 5.1: High-level pipeline of the proposed approach for comparing SQL queries generated via a text-to-SQL method to the corresponding reference queries on the BIRD databases.

5. QUERY AFFINITY SCORE

to produce more consistent and reliable similarity scores compared to general-purpose models. The embedding models work by mapping queries into dense vectors in a high-dimensional vector space, ensuring that semantically similar queries are positioned closer together in this space. The choice of code-specific embedding models offers several advantages. Domain-specific nuances, such as the meaning of SQL keywords, the use of subqueries, aliases, and the equivalence of different query formulations, are more accurately captured. This approach also addresses a significant limitation of the EM metric—treats any syntactic difference as a mismatch—code-specific embeddings can recognize when different syntactic forms express the same query intent.

5

Formally, let E denote an embedding function that maps a query q to a vector in \mathbb{R}^n , where n is the dimensionality of the embedding space. Given two queries q_1 and q_2 , their semantic similarity S_C is defined as the cosine similarity between their embeddings:

$$S_C(q_1, q_2) = \frac{E(q_1) \cdot E(q_2)^T}{\|E(q_1)\| \cdot \|E(q_2)\|} \quad (5.1)$$

The cosine similarity measure ranges from -1 to 1 , where 1 indicates identical direction (maximum similarity), 0 indicates orthogonality (no similarity), and -1 indicates opposite directions. In practice, for meaningful queries, the similarity values typically fall in the positive range, with higher values indicating greater semantic resemblance.

5.3.2 Table Similarity (S_T)

The second component of the proposed metric assesses the similarity between the result tables obtained by executing the queries. This perspective is essential because the ultimate purpose of a SQL query is to retrieve specific data from a database, and two queries can be considered equivalent if they produce the same output, regardless of their syntactic differences. Current benchmarks address this aspect using the EX metric, which returns a binary value: 1 if the two result tables are identical and 0 otherwise. While this approach captures the notion of functional equivalence, it lacks granularity and fails to quantify how different the result tables are when they do not match exactly. A query that returns 95% of the correct data is treated the same as one that returns entirely incorrect results. To address this limitation, we propose a refined method for evaluating table similarity. The algorithm proceeds as follows:

1. First, check if the generated SQL query string and the reference SQL query string are identical. If so, by definition, the tables produced by these queries will also be identical, and the similarity score is set to 1 .

2. If the strings differ, execute both SQL queries on the database used by the reference query.
3. If the generated query cannot be executed (due to syntax errors or other issues), the similarity score is set to 0.
4. Otherwise, compare the resulting tables using an edit distance-based approach.

Let T_{gen} and T_{ref} represent the result tables produced by the generated query and the reference query, respectively. For each pair of columns (c_i, b_j) , where $c_i \in T_{\text{gen}}$ and $b_j \in T_{\text{ref}}$, we calculate the edit distance required to transform c_i into b_j . The *edit distance* is defined as the minimum number of replace, delete, and insert operations necessary to align the two columns.

The overall table similarity score S_T between T_{gen} and T_{ref} is calculated as the normalized aggregation of the edit distances across all column combinations. For n columns in T_{gen} and m columns in T_{ref} , we define:

$$S_T = 1 - \frac{1}{\max(n, m)} \sum_{i=1}^n \frac{\min_{j=1}^m d(c_i, b_j)}{\max(\text{len}(c_i), \text{len}(b_j))} \quad (5.2)$$

where $d(c_i, b_j)$ denotes the edit distance between columns c_i and b_j , and $\text{len}(\cdot)$ returns the number of rows in a column. The normalization by the maximum column length ensures scale invariance, meaning that the metric is not biased by the absolute size of the tables. The aggregation across column pairs accounts for the overall structural correspondence between the two tables. The result is a similarity measure S_T that ranges in the interval $[0, 1]$, where 1 represents identical tables, and 0 represents complete dissimilarity. This algorithm is robust to variations in column alignment and data length, enabling meaningful comparisons across diverse table sizes.

5.3.3 Query Affinity Score (QAS)

The QAS represents the final, integrated metric that combines S_C and S_T into a unified measure. This combination is essential because neither component alone provides a complete picture of query quality. A query may be semantically similar to the reference but produce different results due to a subtle difference in a filter condition. Conversely, a query may produce identical results to the reference but achieve this through an entirely different approach.

The QAS is computed as a weighted average of the S_C and S_T components:

$$\text{QAS} = (1 - w) \cdot S_C + w \cdot S_T \quad (5.3)$$

5. QUERY AFFINITY SCORE

where w ($0 \leq w \leq 1$) is a weighting factor that balances the contributions of the two components. When $w = 0$, the QAS considers only semantic similarity. When $w = 1$, the QAS considers only table similarity. Intermediate values of w produce a blend of both perspectives. The QAS score ranges between 0 and 1, inheriting this property from both S_T and S_C , which also produce scores within the $[0, 1]$ interval. A QAS of 1 indicates perfect similarity in both semantic structure and execution results, while a QAS of 0 indicates complete dissimilarity. Values between these extremes provide meaningful gradations that reflect partial correctness and similarity.

The weighting parameter w allows the metric to be adapted to different evaluation priorities. In scenarios where the correctness of execution results is paramount, a higher value of w emphasizes table similarity. In scenarios where understanding query intent is more important, a lower value of w emphasizes semantic similarity. For general-purpose evaluation, an equal weighting ($w = 0.5$) provides a balanced assessment that considers both perspectives equally.

5

5.4 Experimental Evaluation

This section describes the experimental setup designed to evaluate the proposed QAS metric. The experiments aim to assess the behavior and effectiveness of the metric across multiple dimensions, including its sensitivity to the weighting parameter, its distributional properties, its relationship to existing metrics, its computational requirements, and its performance on specifically designed edge cases.

The experiments were conducted using the *dev* branch of the BIRD benchmark [97]. BIRD was selected due to its optimal balance between realistic size, complexity, and representativeness, which ensures the robustness of the experimental evaluation. As one of the benchmarks most closely approximating real-world scenarios for the text-to-SQL task, BIRD provides an appropriate testbed for assessing metric performance.

The evaluation framework considered outputs from 11 diverse text-to-SQL models of five different families: (i) C3 SQL [41], (ii) DAIRSQL and DAIRSQL SC [46], (iii) RESDSQL 3 EK, RESDSQL Base EK, and RESDSQL Large EK [95], (iv) SFT CodeS 1B EK, SFT CodeS 3B EK, SFT CodeS 7B EK, and SFT CodeS 15B EK [96], and (v) SuperSQL [93]. These models represent a range of architectural paradigms, including fine-tuned models and ICL implementations, providing a comprehensive basis for evaluating how the proposed metric performs across different model types. The outputs of these models were obtained from publicly available results in GitHub repositories.

Baseline performance was assessed using the EM and EX metrics, implemented via the official BIRD repository. For semantic similarity computation, two specialized code embedding models were evaluated:

- UAE-Code-Large-V1 [99]: Selected for its innovative approach that reduces the negative impact of the saturation zone of the cosine function, representing the state-of-the-art for code embeddings.
- all-MiniLM-RAGSQL-code: Included due to its popularity and proven effectiveness in generating embeddings.

These embedding models are robust tools for interpreting complex queries, leveraging their ability to map words and phrases into dense vector representations for semantic understanding. They employ positional encoding and attention mechanisms to achieve syntactic awareness, learning the structure and relationships within queries.

Since there is no a priori relative importance to assign to semantic and table similarity, we performed initial testing for the weight parameter w ranging from 0 to 1 with a step of 0.25. The QAS scores obtained for different values of w are not directly comparable as they represent different metrics. However, comparing the induced rankings and distributions of values helps understand the influence of w and select a reasonable value for subsequent experiments. After preliminary evaluation, the UAE-Code-Large-V1 model was selected for its superior performance in interpreting SQL queries. This model excelled in capturing the nuances of query structure and semantics and spaced query vectors more effectively in the vector space, resulting in a more distributed range of similarity scores between 0 and 1.

The experiments were conducted on hardware consisting of an NVIDIA A100 GPU (limited to 20 GB of VRAM), 8 CPU cores, and 128 GB of RAM. Computational efficiency was a critical consideration, as the time required for metric calculations may significantly impact the practical applicability of the proposed approach.

Based on the experimental setup described above, a series of evaluations were conducted to assess different aspects of the proposed metric. A parameter sensitivity study examined how varying the value of w influences the resulting rankings and similarity scores, helping to determine an appropriate balance between semantic and table similarity. Distribution analysis investigated the patterns of semantic similarity, table similarity, and the combined QAS across outputs from all considered models. The metric was also compared with existing measures, including EM and EX, to highlight differences in behavior, particularly in capturing partial correctness and structural variations in queries. Computational efficiency was evaluated by measuring the time required to compute the

5. QUERY AFFINITY SCORE

metric for queries of different sizes, assessing its practical feasibility. Finally, edge case analysis involved targeted experiments on specially constructed queries to test the metric's sensitivity to specific modifications, such as changes in row or column counts or alterations in ordering clauses.

5.4.1 Evaluation Methodology for Edge Cases

To verify that the proposed metric behaves as expected and properly reflects query differences, we created a set of specially designed test queries. These queries were derived from a reference query on the `california_school` database provided by the BIRD benchmark. The reference query was selected because it includes a diverse set of operators, returning a single-row, single-column table due to the presence of a limit clause. To measure all components of the proposed metric, the reference query was rewritten by replacing the `JOIN` operation with a subquery. This modification preserves identical results while introducing slight differences in query text, allowing greater control over query variants. Using a subquery makes it possible to modify both the number of rows and the sorting of the results independently.

Six groups of query variants were created, each containing 4 variations of the reference query, resulting in 24 test queries:

1. **Column Number:** This group evaluates the metric as the number of columns increases. Additional columns are incrementally added, selected iteratively from available columns in the database. We expect the metric to decrease monotonically as more columns are added, since additional columns increase the divergence from the reference output.
2. **Row Number:** This group examines the metric behavior with increasing rows. Rows are added incrementally from 1 to 4 by adjusting the argument of the `LIMIT` operator. Again, we expect decreasing similarity scores as the number of rows diverges from the reference.
3. **Columns with Additional Rows:** This group investigates the metric response when the number of columns increases from 1 to 4 while keeping the number of rows fixed at 4. This tests the interaction between row and column differences.
4. **Rows with Additional Columns:** This group evaluates the metric as the number of rows changes while keeping the number of columns fixed at 4. This provides the complementary test to the previous group.

5. **Rows and Columns Combined:** In this group, both rows and columns are incrementally increased together, starting from one row and one column and progressing to four rows and four columns. This tests cumulative divergence.
6. **Rows with Different Ordering:** This group explores how ordering affects the metric. Variants are created by changing the `ORDER BY` clause from `DESC` to `ASC` and incrementally increasing the number of rows from 1 to 4. This is important because changing the sort order can produce completely different result sets when combined with `LIMIT`.

These test cases are crucial for ensuring that the metric behaves as intended. As a query diverges from the reference, the metric should produce proportionally lower scores. Additionally, the results should reflect a balanced weighting, so that both semantic and syntactic aspects as well as execution outcomes, contribute appropriately to the final score.

5.5 Results

This section presents the analysis of experimental results. The findings are organized according to the evaluation dimensions outlined in Section 5.4.

5.5.1 Selection of the Parameter w

To systematically characterize the sensitivity of the weighting parameter w , a comprehensive comparative analysis was conducted across multiple values. The objective was to identify a configuration that achieves a balanced integration of semantic and table similarities. Rankings generated under different w values were compared using the Kendall distance [83], a widely adopted metric for quantifying the degree of dissimilarity between permutations. Higher Kendall distances reflect greater discrepancy between rankings, while lower values indicate stronger concordance as shown in Figure 5.2.

The analysis in Figure 5.3 revealed consistent patterns across all models. The distances were larger when comparing rankings generated with $w = 0$ and $w = 1$, as these two settings emphasize exclusively S_T and S_C , respectively. Intermediate values of w (0.25, 0.50, and 0.75) yielded much smaller distances when compared among themselves, indicating smoother and more gradual changes in the resulting rankings. These observations suggest that when w lies in the intermediate range, the ranking results are more stable, reflecting a balanced contribution from both semantic content

5. QUERY AFFINITY SCORE

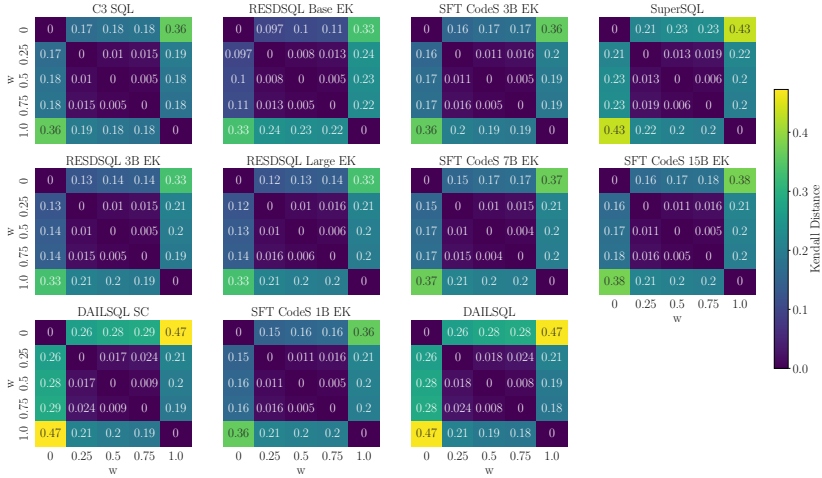


Figure 5.2: Pairwise Kendall distances between model rankings induced by different values of the weighting parameter w . Each cell (i, j) represents the Kendall distance between the ranking produced with $w = w_i$ and the ranking produced with $w = w_j$. The matrix is symmetric, with zeros on the diagonal. Results are shown separately for each text-to-SQL model (one panel per model). Lower values indicate greater ranking agreement between the corresponding weight configurations.

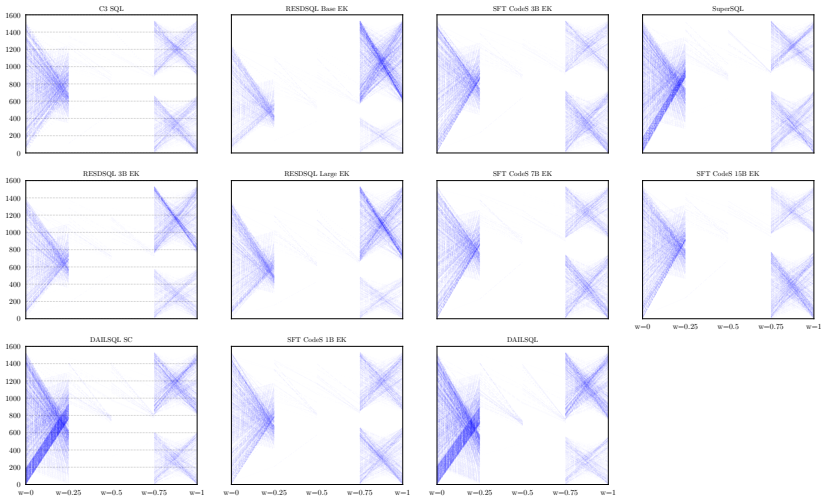


Figure 5.3: Query ranking shifts across different values of the weighting parameter w . Lines connect the same query across w values, with darker blue indicating larger rank changes.

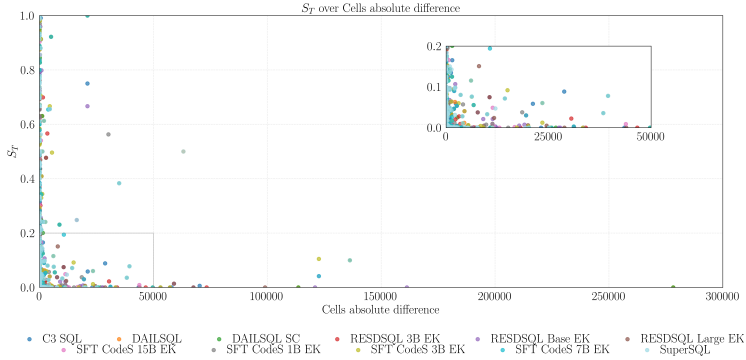


Figure 5.4: Correlation between S_T and the difference in the number of cells between the two tables.

and table structure. Significant ranking variations occurred primarily in the transitions from $w = 0$ to $w = 0.25$ and from $w = 0.75$ to $w = 1$, corresponding to the extremes where a single similarity component dominates. Rankings appeared relatively stable across the intermediate values ($w = 0.25, 0.5$, and 0.75), supporting the view that these configurations promote smoother and more balanced adjustment.

Based on this analysis, $w = 0.5$ was selected for subsequent experiments to equally weight the two components without privileging one aspect over the other.

5.5.2 Distribution of Table Similarity Score (S_T)

To emphasize the significance of the proposed metric, the distribution of S_T scores was analyzed against the absolute difference in the number of cells between reference and generated tables. The results in Figure 5.4 demonstrate that merely comparing the shapes of tables provides no meaningful insight into their actual similarity.

Tables with minimal or no difference in the number of cells exhibited S_T scores spanning the entire range $[0, 1]$. This variability highlights the absence of a consistent relationship between S_T and the difference in cell count. This pattern was consistent across all models. When the difference in the number of cells exceeded 50000, the similarity score was consistently zero or near zero. However, such extreme differences were rare, as most tables had comparable sizes but exhibited a wide range of similarity values. This finding underscores the importance of developing a metric that accounts for additional aspects beyond simple shape comparison. Simpler methods, such as comparing the difference in the number of cells, fail to provide sufficient information about query similarity. The S_T

5. QUERY AFFINITY SCORE

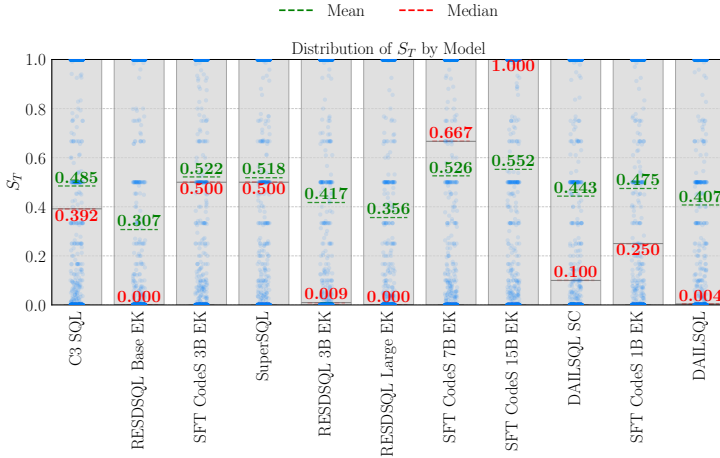


Figure 5.5: Boxplot showing the distribution of S_T scores for each of the 11 models.

score offers significant advantages over traditional binary metrics. Unlike binary metrics, the proposed approach generates well-distributed similarity scores across the $[0, 1]$ range, allowing for more nuanced and comprehensive evaluation.

Analysis of the S_T score distributions in Figure 5.5 across models revealed important patterns. In many models, the median was not at the extreme values (0 or 1) but positioned closer to the middle of the distribution. This highlights how current metrics like EX often classify many results as incorrect even when they are partially correct, or vice versa. It suggests that erroneous results may still contain information relevant to the natural language request.

The increased granularity allows more nuanced and accurate comparisons between models. A similarity value within the set of real numbers is far more informative than a binary value. With non-binary similarity scores, both mean and median can be calculated more accurately, offering a more comprehensive representation of model capabilities.

For example, although the C3 SQL and SFT CodeS 1B EK models exhibited similar average performance, SFT CodeS 1B EK had a significantly lower median performance, making C3 SQL the better model of the two. Similarly, while SFT CodeS 7B EK and SFT CodeS 15B EK showed comparable mean performance, SFT CodeS 15B EK demonstrated a higher median accuracy, making it the better-performing model overall. This observation is consistent with model design expectations, as SFT CodeS 7B EK has fewer parameters than SFT CodeS 15B EK.

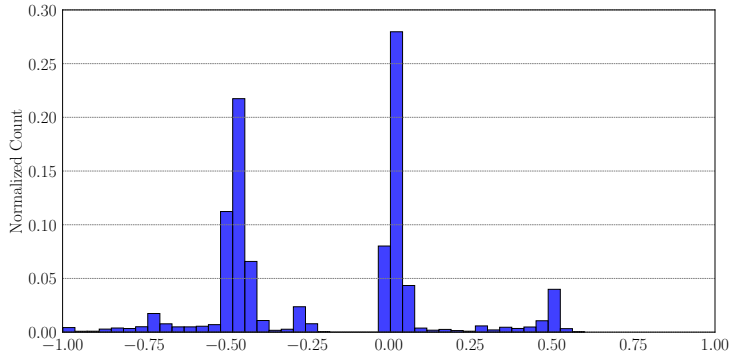


Figure 5.6: Comparison of the proposed approach with EX. The x-axis represents the magnitude of the difference between EX and the proposed metric.

5.5.3 Comparison with Existing Metrics

The comparative analysis between EX and the proposed metric revealed several significant patterns. The distribution of differences between EX and QAS, calculated as $(EX - QAS)$, showed that the majority of differences were concentrated around 0, with a secondary peak near 0.5.

The portion of the distribution in Figure 5.6 highlights the binary nature of EX and reveals instances where it mistakenly assigns perfect scores. Such cases typically occur when generated tables include duplicate values that correspond to a single correct entry.

The negative skew in the distribution, centered around -0.5 , illustrates EX's tendency to assign zero scores in cases where the proposed metric recognizes partial correctness. A tail extending from -0.5 to -1 was also evident, highlighting EX's tendency to assign zero scores to partially correct results.

5.5.4 Distribution of Query Affinity Score (QAS)

The QAS distributions for each model showed a significant increase in mean and median values compared to distributions based exclusively on S_T .

This trend in Figure 5.7 suggests that models generate queries that are semantically well-aligned with those in the BIRD benchmark. Adding the semantic similarity component did not lead to significant changes in the ranking of models. All distributions consistently included values above 0.2, indicating that while many queries produce significantly different tables, their semantics remain closely aligned with reference queries.

5. QUERY AFFINITY SCORE

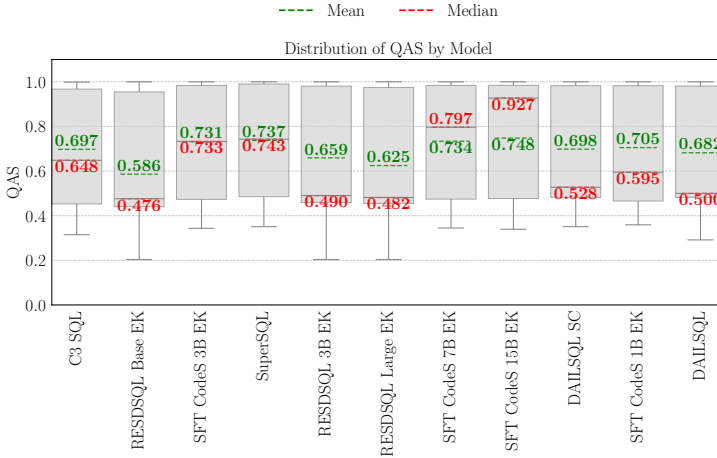


Figure 5.7: QAS distribution across all the models.

An important insight from the analysis is the variability of scores for each model. Although some models (RESDSQL 3B EK, RESDSQL Large EK, and DAILSQL) exhibited similar means and medians indicating comparable overall performance, closer analysis of their variability revealed differences in robustness. The DAILSQL model demonstrated a narrower range of variability compared to RESDSQL 3B EK and RESDSQL Large EK. In particular, DAILSQL maintained a minimum score of 0.3, whereas the other two models reached a lower minimum of 0.2. This reduced variability and higher minimum performance highlight the greater consistency and robustness of DAILSQL, making it the most reliable option among the three.

5.5.5 Time Analysis

The computation time required to calculate S_T scores was analyzed as the total number of cells in tables increased. For the vast majority of queries, the table similarity calculation took less than a minute, primarily because the number of cells was typically fewer than 1000 as shown in Figure 5.8.

However, rare instances were identified where processing times extended up to 4 hours, with a single outlier requiring 7.3 hours. As expected, computation time increased linearly with the growth in the number of cells. This linear relationship is due to the design of the S_T metric, which compares rows and columns. As the number of rows and

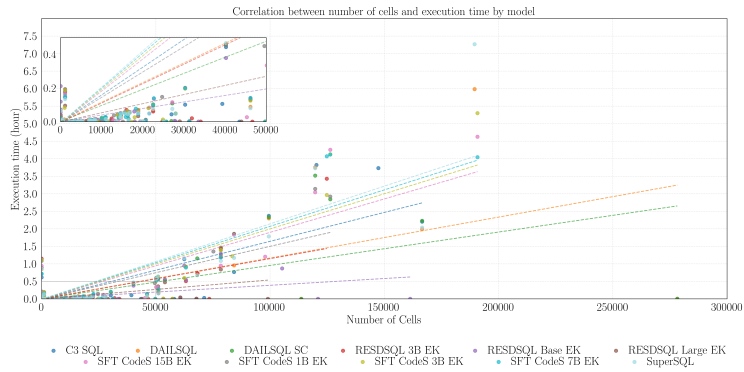


Figure 5.8: Graph showing the computation times for calculating the S_T metric. The y-axis represents time in minutes, where each increment of 0.5 corresponds to 30 minutes.

columns increases, the number of comparisons grows proportionally.

Cases with execution times close to zero despite large numbers of cells corresponded to situations where the SQL query strings were identical. In these instances, table similarity equals 1 by definition, eliminating the need for comparisons.

5.5.6 Edge Cases Analysis and Particular SQL Scenarios

The analysis of specially constructed query variants revealed important insights about the metric's behavior.

```
SELECT T2.MailStreet
FROM frpm AS T1 INNER JOIN schools AS T2 ON T1.CDSCode = T2.CDSCode
ORDER BY T1.`FRPM Count (K-12)` DESC
LIMIT 1;
```

Query 1: Reference query for edge cases analysis.

The experiments in Figure 5.9 demonstrated that semantic similarity and table execution results, on their own, are not sufficient as evaluation criteria.

These findings justify why the proposed metric combines semantic and table similarities in a weighted manner to compute the QAS. Neither component alone provides a complete picture of query similarity. Further details about the SQL queries in groups Column Number (group 1), Row Number (group 2), Columns with Additional Rows (group

5. QUERY AFFINITY SCORE

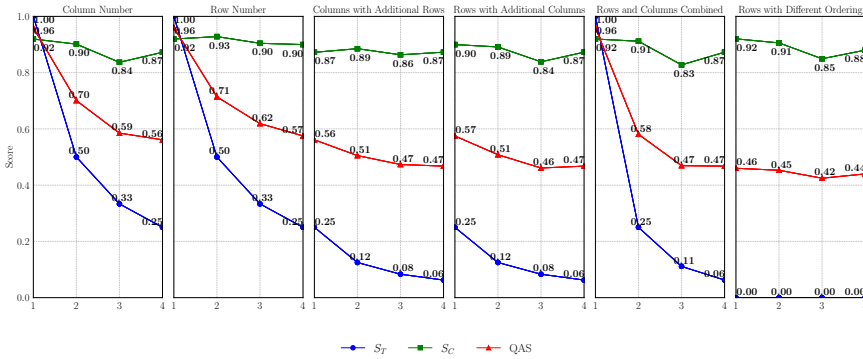


Figure 5.9: Graphs illustrating the trend of the QAS and its partial component scores, S_T and S_C . The x-axis represents the i -th variation of Query 1 within each specific group

3), Rows with Additional Columns (group 4), Rows and Columns Combined (group 5), and Rows with Different Ordering (group 6) are provided in Appendix B.

In all groups except Row Number (group 2), the S_C metric yielded unexpectedly higher scores for the fourth SQL query variation compared to the third, despite the fourth variation deviating more significantly from the reference query. This behavior was evident from rising trends in the S_C score between the third and fourth positions. In contrast, the S_T score either continued to decrease or remained at zero. Since these queries were specifically constructed with known properties, we know that the first five groups contain queries that progressively deviate further from the reference. However, this difference was not effectively captured by the pipeline component responsible for calculating S_C .

Conversely, the S_T score in the Rows with Different Ordering group (group 6) consistently returned zero, even for meaningful structural variations such as changing the order from ASC to DESC. This highlights the need for a more nuanced scoring approach that combines both similarity components.

These findings justify why the proposed metric combines semantic and table similarities in a weighted manner to compute the QAS. Neither component alone provides a complete picture of query similarity.

5.5.7 Error Identification in SQL Queries

A significant advantage of the metric is its ability to provide insights about possible key differences between SQL queries. Several examples illustrate this capability.

Query 2 is designed to evaluate the metric's ability to capture semantic representation with aggregation operators.

```
SELECT DISTINCT(t1.player_name)
FROM Player AS t1 INNER JOIN Player_Attributes AS t2
  ON t1.player_api_id = t2.player_api_id
WHERE t2.potential = (
  SELECT MAX(potential)
  FROM Player_Attributes
);
```

Query 2: Reference query with DISTINCT aggregation operator.

```
SELECT player.player_name
FROM player JOIN player_attributes
  ON player.player_api_id = player_attributes.player_api_id
WHERE player_attributes.potential = (
  SELECT MAX(potential)
  FROM player_attributes
);
```

Query 3: Generated query without aggregation operator.

We evaluated the proposed metric on Query 2 and the generated Query 3. The results show $S_C = 0.95$, $S_T = 0.16$, and $QAS = 0.55$. Analyzing S_T and S_C separately reveals that S_T is significantly lower than S_C , indicating that while the queries are semantically similar, their execution results differ considerably. This discrepancy highlights the substantial impact that seemingly minor differences, such as missing aggregation operators, can have on execution results.

A similar observation applies to Query 4 and the generated Query 5.

```
SELECT T2.Zip
FROM frpm AS T1 INNER JOIN schools AS T2 ON T1.CDSCode = T2.CDSCode
WHERE T1.`District Name` = 'Fresno County Office of Education'
AND T1.`Charter School (Y/N)` = 1;
```

Query 4: Reference query with JOIN operator between tables.

```
SELECT zip
FROM schools
```

5. QUERY AFFINITY SCORE

```
WHERE charter = 1
AND district = 'Fresno County Office of Education';
```

Query 5: Generated query involving a single table.

Here, the S_T score is again notably lower than S_C , which can be attributed to the missing JOIN operation. Although both queries include a SELECT statement for the zip code, the results differ substantially due to the missing JOIN.

While the metric may not always provide precise explanations or pinpoint errors with absolute certainty, it remains a valuable tool for identifying potential issues. In contrast, traditional metrics such as EX and EM fail to offer meaningful insights. Across all examined cases, these metrics consistently returned 0, providing no means to detect or analyze errors or any indication of their severity.

5

5.6 Discussion

The experimental results presented in Section 5.5 yield several important insights and enable us to address the research questions posed in the introduction.

The first research question asked whether a continuous, non-binary metric can provide more informative assessments of SQL query similarity compared to existing binary approaches. The answer is yes, as moving from a binary scoring system to a more granular similarity measure enables a finer-grained and more informative evaluation of generated queries, capturing partial correctness and nuanced differences that a simple pass would otherwise obscure —fail assessment. The comparison between EX and the proposed metric demonstrated that EX frequently assigns zero scores to queries that the proposed metric recognizes as partially correct. This was evidenced by the distribution of differences between the two metrics, which showed a substantial negative skew centered around -0.5 . The granular nature of the proposed metric enables more precise comparisons between text-to-SQL models. When comparing models with similar mean performance, the proposed metric revealed differences in median performance and variability that would be invisible to binary metrics. For instance, the distinction between C3 SQL and SFT CodeS 1B EK based on median performance, or between SFT CodeS 7B EK and SFT CodeS 15B EK based on consistency, could only be made because the metric provides continuous scores.

These findings were expected given the theoretical limitations of binary metrics. Any metric that maps the rich space of possible query differences to a single binary value

must inevitably lose information. The proposed metric preserves this information, enabling richer analyses and more nuanced conclusions.

c5RQ1 Answer

Yes, continuous metrics provide substantially more informative assessments than binary approaches. The QAS captures partial correctness and nuanced differences that EX obscures, with the distribution of $(EX - QAS)$ differences showing substantial negative skew centered around -0.5 . Granular scores enable differentiation between models with similar mean performance based on median values, variability, and minimum scores—distinctions invisible to binary metrics.

The second research question concerned how semantic similarity and execution result similarity contribute individually and jointly to the overall assessment. The analysis of parameter w and the edge case experiments provide clear answers. The Kendall distance analysis showed that extreme values of w (0 and 1) produce substantially different rankings compared to intermediate values. This indicates that the two components capture genuinely different aspects of query quality. The relative stability of rankings across intermediate values ($w = 0.25$ to 0.75) suggests that both components provide complementary information that smoothly integrates when combined. The edge case analysis revealed specific situations where the components behave differently. In the Rows with Different Ordering group, S_T consistently returned zero because changing sort order produces completely different result sets when combined with LIMIT. Meanwhile, S_C remained high because the queries were structurally similar. This illustrates how S_T can detect functional differences that S_C misses. Conversely, in other groups, S_C sometimes yielded unexpectedly high scores for queries that were known to deviate significantly from the reference. This suggests limitations in the embedding model's ability to capture certain types of query differences. The table similarity component helps correct for these cases by evaluating actual execution outcomes. The decision to weight both components equally ($w = 0.5$) in the main experiments reflects the complementary nature of these contributions. Neither component alone is sufficient, and their combination provides a more complete assessment.

5. QUERY AFFINITY SCORE

c5RQ2 Answer

Semantic similarity (S_C) and table similarity (S_T) capture genuinely different aspects of query quality. Kendall distance analysis shows extreme w values (0 and 1) produce substantially different rankings, while intermediate values yield stable rankings. Edge cases reveal complementary behavior: S_T detects functional differences (e.g., ordering changes) that S_C misses, while S_C captures structural similarity when S_T returns zero. Neither component alone suffices; their combination ($w = 0.5$) provides comprehensive assessment.

5

The third research question asked whether the proposed metric can effectively differentiate between text-to-SQL models and provide insights into their relative strengths and weaknesses. The results demonstrate that it can. The QAS distributions clearly differentiated among the 11 evaluated models. Beyond simple mean comparisons, the metric enabled analysis of distribution characteristics such as median values, variability, and minimum scores. These characteristics revealed important distinctions. For example, DAILSQL demonstrated greater consistency and robustness than RESDSQL₃ EK and RESDSQL_{Large} EK despite similar average performance, maintaining a higher minimum score and narrower variability range. The metric also enabled verification of expected relationships between models. The observation that SFT_{CodeS}_{15B} EK outperformed SFT_{CodeS}_{7B} EK is consistent with the expectation that larger models in the same family would achieve better results. Binary metrics could confirm this relationship only in aggregate; the proposed metric provides more detailed evidence. These findings have practical implications. When selecting a model for real-world applications, understanding not just average performance but also consistency and worst-case behavior is crucial. The proposed metric provides the granularity needed for such assessments.

c5RQ3 Answer

The QAS effectively differentiates among 11 text-to-SQL models from 5 families. Beyond mean comparisons, distribution analysis reveals consistency and robustness differences: DAIRSQL maintains higher minimum scores and narrower variability than models with similar averages (RESDSL 3 EK, RESDSL Large EK). The metric confirms expected relationships (e.g., SFT CodeS 15B EK outperforming SFT CodeS 7B EK) with more detailed evidence than binary aggregates provide.

The fourth research question concerned the extent to which the metric can identify specific types of errors in generated queries. The results show that the metric provides meaningful signals about error types, though with limitations. The pattern of S_T being significantly lower than S_C was consistently associated with specific error types: missing aggregation operators, opposite sorting orders, and different filtering conditions. In these cases, queries are semantically similar (similar structure and intent) but produce different results due to subtle differences. This pattern provides a diagnostic signal that can guide error investigation. However, interpreting query errors based only on metric scores can be challenging. Different error types can produce similar score patterns. For instance, a low S_T with high S_C might indicate a missing `DISTINCT` operator, an incorrect `ORDER BY` direction, or a missing `JOIN`. Additional analysis is needed to pinpoint the exact issue. Despite these limitations, the proposed metric represents a significant improvement over existing approaches. Traditional metrics like `EM` and `EX` provide no diagnostic information whatsoever. They return 0 for all incorrect queries regardless of how close those queries are to being correct or what type of error they contain. The proposed metric at least provides signals about whether the problem lies in the semantic structure or the execution results, narrowing the search space for error identification.

c5RQ4 Answer

The metric provides meaningful diagnostic signals about error types, though with limitations. A consistent pattern of $S_T \ll S_C$ indicates specific errors: missing aggregation operators, incorrect sorting orders, or different filtering conditions—queries semantically similar but functionally different. While the metric cannot pinpoint exact errors (different types produce similar patterns), it narrows error identification scope significantly beyond EM and EX, which provide no diagnostic information whatsoever.

5.6.1 Potential Applications

One potential advantage of using granular metrics lies in their ability to deliver more precise feedback during model training. Traditional binary metrics provide limited guidance, labeling a generated query as either entirely correct or entirely incorrect. This all-or-nothing feedback can impede learning, particularly for nearly correct queries. The proposed metric can be used as a loss function or integrated into training objectives for text-to-SQL models. Conveying gradient information that reflects the degree of correctness, it allows for more nuanced optimization. Models can receive partial credit for approaching the correct answer, potentially accelerating learning and improving overall performance.

Another application lies in model selection and evaluation. The proposed metric enables more sophisticated comparisons between models by revealing characteristics that binary metrics obscure. Practitioners can use these insights to select models that best match their requirements, whether those requirements emphasize average performance, consistency, or worst-case behavior.

5.6.2 Limitations

The proposed approach has limitations, particularly regarding computational requirements. Calculating similarity between tables with large numbers of rows and columns can be time and memory-demanding. During experiments, 49 instances across all models could not compute a similarity score due to resource limitations. The linear relationship between computation time and cell count means that queries producing very large result tables may be impractical to evaluate. Future work should address this limitation through more efficient algorithms or approximate methods. Additionally, while the met-

ric provides signals about error types, it cannot definitively identify specific errors. Human analysis remains necessary to interpret the signals and understand exactly what went wrong in a generated query.

5.7 Conclusions

This chapter has presented a novel metric for evaluating text-to-SQL systems that addresses fundamental limitations of existing evaluation approaches. By integrating semantic similarity and execution result similarity into the QAS, the proposed metric provides granular, continuous assessments of SQL query quality that capture nuances invisible to traditional binary metrics.

The key contributions of this work include:

1. A principled approach to measuring SQL query similarity that considers both the structural intent of queries (through code-specific embeddings) and their functional outcomes (through edit distance-based table comparison).
2. Empirical demonstration that continuous metrics provide richer information than binary ones, enabling more nuanced model comparisons and revealing distinctions in consistency and robustness.
3. Evidence that the proposed metric can provide diagnostic signals about error types, helping to identify whether problems stem from semantic structure or execution results.
4. A configurable weighting parameter that allows the metric to be adapted to different evaluation priorities.

The importance of these contributions extends to both research and practical applications. For researchers, more accurate evaluation metrics are essential for understanding where models struggle and how to improve them. The proposed metric enables targeted improvement efforts by providing fine-grained feedback about query quality. For practitioners, reliable evaluation is crucial for deciding whether to deploy text-to-SQL systems in real-world applications. The proposed metric provides a more realistic assessment of system capabilities compared with binary measures, which can either overestimate or underestimate performance. Its adoption could influence both the scientific community and practical applications in several ways. It offers researchers more informative signals for identifying model weaknesses and guiding architectural improvements. It can also be used as a loss function during training, delivering precise feedback that helps

5. QUERY AFFINITY SCORE

models learn to generate accurate queries. Additionally, it supports more meaningful benchmarking of text-to-SQL systems, facilitating informed decisions regarding model selection and deployment.

Future work will focus on addressing the computational limitations of the current approach. Developing a more relaxed version of the metric to improve efficiency and reduce resource demands is a priority. Additionally, enhancing and parallelizing the computation of table distances could enable practical application to queries with larger result sets. The democratization of database access through natural language interfaces represents an important technological frontier. As text-to-SQL systems continue to improve, rigorous evaluation becomes increasingly critical for ensuring that these systems meet the reliability requirements of real-world applications. The metric proposed in this chapter contributes to this goal by providing the evaluation granularity needed to distinguish truly capable systems from those that merely appear successful under coarse-grained assessment.

III

Sustainability and Green AI

6

Green and Effective Feature Selection toward Hotfix Bug Taxonomy

This chapter presents HotCat, a pipeline that employs multi-objective optimization to balance classification accuracy with computational efficiency in software bug taxonomy. By leveraging NSGA-II for feature selection on 88 hotfix records across 17 categories, we achieve 63% accuracy in 132 seconds, demonstrating that higher accuracy can be obtained without increasing resource consumption. This investigation directly addresses the thesis research question RQ4 (sustainable LLM deployment) within the domain of software maintenance, contributing to the broader green AI agenda.

Authors: Luis de la Cal, Yazhuo Cao, Irmak Ercevik, Giovanni Pinna, Luke Twist, David Williams, Karine Even-Mendoza, William B. Langdon, Hector D. Menendez, Federica Sarro

Original title: HotCat: Green and Effective Feature Selection for HotFix Bug Taxonomy

Published in: The 17th Symposium on Search-Based Software Engineering 2025 (SSBSE 2025): SSBSE 2025 Challenge Case: Hot Fixing Benchmark. 2025.

<https://kclpure.kcl.ac.uk/portal/en/publications/hotcat-green-and-effective-feature-selection-for-hotfix-bug-taxon/>

Candidate's contribution: The candidate contributed to the conception of the project, the design of the experiments, the analysis and interpretation of the results, and the writing of parts of the manuscript, as well as revising the manuscript.

6.1 Introduction

Software maintenance represents a critical phase in the SD lifecycle, consuming a substantial portion of overall development resources. Within this domain, hotfixes constitute a particularly challenging category of software patches that address time-sensitive issues requiring immediate deployment to production systems [75]. Unlike regular maintenance patches that follow standard release cycles, hotfixes must balance the urgency of rapid deployment with the fundamental requirement of system stability, creating inherent tensions between time criticality and code quality assurance [57].

The systematic classification and organization of bugs into coherent taxonomies has long been recognized as essential for effective software maintenance [24]. A well-defined bug taxonomy enables development teams to prioritize resources, identify recurring patterns, and implement targeted preventive measures. However, constructing such taxonomies for hotfixes presents unique challenges. First, hotfix data is typically sparse and imbalanced, as urgent patches represent only a small fraction of overall software changes [26]. Second, the semantic analysis required to capture edit intent and classify patches into meaningful categories traditionally demands substantial computational resources, particularly when leveraging LLM.

This chapter addresses a critical concern that permeates contemporary AI research: the tension between model capability and computational sustainability. As discussed in Chapter 1, the remarkable capabilities of LLMs come at substantial computational cost. The environmental impact of AI, examined in Section 2.2.3, has emerged as a pressing concern within the research community, with the green AI movement advocating for approaches that prioritize computational efficiency alongside accuracy [137]. This perspective is particularly relevant for LLM-based analyses, where training and inference costs can be substantial.

In this context, feature selection emerges as a promising strategy for reducing computational overhead while maintaining classification effectiveness. By identifying and retaining only the most informative features, it becomes possible to decrease the dimensionality of input data, thereby reducing both processing time and energy consumption. This approach directly addresses the thesis research question RQ4, which asks how we can develop and deploy LLM-based systems while minimizing their environmental footprint. Building upon the MOO foundations established in Section 2.2.3, we employ NSGA-II [37] to balance multiple competing objectives such as classification accuracy and computational efficiency.

6. GREEN FEATURE SELECTION

This chapter presents HotCat, an automated pipeline that extends PatchCat [43] for hotfix analysis. HotCat integrates project metadata with MOO to construct a taxonomy of time-critical software bugs. The approach leverages Sentence-BERT embeddings [130]—which derive from the Transformer architectures discussed in Section 2.1—with short-text clustering to classify hotfixes into semantic categories while employing NSGA-II to identify feature subsets that balance predictive accuracy with computational efficiency. This work demonstrates how the evolutionary computation techniques introduced in Section 2.2 can be applied to reduce the environmental footprint of LLM-based analysis pipelines.

6.1.1 Research Questions

This chapter contributes to the thesis research question RQ4 (sustainable LLM deployment) by addressing the following chapter-specific research questions:

c6RQ1: *What is the trade-off between classification performance and runtime when classifying hotfixes using HotCat?*

This question investigates the non-dominated configurations that emerge from MOO, examining how different feature subsets affect both the quality of bug classification and the computational resources required. Understanding these trade-offs is essential for practitioners who must balance analytical accuracy against resource constraints, directly contributing to RQ4.

c6RQ2: *Do the predicted bug categories improve real maintenance workflows?*

This question evaluates the practical utility of HotCat’s taxonomy by assessing whether the automatically generated classifications align with manual expert annotations and whether they generalize to previously unseen hotfix records. This evaluation is critical for determining whether sustainable approaches can provide actionable insights without sacrificing practical utility.

The primary contributions of this chapter are:

- An adaptation of PatchCat to operate directly on hotfix patches from the HotBugs.jar dataset [57], integrating project metadata and LLM-based summarization for comprehensive bug characterization.
- A methodology addressing data sparsity and class imbalance through centroid-based description, LLM-generated augmentation, and semantic enrichment techniques.

- The application of NSGA-II for feature selection, demonstrating how MOO can balance classification quality with computational efficiency and responsible resource usage, thereby contributing practical strategies for green AI deployment.
- Empirical evidence that extending PatchCat with search-based optimization represents a viable approach toward automatically extracted hotfix bug taxonomies while respecting sustainability constraints.

6.2 Related Work

This section contextualizes HotCat within the intersection of hotfix practices, defect taxonomy, and sustainable SE. Building upon the MOO and green AI foundations established in Section 2.2.3, we focus here on domain-specific developments in software maintenance and patch analysis.

HOTFIX PRACTICES AND DEFECT CLASSIFICATION. Prior work on runtime patching emphasizes the critical trade-off between urgency and system stability [75], noting that while hotfixes are essential for severe crash bugs, they carry significant risks of human error and instability [19, 171]. To manage such defects effectively, classification frameworks have evolved from the manual Orthogonal Defect Classification [24] to automated, DL-based approaches capable of feature learning [153] and domain-specific fault categorization [72].

SEMANTIC PATCH ANALYSIS. The analysis of software patches has shifted from syntactic to semantic methods that better capture edit intent, exemplified by tools like SemFix [113]. Recent LLM-guided approaches such as PatchCat [43] utilize dense embeddings for clustering code changes, leveraging models like Sentence-BERT [130] and CodeBERT [45] that build upon the Transformer architectures discussed in Section 2.1.

SUSTAINABLE SOFTWARE ENGINEERING. These technical advancements must be contextualized within the green AI principles introduced in Section 2.2.3. Given the substantial environmental cost of training and deploying large models, recent studies highlight the necessity of sustainable strategies—such as feature selection [150] and MOO—to mitigate the carbon footprint of data-intensive SE tasks. HotCat addresses this need by employing NSGA-II to discover configurations that balance classification effectiveness with computational efficiency.

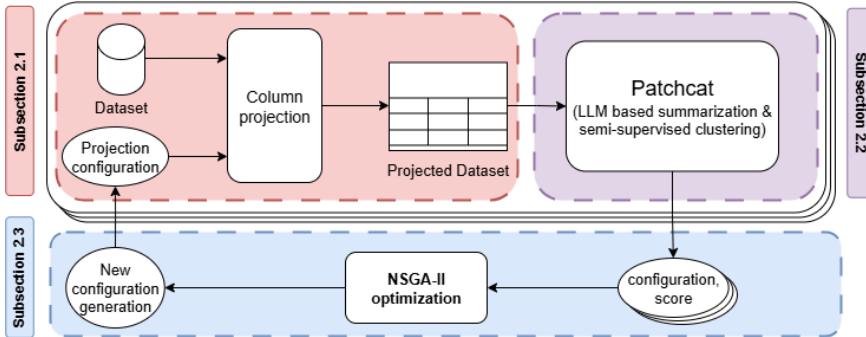


Figure 6.1: Architecture diagram of HotCat. The pipeline consists of three stages: dataset projection guided by a binary configuration bitmask (red, Section 6.3.1), PatchCat-based LLM summarization and semi-supervised clustering (purple, Section 6.3.2), and NSGA-II multi-objective optimization that evolves new configurations (blue, Section 6.3.3).

6.3 Methodology

The HotCat methodology introduces an automated pipeline designed to construct a taxonomy of time-critical software bugs by integrating dataset projection, LLM summarization, and MOO. The overall process, illustrated in Figure 6.1, comprises three interconnected stages: feature subset selection produces a projected dataset (Section 6.3.1), each projection is evaluated with PatchCat to assess classification performance (Section 6.3.2), and the resulting fitness scores are fed into NSGA-II to evolve improved generations of feature subsets (Section 6.3.3).

6.3.1 Dataset, Projected Dataset, and Configuration Definition

The foundation of this approach is the `HotBugs.jar` dataset [57], which is significantly enriched with multi-modal data including source code diffs, temporal metrics (time-to-fix), participant counts, and Jira metadata. To address the challenge of data sparsity within the 17 identified bug categories, the methodology employs a two-stage augmentation strategy involving category balancing and post-optimization record generation. A central tenet of HotCat is the rejection of using all available features blindly, as this increases computational noise and energy consumption; instead, the system defines a “configuration” as a binary bitmask that selectively includes or excludes specific fields from the 18 available attributes, aligning with green AI principles.

6.3.2 PatchCat and Fitness Function

The classification engine extends the existing PatchCat framework [43] to function as a fitness evaluator within an optimization loop. Unlike the original implementation, which relied solely on code diffs, this modified component accepts the extended feature subsets defined by the configuration bitmask. For each candidate configuration, the pipeline projects the dataset onto the selected features, utilizes an LLM to generate concise natural language summaries of the hotfixes, and subsequently computes semantic embeddings using Sentence-BERT. These embeddings are then organized via K-Means clustering with iterative refinement to produce the final classification. This process effectively transforms raw, heterogeneous project data into structured semantic clusters that capture the underlying intent of the hotfixes.

6.3.3 Multi-Objective Optimization

To identify the most effective feature combinations, the methodology formulates the configuration search as a MOO problem solved via the NSGA-II algorithm [37]. The optimization navigates a search space of 2^{18} possible configurations by simultaneously evaluating three competing objectives: maximizing classification Accuracy, maximizing Normalized Mutual Information (NMI) relative to ground truth labels, and minimizing computational Runtime. Through evolutionary operators such as binary crossover and bit-flip mutation, the algorithm approximates a Pareto front of non-dominated solutions. This allows for the identification of optimal trade-offs between the richness of the semantic representation and the computational efficiency required for sustainable analysis.

6.4 Experimental Evaluation

The experimental framework is implemented through a hybrid architecture combining Bash orchestration scripts and Python modules for core processing tasks. The optimization engine leverages the pymoo* [13] to execute the NSGA-II algorithm, while semantic analysis is supported by Sentence-BERT for embedding generation and Ollama for local LLM inference. To effectively balance search space exploration with computational constraints, the evolutionary algorithm is configured with a population size of 20 individuals evolved over 20 generations, utilizing binary crossover and bit-flip mutation as the pri-

*version 0.6.1.5

6. GREEN FEATURE SELECTION

mary genetic operators. Clustering operations follow the iterative refinement approach proposed by Rakib et al. [129].

Performance evaluation is based on two complementary metrics. Accuracy measures the direct correspondence between predicted labels and ground truth, while NMI quantifies the information-theoretic alignment between predicted clusters and the true categories. The assessment protocol is bifurcated into an initial optimization phase performed on the balanced dataset of 155 records, followed by a generalization phase designed to test robust applicability. In this final stage, the derived non-dominated models are evaluated on a distinct set of 51 manually annotated, challenging bug reports. To mitigate the stochastic nature of the LLM-based components, the generalization results are determined through an ensemble method using majority voting across five independent repetitions per configuration.

6

6.5 Results

This section presents the experimental results addressing the two research questions.

6.5.1 RQ1: Performance and Runtime Trade-off

Table 6.1 presents the non-dominated individuals obtained from NSGA-II optimization, and Figure 6.2 visualizes the three-dimensional Pareto front.

An important characteristic of the results is the presence of individuals with sub-identifiers (e.g., 1.1, 1.2, 1.3, 1.4). These represent *the same bitmask configuration* evaluated multiple times, yielding different fitness values due to the non-deterministic behavior of the LLM-based summarization component in the PatchCat pipeline. Since the LLM generates slightly different summaries for each invocation, the downstream clustering and classification results vary across repeated evaluations of the same feature subset. The notation $X.Y$ denotes individual X at repetition Y ; entries without a sub-identifier (e.g., individuals 2, 3, 6, 7, 9) represent configurations that appeared only once on the Pareto front. This stochastic variation underscores the importance of ensemble-based evaluation, which we adopt in RQ2.

The enumeration of results (e.g., 1.1, 1.2, 1.3, 1.4) reflects configurations sharing identical feature subsets but exhibiting different objective values across evaluations. These duplicates arise from the inherent stochasticity of the LLM-based pipeline: identical inputs may produce slightly different summaries across runs, leading to variations in embedding representations and subsequent clustering outcomes. With the limited evalua-

Table 6.1: Approximated Pareto front individuals with accuracy, NMI, and runtime metrics.

Individual	Accuracy (%)	NMI	Seconds
1.1	59	0.58	129
1.2	55	0.56	125
1.3	54	0.50	119
1.4	56	0.56	125
2	59	0.55	125
3	63	0.57	132
4.1	58	0.53	121
4.2	54	0.52	120
5.1	52	0.48	116
5.2	54	0.49	119
6	57	0.54	123
7	51	0.49	118
8.1	57	0.53	121
8.2	55	0.55	124
9	52	0.48	118

tion budget, NSGA-II may revisit similar regions of the search space, resulting in multiple non-dominated variants of essentially the same feature configuration. This behavior reflects both the noisy fitness landscape and the constrained optimization budget, as discussed further in Section 6.6.

It is important to note that the Pareto front individuals represent the non-dominated solutions discovered by the optimization process. The full bitmask configuration (all 18 features selected) was included in the initial NSGA-II population; however, due to the stochastic nature of the LLM-based pipeline, this configuration did not appear on the final Pareto front, indicating that configurations with fewer features consistently achieved comparable or superior trade-offs between accuracy, NMI, and runtime.

The results reveal several important findings:

Runtime Distribution: Individual 3 exhibits the slowest execution time at 132 seconds, while Individual 5.1 achieves the fastest runtime at 116 seconds—a difference of approximately 14%.

Accuracy Range: Individual 3 achieves the highest accuracy at 63%, whereas Individual 7 records the lowest at 51%. This range of 12 percentage points demonstrates the significant impact of feature selection on classification performance.

NMI Performance: Individual 1.1 obtains the best NMI score at 0.58, while Individual 5.1 yields the lowest at 0.48. Notably, the configurations achieving optimal accuracy

6

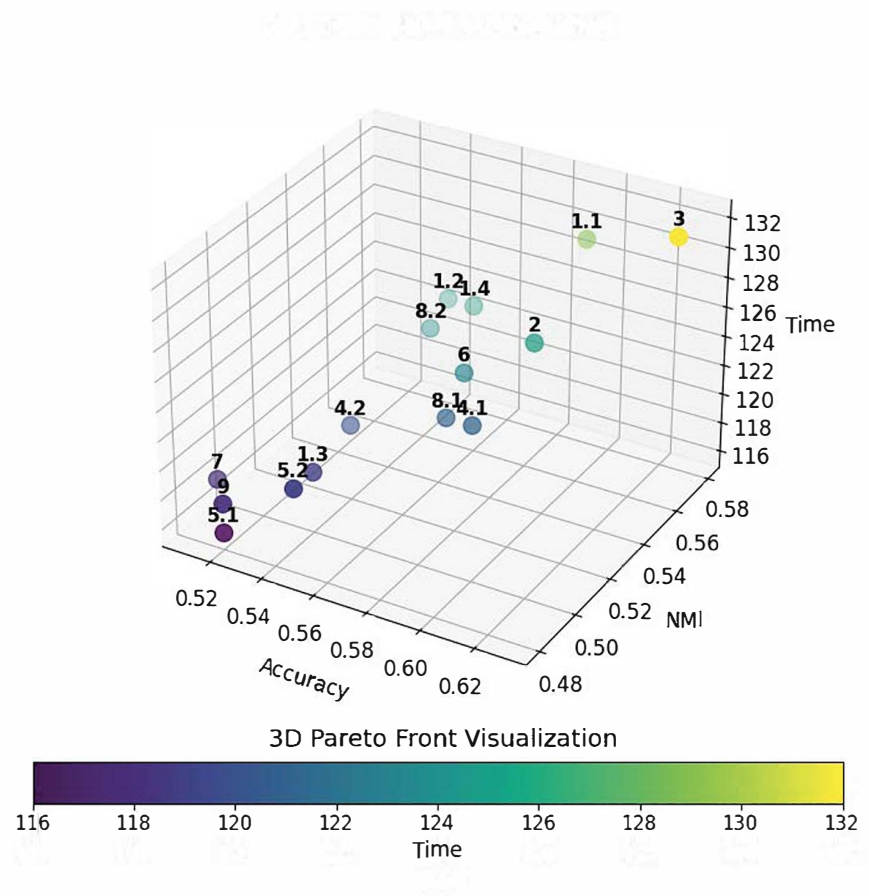


Figure 6.2: Three-dimensional visualization of the approximated Pareto front showing trade-offs between accuracy, NMI, and runtime.

and optimal NMI differ, reflecting the multi-objective nature of the problem.

Balanced Solutions: Individual 6 combines solid accuracy (57%) and NMI (0.54) with competitive runtime (123 seconds), representing a reasonable compromise for practitioners prioritizing balanced performance. Individual 1.1 emerges as the most balanced solution overall, achieving high accuracy (59%), the best NMI (0.58), and runtime close to the median (129 seconds).

6.5.2 RQ2: Generalization to Challenging Hotfixes

To evaluate generalization capability, we tested the non-dominated models on 51 manually annotated bug reports that were excluded from the original `HotBugs.jar` dataset due to their challenging nature. On the original dataset, the ensemble of non-dominated models achieved average accuracy of 55% and NMI of 0.52, representing moderate but meaningful classification capability. When evaluated on the augmented dataset (with 50 additional records per category), performance improved substantially to average accuracy of 72% and NMI of 0.75. The ensemble underperformed relative to training results on the challenging holdout set, with labels collapsing into a few dominant categories (notably categories 13 and 16). Analysis revealed two primary causes:

1. Small, imbalanced training set: With only 88 original records distributed across 17 categories, many categories contained insufficient examples for robust pattern learning.
2. Vocabulary overlap: Several bug categories share substantial semantic vocabulary, blurring fine-grained boundaries. For instance, memory leaks share terminology with performance issues and concurrency problems.

6.6 Discussion

The experimental results provide insights into both the capabilities and limitations of the `HotCat` approach, with implications for future research on sustainable bug taxonomy construction.

6.6.1 Performance-Runtime Trade-off

A central finding of this work is that, within the explored search space, accuracy improvements can be achieved without corresponding increases in resource consumption.

6. GREEN FEATURE SELECTION

Individual 3, which achieved the highest accuracy at 63%, required only 132 seconds—merely 3 seconds more than the best-balanced Individual 1.1. This observation supports the green AI principle that computational efficiency and model effectiveness need not be mutually exclusive [137], although the limited evaluation budget (20 individuals over 20 generations) means that the obtained Pareto front represents an approximation; further optimization may reveal additional trade-offs. The feature selection mechanism serves two key purposes, reducing the dimensionality of input data—which lowers LLM prompt sizes and inference costs—while also enhancing classification quality by removing noisy or redundant features. This aligns with findings from Verdecchia et al. [150] that feature reduction can yield substantial energy savings with minimal impact on predictive performance.

During optimization, runtime served as a proxy for computational efficiency.

c6RQ1 Answer

The best trade-off configuration (Individual 1.1) identified by NSGA-II achieved 59% accuracy and 0.58 NMI in 129 seconds.

The maximum accuracy of 63% (Individual 3) required only marginally longer runtime at 132 seconds, demonstrating that higher accuracy can be achieved without substantially increasing resource consumption.

These results represent the best configurations discovered within a limited evaluation budget and should be interpreted as promising trade-offs rather than globally optimal solutions.

6.6.2 Challenges of Small-Scale Hotfix Data

The limitations observed in RQ2 highlight fundamental challenges when constructing taxonomies from sparse, domain-specific data. With 88 records distributed across 17 categories, the average category size is approximately 5 examples—insufficient for robust pattern learning, particularly for rare bug types. The vocabulary overlap problem reflects the inherent semantic relationships between bug categories. Performance issues may stem from memory leaks, which themselves may arise from concurrency problems. These overlapping causal chains create semantic ambiguity that challenges purely text-based classification approaches. The improvement observed with data augmentation (from 55% to 72% accuracy) suggests that expanding the training corpus represents a viable path toward improved generalization. However, the quality of augmented data

is critical; synthetic examples must preserve the semantic characteristics of authentic hotfixes to be useful for training.

c6RQ2 Answer

On 51 specially challenging hotfixes, the ensemble underperformed relative to training metrics.

Labels collapsed into a few dominant classes due to the small, imbalanced training dataset (88 records across 17 categories) and strong vocabulary overlap between categories.

However, data augmentation improved results substantially (72% accuracy, 0.75 NMI on augmented data), suggesting that expanded training data and class balancing techniques could significantly enhance practical utility.

6.6.3 Implications for LLM Non-Determinism

The presence of repeated solutions with different fitness values (e.g., 1.1, 1.2, 1.3, 1.4) reflects the inherent stochasticity of LLM-based summarization. Even with identical inputs, LLMs may produce slightly different summaries, leading to variations in embedding representations and subsequent clustering outcomes. This non-determinism presents both challenges and opportunities. On one hand, it complicates reproducibility and makes optimization landscapes noisy. On the other hand, the ensemble of solutions can provide robustness through majority voting, as employed in our generalization evaluation.

6.6.4 Limitations of the Optimization Process

A key limitation of this work concerns the scale of the NSGA-II optimization process. Due to the computational cost of each evaluation—which involves LLM-based summarization, embedding computation, and clustering—the total number of evaluated configurations remains limited. Under these constraints, convergence of NSGA-II to a true Pareto front cannot be reasonably expected. The non-dominated solutions identified should therefore be interpreted as promising configurations discovered through guided search rather than globally optimal trade-offs.

Alternative approaches better suited to expensive black-box optimization, such as surrogate-assisted evolutionary algorithms [81] or Bayesian optimization methods [139], could potentially achieve better convergence with fewer evaluations by building prob-

6. GREEN FEATURE SELECTION

abilistic models of the objective landscape. These represent promising directions for future work.

The presence of many duplicate or near-duplicate solutions in the non-dominated front (e.g., Individuals 1.1–1.4, 4.1–4.2, 5.1–5.2, 8.1–8.2 in Table 6.1) further reflects these limitations. These duplicates arise because identical feature subsets, when evaluated with a stochastic LLM-based pipeline, produce slightly different fitness values across evaluations due to the inherent non-determinism of LLM inference. With a limited evaluation budget, NSGA-II may re-evaluate similar or identical configurations without sufficient selective pressure to diversify the population, resulting in multiple non-dominated variants of essentially the same solution. This observation underscores both the challenge of optimizing in noisy, expensive fitness landscapes and the need for strategies that explicitly account for evaluation noise, such as re-evaluation averaging or noise-handling mechanisms within the evolutionary algorithm.

6

6.6.5 Practical Applications

The taxonomy derived through HotCat enables systematic analysis of urgent patches and opportunities for automated reasoning in software maintenance. Practitioners can leverage the category structure to:

- Prioritize resources: Categories such as “Crash or Hang” and “Security Vulnerability” may warrant different response protocols than “Test suite” issues.
- Identify patterns: Recurring categories within a project may indicate architectural weaknesses or areas requiring preventive investment.
- Automate triage: Classification models trained on the taxonomy could support automated routing of incoming bug reports to appropriate teams.

The integration with Jira issue tracking systems, as demonstrated through the meta-data enrichment process, suggests potential for seamless deployment within existing software development workflows.

6.7 Conclusions

This chapter presented HotCat, a novel pipeline that integrates semantic patch analysis with MOO to construct taxonomies for time-critical software bugs. The empirical results provide evidence that classification effectiveness need not be sacrificed for compu-

tational efficiency, aligning the approach with **green AI** principles. Specifically, the optimal balanced configuration achieved an accuracy of 59% and an **NMI** score of 0.58 with a runtime of 129 seconds, while the maximum accuracy configuration reached 63% in 132 seconds. These findings indicate that higher accuracy can be attained without substantial increases in resource consumption. Furthermore, the study demonstrated the critical impact of data augmentation, which significantly improved generalization performance from 55% to 72%, thereby helping to mitigate the inherent challenges of vocabulary overlap and class boundary blurring observed in text-based classification.

Despite these advancements, the study acknowledges limitations regarding the modest scale of the dataset, the use of runtime as a proxy for energy efficiency, and the constrained NSGA-II budget which may not guarantee convergence to the true Pareto front. Surrogate-assisted optimization methods or Bayesian approaches could potentially improve search efficiency under such evaluation constraints and represent a promising direction for future work. Consequently, future research directions prioritize the expansion of the corpus across diverse open-source ecosystems and the incorporation of direct carbon emission metrics as primary objectives within the NSGA-II optimization loop. Additionally, further investigation is warranted into hierarchical taxonomies to better capture causal relationships between bug types and strategies to reduce **LLM** variance. Ultimately, this work contributes a replicable framework to the field of sustainable **SE**, offering practitioners a scalable method for automating hotfix analysis within issue tracking systems like Jira while adhering to resource-constrained operational environments.

7

Greener Agent for Greener Code via Multi-Objective Optimization

This chapter introduces GA4GC (Greener Agent for Greener Code), a framework that employs multi-objective optimization to discover non-dominated configurations for coding agents. By simultaneously optimizing for reduced computational overhead and improved task performance, GA4GC achieves 135-fold hypervolume improvement and 37.7% runtime reduction while improving correctness. This investigation directly addresses the thesis research question RQ4 (sustainable LLM deployment) within the context of agentic coding workflows, establishing configuration optimization as a viable sustainability lever.

Authors: Jingzhi Gong, Yixin Bian, Luis de la Cal, Giovanni Pinna, Anisha Uteem, David Williams, Mar Zamorano, Karine Even-Mendoza, William B. Langdon, Hector D. Menendez, Federica Sarro,

Original title: GA4GC: Greener Agent for Greener Code via Multi-Objective

Published in: The 17th Symposium on Search-Based Software Engineering 2025 (SSBSE 2025), SSBSE 2025 Challenge Case: Green SBSE.

<https://kclpure.kcl.ac.uk/portal/en/publications/ga4gc-greener-agent-for-greener-code-via-multi-objective/>

Candidate's contribution: The candidate contributed to the conception of the project, the design of the experiments, the analysis and interpretation of the results, and the writing of parts of the manuscript, as well as revising the manuscript.

7.1 Introduction

Code performance optimization represents a fundamental pillar of modern software development, directly influencing system scalability, resource consumption, and end-user experience [140, 154]. The process of improving software efficiency has traditionally required substantial human expertise and manual effort, making it a prime candidate for automation through emerging artificial intelligence technologies. In recent years, LLM have demonstrated considerable promise in automating this traditionally labor-intensive process [53, 146]. However, current LLM-based approaches predominantly focus on simplified programming benchmarks such as HumanEval [23], which involve self-contained algorithmic problems that fail to capture the complexity inherent in real-world software engineering challenges [79].

To address this fundamental limitation, both academic researchers and industry practitioners have increasingly turned to *agentic workflows*—sophisticated multi-step processes wherein LLMs operate as autonomous agents capable of iterative reasoning, tool utilization, and complex decision-making [7, 105]. As discussed in Section 2.3.4, these agents integrate LLM reasoning cores with memory systems and external tools, enabling them to handle repository-level tasks that single-shot generation cannot address. These agentic approaches have demonstrated particular effectiveness when evaluated on realistic SE benchmarks such as SWE-bench [80] and the more recent SWE-Perf [61], which provide authentic repository contexts and genuine SE tasks reflecting the complexity that agents encounter in production environments.

Nevertheless, unlike single-shot code LLMs, coding agents operate through iterative reasoning processes that necessitate multiple LLM calls, each consuming significant computational resources [12, 158]. While these agents can successfully solve complex real-world coding tasks, a single agent run on real-world SE problems can consume over 100,000 tokens [6], translating to substantial monetary costs per optimization task. Moreover, without careful hyperparameter tuning, the energy consumed by an optimization agent can be so substantial that the resulting code may need to be executed hundreds of thousands of times to reach an energetic “break-even point,” rendering some optimizations a net energy loss [27]. As organizations scale coding agent deployments across development teams and projects, this computational intensity creates prohibitive operational costs while threatening to exponentially increase the environmental footprint [74], directly conflicting with the GSE principles and green AI objectives discussed in Section 2.2.3.

This chapter addresses these critical challenges, directly contributing to the thesis research question RQ4, which asks how we can develop and deploy LLM-based systems while minimizing their environmental footprint and ensuring sustainable resource utilization. Building upon the MOO foundations established in Section 2.2.3 and extending the sustainability-focused approach of Chapter 6, we introduce *Greener Agent for Greener Code* (GA4GC), a novel framework that optimizes the trade-off between resource consumption of the coding agent and performance of the generated code. The key insight underlying GA4GC is that the vast configuration space of coding agents—encompassing prompt templates, LLM-specific hyperparameters, and agent-specific operational constraints—is too complex for manual exploration. Therefore, GA4GC employs NSGA-II [36], the well-established Multi-Objective Genetic Algorithm introduced in Section 2.2.3, to discover non-dominated agent configurations automatically.

7.1.1 Research Questions

This chapter contributes to the thesis research question RQ4 (sustainable LLM deployment) by addressing three chapter-specific research questions:

c7RQ1: *To what extent can GA4GC improve the resource consumption and performance trade-offs of coding agents compared to default configurations?*

This research question examines whether MOO can discover agent configurations that simultaneously reduce computational overhead while maintaining or improving task effectiveness. Answering this question is crucial for practitioners seeking to deploy coding agents at scale without incurring prohibitive energy and monetary costs, directly contributing to RQ4.

c7RQ2: *How do different hyperparameters influence agent resource consumption and task performance in the optimization process?*

Understanding the relative importance and directional effects of individual hyperparameters enables practitioners to make informed configuration decisions even without running full optimization procedures. This knowledge facilitates the development of practical guidelines for sustainable agent deployment.

c7RQ3: *What actionable strategies can be derived from the non-dominated configurations for sustainable coding agent deployment?*

Beyond theoretical insights, this research question seeks to translate optimization results into concrete recommendations that practitioners can immediately

apply across different deployment scenarios, balancing runtime efficiency, code quality, and performance requirements.

These chapter-specific research questions directly address the thesis-level question RQ4. By showing that configuration optimization can significantly reduce computational overhead (c7RQ1), identifying which hyperparameters most influence resource consumption (c7RQ2), and providing actionable strategies for deployment (c7RQ3), this work contributes practical approaches for sustainable LLM deployment.

7.1.2 Contributions

The main contributions of this chapter are threefold:

1. **GA4GC Framework:** A MOO framework that discovers non-dominated coding agent configurations, achieving up to 37.7% runtime reduction (943 seconds versus 1513 seconds) while improving correctness, with up to 135-fold Hypervolume (HV) improvement over default configurations.
2. **Hyperparameter Influence Analysis:** Comprehensive analysis via Random Forest regression revealing that temperature exhibits the highest overall influence on agent behavior. LLM hyperparameters primarily impact task effectiveness, while agent constraints predominantly affect resource consumption.
3. **Actionable Deployment Strategies:** Practical suggestions for coding agent practitioners across three deployment scenarios: runtime-critical environments requiring low temperature with restrictive `top_p`, performance-critical scenarios benefiting from moderate temperature with balanced `top_p`, and context-specific optimization via GA4GC for tailored requirements.

The remainder of this chapter is organized as follows. Section 7.2 reviews related work on coding agents and their optimization. Section 7.3 presents the GA4GC methodology in detail. Section 7.4 describes the experimental setup, including the benchmark, agent architecture, and evaluation metrics. Section 7.5 presents the experimental results. Section 7.6 discusses the findings and answers the research questions. Finally, Section 7.7 concludes the chapter and outlines future research directions.

7.2 Related Work

This section contextualizes the GA4GC framework within the intersection of LLM-based coding agents and sustainable AI deployment. Building upon the MOO and green AI foun-

dations established in Section 2.2.3, we focus here on the specific challenges of agentic code optimization workflows.

7.2.1 Environmental Impact of AI Agents

As discussed in Section 2.2.3, the environmental impact of AI has emerged as a pressing concern, with the distinction between “Red AI” (accuracy at any cost) and “Green AI” (efficiency as a primary metric) formalized by Schwartz et al. [137]. Recent literature indicates a pivot from training to inference as the dominant source of energy consumption, with Wilkins et al. [157] demonstrating that a single year of inference can consume over $25\times$ the energy required for training. This is particularly relevant for coding agents that rely on iterative inference, where a single agent run on real-world SE problems can consume over 100,000 tokens [6].

While LLMs can theoretically be more carbon-efficient than human labor for specific tasks [131], this advantage is highly sensitive to model configuration and geographical deployment choices, which can vary emissions by a factor of 30 due to grid carbon intensity [65]. These findings motivate the configuration optimization approach developed in this chapter.

7.2.2 LLM-based Coding Agents

The shift from simple code generation—evaluated on benchmarks such as HumanEval discussed in Section 2.4.1—to autonomous problem-solving is benchmarked by SWE-bench [80], which tests agents on real-world GitHub issues. Research in this area reveals a tension between architectural complexity and model capability. Early complex architectures like SWE-agent [161] showed significant gains over baselines. However, recent lightweight implementations such as mini-SWE-agent [49] and Agentless [160] demonstrate that as base models improve, simpler scaffolding can achieve competitive resolution rates (up to 74% on SWE-bench Verified) at lower costs. Despite these advances, operational costs remain high (exceeding \$1.15 per issue for state-of-the-art models), and significant capability gaps persist in performance optimization tasks, where agents trail human experts substantially [61].

7.2.3 Multi-Objective Optimization for AI Systems

The NSGA-II algorithm, introduced in Section 2.2.3, provides the optimization framework for GA4GC. Recent studies have successfully applied MOO to Generative AI systems.

Projects like GreenStableYolo [50] and SustainDiffusion [32] utilized NSGA-II to optimize image generation models, achieving substantial reductions in energy and bias with minimal quality loss. Similarly, reinforcement learning (RL) approaches in code generation have realized energy savings of up to 50% via dynamic early exits [73].

Although MOO has been applied to hyperparameter tuning and image generation, no prior work has systematically optimized the configuration of LLM-based coding agents. The unique, iterative nature of agentic workflows creates a distinct configuration space that GA4GC addresses by discovering non-dominated solutions that balance task effectiveness with resource consumption.

7.3 Methodology

This section delineates the GA4GC framework, which formalizes the configuration of LLM coding agents as a MOO problem. The primary objective is to identify non-dominated configurations that balance competing trade-offs in agentic workflows. Figure 7.1 illustrates the overall workflow, where the NSGA-II optimizer explores the agent configuration space defined by a tuple $\mathcal{C} = (\theta_{\text{LLM}}, \theta_{\text{Agent}}, \tau)$, comprising LLM-specific hyperparameters, agent operational constraints, and the prompt template variant. The optimization process seeks to simultaneously minimize three distinct objective functions: (i) the rate of incorrect patch generation (effectively maximizing correctness), (ii) the negative performance gain (thereby maximizing the computational speedup of the generated code), (iii) and the total agent execution runtime.

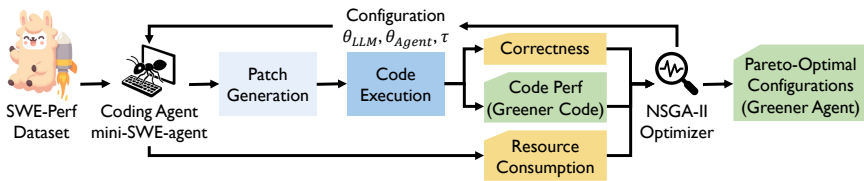


Figure 7.1: Workflow of the GA4GC multi-objective configuration optimization. Given a task from the SWE-Perf dataset, the coding agent (mini-SWE-agent) generates patches under a candidate configuration $\mathcal{C} = (\theta_{\text{LLM}}, \theta_{\text{Agent}}, \tau)$. Code execution yields three fitness objectives—correctness, code performance (greener code), and resource consumption—which are fed into the NSGA-II optimizer to evolve Pareto-optimal configurations (greener agent).

The optimization explores a heterogeneous search space that integrates mixed-variable types. This space includes continuous LLM sampling parameters such as

temperature and `top_p`, integer-based operational constraints like `maximum token counts` and `step limits`, and categorical variables representing different prompt instructional strategies. To assess the fitness of these configurations, the framework utilizes a rigorous evaluation pipeline where agents address tasks from the SWE-Perf benchmark. The resulting patches are validated within isolated Docker environments to ensure reproducibility, providing precise measurements of functional correctness against test suites and relative performance improvements compared to baseline implementations. To traverse this complex configuration landscape, the framework employs the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [36]. This algorithm is selected for its computational efficiency of $O(MN^2)$ and its robust elitism mechanisms, which preserve high-quality solutions across generations. Diversity is maintained through crowding distance assignment rather than explicit sharing parameters. The evolutionary process utilizes binary tournament selection and applies specialized crossover and mutation operators adapted for the specific nature of the variables—using simulated binary crossover for continuous parameters and distinct operators for integer and categorical constraints—to iteratively refine the population toward the optimal Pareto front.

7.4 Experimental Setup

The experimental evaluation leverages the SWE-Perf benchmark [61], a dataset designed to rigorously assess code performance optimization capabilities. To mitigate the substantial computational expenditures associated with MOO involving iterative LLM calls, the study focuses specifically on the astropy project, a representative scientific computing library. The dataset is partitioned into a training set of nine instances for the optimization process and a held-out validation set of three instances to assess out-of-sample generalization. The agentic framework is implemented using mini-SWE-agent [49] driven by the Gemini 2.5 Pro model [28]. This architecture facilitates a transparent, multi-stage workflow encompassing task reception, context retrieval, iterative reasoning, and patch generation, while allowing for precise programmatic control over the hyperparameters under investigation.

The optimization process employs the NSGA-II algorithm via the *pymoo* library [13]. Due to the high latency of individual evaluations—which range from 15 to 60 minutes per configuration—the parameters are conservatively bounded to a population size of five and a total of five evolutionary generations, resulting in twenty-five unique configuration

evaluations. These experimental constraints reflect the significant resource demands of the study, where each complete optimization run necessitates approximately 30 hours of wall-clock time and substantial API costs. The optimized configurations are evaluated against a permissive baseline characterized by high token limits and extensive iteration capabilities to establish a rigorous performance standard.

To ensure reproducibility and isolate performance metrics from environmental noise, all experiments are conducted in a controlled Google Cloud Platform environment. Performance gains are validated through twenty independent executions per instance, with statistical significance determined via the Mann-Whitney U test suitable for small sample sizes. The quality of the optimization is quantified using the HV indicator [174], which captures both the convergence and diversity of the Pareto front. Specifically, the HV is computed using *pymoo* with all three objectives normalized to $[0, 1]$ and the runtime objective inverted (since it is to be minimized). The HV indicator is computed using a reference point defined as the worst observed value for each objective across all evaluated configurations, shifted by a small margin (10%) to ensure that all non-dominated solutions contribute positively to the indicator. Specifically, the reference point is set to (r_1, r_2, r_3) where $r_i = \text{worst}_i \times 1.1$ for minimization objectives. The HV values reported in Table 7.1 are computed for each individual configuration relative to this reference point, measuring the volume of objective space dominated by that single configuration. Cumulative HV values (used in the comparison with random search) measure the total volume dominated by the entire set of non-dominated solutions.

Furthermore, the study incorporates a post-hoc analysis using Random Forest regressors and Spearman correlation coefficients to empirically determine the relative importance and directional influence of specific hyperparameters on the optimization objectives [50, 116].

7.5 Results

This section presents the experimental results organized around the three research questions. We first examine optimization effectiveness (c7RQ1), then analyze hyperparameter influences (c7RQ2), and finally derive actionable strategies (c7RQ3).

Table 7.1 presents the comparison between default and GA4GC-optimized configurations. The NSGA-II optimization process evaluates a total of 25 candidate configurations (population of 5 across 5 generations), each identified by a sequential number (#1 through #25) corresponding to the order in which it was evaluated. Of these, five con-

7. GA4GC

figurations are identified as non-dominated—*i.e.*, they are not dominated by any other evaluated configuration across all three objectives. The non-consecutive numbering (#4, #5, #9, #15, #16) reflects the positions of these non-dominated solutions within the full set of 25 evaluated configurations.

Table 7.1: Comparison of default versus GA4GC-optimized configurations. Corr = Correctness (correct patches out of 9), Perf = Performance gain (%), Runt = Runtime (seconds), HV = Hypervolume (%), VHV = Validation hypervolume (%). Green-shaded cells indicate improvements over default.

Config	Temp	TopP	Token	Step	Cost	ETi	LTi	Pr	Corr	Perf	Runt	HV	VHV
Default	0.000	1.000	4096	240	3.00	60	60	-	2/9	0.00%	1513.3	0.52%	1.1%
#4	0.085	0.135	1120	36	9.26	41	57	2	4/9	0.00%	943.1	5.82%	4.1%
#5	0.692	0.384	2972	38	6.73	40	56	3	8/9	6.43%	984.8	70.28%	14.9%
#9	0.725	0.412	2972	22	6.73	43	41	3	7/9	0.00%	958.1	9.25%	21.6%
#15	0.657	0.384	2972	38	6.73	40	56	2	7/9	10.67%	1400.1	33.42%	2.7%
#16	0.085	0.131	1120	36	6.91	41	57	2	0/9	0.00%	853.3	1.10%	21.6%

The configuration identifiers (#4, #5, #9, #15, #16) correspond to the sequential evaluation order across all generations. The five non-dominated configurations span different stages of the evolutionary process: some were discovered early (e.g., #4, #5), while others emerged in later generations (e.g., #15, #16). The presence of high-performing configurations in early evaluations reflects the semi-random initialization of the first generation. However, solutions discovered in later generations (e.g., #15, achieving the highest performance gain of 10.67%) demonstrate that the evolutionary process contributes additional optimization beyond initial sampling. The limited total budget of 25 evaluations constrains the extent to which later generations can refine the population, a limitation discussed further in Section 7.6.

Configuration #4 achieves a 37.7% runtime reduction (943.1 seconds versus 1513.3 seconds) while simultaneously doubling correctness (4/9 versus 2/9). Configuration #16 achieves the maximum runtime reduction of 43.6% (853.3 seconds), though at the cost of zero correctness. This extreme trade-off illustrates the Pareto front boundary where runtime is minimized but task effectiveness is sacrificed. Configuration #5 delivers four-fold better correctness (8/9 versus 2/9) while simultaneously achieving 6.43% performance gain—a rare configuration that improves multiple objectives concurrently. This represents a substantial advancement over the default, which achieved neither performance gains nor competitive correctness. Configuration #15 achieves the highest performance gain of 10.67%, approaching expert-level improvements on SWE-Perf, though with increased runtime overhead (1400.1 seconds). This configuration represents

the quality-optimized boundary of the Pareto front. The HV indicator, which quantifies the proportion of the normalized objective space dominated by each configuration (see Section 7.4 for computation details), reveals substantial improvements across all optimized configurations. Configuration #5 achieves an HV of 70.28%, compared to only 0.52% for the default—a 135-fold improvement—demonstrating comprehensive progress across all three objectives simultaneously. This means that Configuration #5 dominates approximately 70% of the feasible trade-off space between correctness, performance gain, and runtime, whereas the default configuration covers a negligible portion. To clarify, the default configuration achieves an HV of 0.52%, meaning it dominates only 0.52% of the objective space bounded by the reference point. Configuration #5 achieves 70.28%, indicating that this single configuration dominates 70.28% of the bounded objective space—a 135-fold improvement ($70.28/0.52 \approx 135$). This dramatic increase reflects the fact that Configuration #5 simultaneously improves across all three objectives (correctness, performance, and runtime) compared to the default. Configuration #15 achieves 64-fold improvement (33.42%), while even the lowest-performing non-dominated configuration (#16) achieves 2-fold improvement (1.10%). Four out of five non-dominated configurations dominate the default configuration in multiple objectives, addressing both “greener agent” (reduced runtime) and “greener code” (improved performance) requirements simultaneously. Only Configuration #16 represents a specialized extreme that sacrifices correctness entirely for maximum runtime efficiency.

To assess generalization capability, we evaluate all non-dominated configurations on the three held-out validation instances. Table 7.2 presents validation hypervolume results, demonstrating that optimized configurations maintain superior performance on unseen tasks.

Table 7.2: Validation hypervolume (VHV) on held-out instances.

Configuration	Default	#4	#5	#9	#15
VHV (%)	1.1	4.1	14.9	21.6	2.7

All non-dominated configurations maintain higher validation hypervolume than the default (1.1%), suggesting that the discovered configurations generalize beyond the training instances rather than overfitting to specific tasks. To verify that NSGA-II provides directed optimization rather than random exploration, we compare against a random search baseline that samples 25 configurations uniformly from the search space.

NSGA-II achieves 83.0% cumulative HV with 5 non-dominated solutions, compared to 53.1% cumulative HV and 3 non-dominated solutions from random search. This 57% improvement in hypervolume demonstrates the effectiveness of guided evolutionary search, even under the constrained evaluation budget of 25 configurations.

Table 7.3, which presents the hyperparameter influence analysis, quantifying how each configuration parameter affects the three optimization objectives. Feature importances are extracted from Random Forest regressors trained on all 25 evaluated configurations, using each configuration's hyperparameter values as features and each objective value as the target variable (one Random Forest per objective). The importance values represent the mean decrease in impurity (Gini importance) when splitting on each hyperparameter, averaged across all trees in the ensemble. Higher values indicate that the hyperparameter has a stronger influence on the corresponding objective. This approach is commonly used for post-hoc sensitivity analysis in hyperparameter optimization studies [116].

Table 7.3: Random Forest feature importance for hyperparameters on optimization objectives. Colors indicate importance magnitude: Low (0.0–0.1), Medium (0.1–0.2), High (>0.2).

Category	Hyperparameter	Correctness	Performance	Runtime
LLM	Temperature	0.152	0.392	0.199
	Top_p	0.199	0.051	0.097
	Max_tokens	0.057	0.090	0.089
Agent	Step_limit	0.140	0.119	0.049
	Cost_limit	0.199	0.076	0.128
	Env_timeout	0.060	0.034	0.298
	LLM_timeout	0.120	0.109	0.102
Prompt	Template Variant	0.072	0.130	0.038

Temperature emerges as the most critical hyperparameter with the highest importance for performance impact (0.392). High-performing configurations #5 and #15 use moderate temperatures (0.66–0.69), while low-temperature configurations #4 and #16 achieve faster runtime but no performance gains. This pattern indicates that temperature balances exploration (finding creative optimization strategies) against exploitation (generating reliable, deterministic patches). Moderate temperatures enable sufficient exploration to discover effective optimizations while maintaining enough focus to produce valid solutions. The top_p parameter shows substantial influence on correctness (0.199) with successful configurations using mid-range values (0.38–0.41). This sug-

gests that balanced vocabulary sampling avoids both overly restrictive token selection (limiting solution diversity) and chaotic sampling (introducing noise into generated patches). The `Cost_limit` exhibits influence across correctness (0.199) and runtime (0.128). non-dominated configurations employ higher budgets (\$6.73–\$9.26 versus \$3.0 default), enabling more thorough exploration without premature termination due to cost constraints. The increased budget allows agents to iterate more extensively on challenging tasks. Environment timeout shows the highest runtime impact (0.298), indicating that timeout settings directly control execution duration. Lower timeout values (40–43 seconds in non-dominated configurations versus 60 seconds default) prevent inefficient exploration patterns where the agent waits excessively for slow operations. Finally, prompt template variants show moderate performance influence (0.130), with templates 2 and 3 dominating the Pareto front. This suggests that task-specific prompt engineering significantly impacts optimization effectiveness, though the effect is smaller than LLM hyperparameters.

Based on the hyperparameter influence analysis and non-dominated configuration patterns, we derive actionable strategies for practitioners across three deployment scenarios.

For deployments where minimizing computational cost and latency is paramount:

- Set `temperature` to minimum values (0.0–0.1) to reduce exploration overhead and accelerate convergence.
- Use restrictive `top_p` values (0.13–0.14) to limit vocabulary sampling and decrease generation time.
- Configure moderate `max_tokens` (1120–2000) to bound response length while maintaining sufficient capacity.
- Set conservative `step_limit` (20–36) to prevent excessive iteration.
- Apply tight timeout constraints (40–45 seconds) to terminate slow operations promptly.

The expected outcome for this scenario is up to 43.6% runtime reduction with potential correctness trade-offs.

For deployments prioritizing code optimization quality over resource efficiency:

- Use moderate `temperature` (0.65–0.70) to enable creative optimization strategy exploration.
- Set balanced `top_p` values (0.38–0.41) to maintain solution diversity without introducing excessive noise.

7. GA4GC

- Allocate generous `cost_limit` (\$6.5–\$9.5) to permit thorough exploration.
- Configure higher `step_limit` (35–40) to allow iterative refinement.
- Select performance-oriented prompt templates (variants 2 or 3) optimized for optimization tasks.

The expected outcome for this scenario is up to 10.67% performance gain with 4-fold correctness improvement.

For practitioners with specific requirements that do not align with the above scenarios:

- Apply GA4GC directly to discover non-dominated configurations tailored to particular deployment priorities.
- Define custom fitness functions reflecting organization-specific metrics (e.g., energy consumption, API cost, solution quality).
- Use domain-specific prompt templates aligned with target codebase characteristics.

The expected outcome for this scenario is customized configurations optimized for specific trade-off preferences.

7.6 Discussion

The experimental analysis reveals key insights for sustainable coding agent deployment, addressing each research question through the lens of configuration optimization.

7.6.1 Optimization Effectiveness

The results demonstrate that GA4GC identifies non-dominated configurations that substantially outperform default settings across all three objectives. The 135-fold improvement in HV achieved by Configuration #5 represents a notable advancement within the explored search space, simultaneously improving correctness, runtime efficiency, and code performance. However, given the limited evaluation budget (25 configurations across 5 generations with a population of 5), convergence to the true Pareto front cannot be guaranteed; larger budgets or surrogate-assisted methods [81] may reveal additional trade-offs. Despite this constraint, the results have meaningful implications for the “break-even problem” identified by Coignion et al. [27], where the energy cost of optimization often outweighs the savings generated by the optimized

code. GA4GC mitigates this by reducing agent runtime by up to 37.7% while enhancing correctness, effectively lowering the threshold for net-positive environmental impact.

The validation results provide evidence that discovered configurations generalize beyond training instances. All non-dominated configurations maintained higher validation HV than the default, suggesting that the optimization process identifies robust configurations rather than overfitting to specific tasks, although broader validation across diverse projects would strengthen this conclusion.

c7RQ1 Answer

GA4GC achieves 135-fold higher HV, 37.7% faster runtime while improving correctness, and four out of five non-dominated configurations dominating the default while all maintaining superior HV on unseen tasks.

7.6.2 Hyperparameter Influence Patterns

The experimental analysis reveals key differences from standard practices in coding agent configuration. Contrary to the common preference for deterministic generation, the data indicate that moderate temperatures (0.65–0.73) offer a superior balance between exploration and exploitation, facilitating the discovery of non-trivial optimizations that deterministic settings often miss.

The relationship between cost limits and correctness reveals a counterintuitive aspect of sustainability: aggressive budget constraints may ultimately increase environmental costs by leading to incorrect solutions that then require re-execution. Consequently, non-dominated configurations consistently allocate higher monetary budgets to ensure first-pass correctness.

The results also highlight that agent runtime is frequently dominated by environmental interactions rather than inference; thus, implementing tighter timeouts to enforce “fail-fast” strategies proves more effective than prolonged execution windows. Additionally, the persistence of specific prompt templates on the Pareto front underscores that prompt engineering remains a vital complement to hyperparameter tuning.

The separation between LLM hyperparameters (primarily affecting task effectiveness) and agent constraints (primarily affecting resource consumption) confirms that single-objective tuning is insufficient for sustainable deployment.

c7RQ2 Answer

Temperature shows the highest overall influence across objectives. LLM hyper-parameters primarily impact task effectiveness (correctness and performance), while agent constraints predominantly affect resource consumption (runtime). This separation confirms the need for MOO rather than single-objective tuning.

7.6.3 Actionable Deployment Strategies

The three deployment scenarios derived from non-dominated patterns provide practitioners with concrete guidance. The runtime-critical strategy (Configuration #4, #16) suits environments with strict latency or cost constraints, achieving up to 43.6% runtime reduction. The performance-critical strategy (Configuration #5, #15) targets scenarios where code quality is paramount, delivering up to 10.67% performance gains with four-fold correctness improvement. The context-specific approach enables organizations to apply GA4GC directly, discovering configurations aligned with their unique trade-off preferences.

These strategies establish configuration optimization as an underutilized sustainability lever, demonstrating that substantial efficiency gains are attainable without the prohibitively expensive requirements of model retraining or hardware upgrades.

c7RQ3 Answer

We provide scenario-specific actionable suggestions for green coding agent deployment: runtime-critical (minimize temperature, restrictive top_p), performance-critical (moderate temperature, balanced top_p, generous budgets), and context-specific optimization via GA4GC for tailored requirements.

7.6.4 Limitations of the Optimization Process

The optimization budget of 25 total evaluations (population size 5×5 generations) represents a fundamental constraint of this study, imposed by the high computational cost of each evaluation (15–60 minutes per configuration). With such a limited budget, NSGA-II cannot be expected to converge to the true Pareto front of the configuration space. The non-dominated solutions identified should therefore be interpreted as promising configurations discovered through directed search, rather than provably optimal trade-offs.

This limitation is reflected in the numbering of results in Table 7.1: configurations are numbered sequentially by evaluation order (#4, #5, #9, #15, #16), indicating that some of the best-performing solutions were discovered in different generations of the evolutionary process. Notably, configurations from early generations (e.g., #4, #5) already exhibit strong performance, suggesting that even the initial (semi-random) population contained competitive solutions. This observation is consistent with the limited number of generations available for evolutionary refinement.

Alternative optimization approaches better suited to expensive evaluations, such as surrogate-assisted evolutionary algorithms [81] or Bayesian optimization methods, could potentially discover higher-quality trade-offs with the same evaluation budget by constructing surrogate models of the objective landscape. Future work should investigate these alternatives, particularly as evaluation costs for LLM-based agents continue to grow.

Despite these constraints, the comparison against random search (Section 7.5) demonstrates that NSGA-II provides directed optimization: achieving 83.0% cumulative HV versus 53.1% from random search. This suggests that even with limited evaluations, guided evolutionary search offers meaningful advantages over uninformed exploration.

7.6.5 Positioning and Limitations

Ultimately, this work frames sustainability as an inherently multi-objective challenge, where the trade-offs between a “greener agent” (resource efficiency), “greener code” (performance gain), and solution quality must be explicitly managed rather than optimized in isolation.

GA4GC distinguishes itself from RL frameworks like Green-Code [73] by operating at the configuration level rather than the architectural level, rendering it immediately deployable without model modification. It further advances beyond single-shot optimization techniques like GreenStableYolo [50] by systematically addressing the compounding complexities of multi-turn, iterative agentic workflows, offering a robust, mathematical alternative to ad-hoc manual tuning.

The interpretation of these results is subject to specific validity constraints. The evaluation focuses on the astropy project (12 instances from SWE-Perf) with a single agent architecture (mini-SWE-agent with Gemini 2.5 Pro), which may limit generalizability across different codebases, programming languages, and agent designs. Most critically, the restricted evaluation budget—25 configurations total, dictated by the high computational cost of each evaluation (15–60 minutes per configuration)—means

that NSGA-II cannot be expected to converge to the true Pareto front. The discovered non-dominated solutions should therefore be interpreted as promising configurations that improve substantially over default settings, rather than as globally optimal trade-offs. Alternative optimization strategies better suited to expensive black-box evaluations, such as surrogate-assisted evolutionary algorithms [81] or Bayesian optimization approaches [139], could potentially achieve superior results within the same evaluation budget by constructing approximation models of the objective landscape. These alternatives represent a natural avenue for future work. While runtime serves as a practical proxy for resource consumption, it does not perfectly capture energy expenditure, suggesting future research should incorporate direct measurement tools like CodeCarbon [29]. The stochastic nature of both NSGA-II and LLM inference means that results may vary across runs; independent replications would further strengthen confidence in the reported findings.

7.7 Conclusions

This chapter presented GA4GC, a novel framework that utilizes the NSGA-II algorithm to navigate the complex multi-objective configuration space of LLM-based coding agents. The empirical evaluation demonstrates the framework's potential for resolving the inherent tension between computational resource consumption and task performance, with non-dominated configurations achieving up to a 135-fold improvement in HV and a 37.7% reduction in runtime compared to the default baseline under a constrained evaluation budget of 25 configurations. Crucially, the analysis delineates the distinct impact of hyperparameter categories: while moderate temperatures (0.65–0.73) are essential for governing task effectiveness and exploration, agent operational constraints are the primary determinants of resource efficiency. These findings validate the necessity of MOO over single-objective approaches, establishing configuration tuning as a vital sustainability lever that complements traditional architectural and hardware-focused efficiency measures.

From a broader scientific perspective, GA4GC offers a methodological template for the responsible, scalable deployment of AI in SE, empowering practitioners to make informed trade-offs suited to specific deployment contexts.

Future research will expand upon this foundation by extending evaluations to diverse benchmarks such as EffiBench [71], exploring surrogate-assisted evolutionary optimization [81] or Bayesian approaches [139] to improve search efficiency under expensive

evaluations, and incorporating direct energy measurement via tools like CodeCarbon to move beyond proxy metrics.

Furthermore, potential avenues for advancement include the exploration of many-objective formulations using NSGA-III [35] to optimize for additional criteria such as monetary cost and solution diversity, as well as the development of dynamic adaptation strategies that adjust agent configurations in real-time based on task characteristics.

IV

Conclusions

8

General discussion

This thesis has investigated the challenges that accompany the deployment of LLMs in real-world applications, organized around four complementary research directions: developing meaningful evaluation methodologies, assessing the potential for knowledge democratization, enhancing output reliability through hybrid approaches, and ensuring sustainable deployment through multi-objective optimization.

A central motivation underlying this work is the growing inadequacy of established evaluation practices. Traditional NLP metrics such as BLEU, ROUGE, and EM were designed for narrowly scoped tasks with well-defined ground truth, yet modern LLMs operate across a broad spectrum of applications where outputs can be semantically correct but syntactically diverse, or where subtle errors remain invisible to surface-level comparison. As we demonstrate throughout this thesis, this evaluation gap has concrete consequences: in the legal domain, it means that fluent but inaccurate summaries go undetected; in code generation, it means that functionally flawed programs pass inadequate test suites; and in deployment optimization, it means that efficiency gains remain unexplored when only single objectives are considered.

To address these interconnected challenges, we conducted empirical investigations spanning five application domains—legal text summarization, code generation, text-to-SQL translation, software bug classification, and coding agent optimization—using a combination of human evaluation studies, extensive benchmark experiments, and evolutionary multi-objective optimization. The findings, summarized in Table 8.1 and discussed in detail in the following sections, converge on a unifying insight: responsible LLM deployment requires a multidimensional perspective that jointly considers what

these systems accomplish, how reliably they accomplish it, and at what computational and environmental cost.

8.1 Addressing the Research Questions

In the following subsections, we provide detailed answers to each of the four research questions that guided this thesis, drawing upon the findings from our empirical investigations across multiple application domains.

8.1.1 RQ1: Evaluation Methodologies for LLM Outputs

How can we develop more accurate and meaningful evaluation methodologies for LLM outputs that capture semantic correctness and practical utility beyond surface-level metrics? (Thesis RQ1)

Our research demonstrated that meaningful evaluation of LLM outputs requires domain-specific approaches that measure genuine utility rather than surface-level similarity. We addressed this question through two complementary investigations: human-centered evaluation of legal summarization and the development of continuous metrics for text-to-SQL evaluation.

In the legal domain, we conducted a comprehensive human-centered evaluation involving 75 participants with varying levels of legal knowledge. Our methodology employed comprehension questions designed by domain experts to measure actual understanding rather than perceived quality. The results revealed that traditional automatic metrics fail to capture the nuanced quality of LLM-generated summaries. GPT-4o summaries achieved comprehension improvements of 6 percentage points over original judgments for participants without legal knowledge, yet this modest improvement came at the cost of elevated error rates, particularly among legally knowledgeable participants. This finding underscores that surface-level fluency can mask underlying inaccuracies that become apparent only with deeper engagement. The inverse relationship between legal knowledge and GPT-4o error rates represents a particularly significant finding. Legally knowledgeable participants experienced higher error rates with GPT-4o summaries than participants without legal expertise, suggesting that the model generates plausible-sounding but potentially misleading content. This phenomenon indicates that evaluation methodologies must assess not only whether readers understand content but whether they understand it correctly. Our human-centered approach revealed

Table 8.1: Summary of thesis research questions, contributing chapters, and key findings. Each thesis-level RQ is addressed through specific chapter contributions with quantitative results where applicable.

Thesis RQ	Chapters	Key Contributions & Findings
RQ1: Evaluation Methodologies	Chapter 3 (Legal)	<ul style="list-style-type: none"> • Human-centered evaluation with 75 participants reveals limitations invisible to automatic metrics • QAS metric provides continuous assessment combining semantic (S_C) and table (S_T) similarity
	Chapter 5 (SQL)	<ul style="list-style-type: none"> • Granular metrics enable differentiation between models with similar mean performance
RQ2: Democratization of Specialized Knowledge	Chapter 3 (Legal)	<ul style="list-style-type: none"> • GPT-4o improves comprehension by 6 percentage points but increases error rates • Human-crafted Massime achieve 45-58% accuracy vs. 38% for GPT-4o
	Chapter 4 (Code)	<ul style="list-style-type: none"> • Even GPT-4 solves only 52% of PSB2 problems; smaller models <20% • Inverse expertise-error relationship: legal experts more misled by fluent but inaccurate summaries
RQ3: Hybrid Approaches for Reliability	Chapter 4 (Code)	<ul style="list-style-type: none"> • GI significantly improves LLM-generated code across nearly all tested configurations • Dynamic grammar specialization enables domain-specific optimization • Smaller models benefit more from enhancement (correct structure, flawed details) • Self-correction baseline largely ineffective, validating hybrid approach
RQ4: Sustainable LLM Deployment	Chapter 6 (HotCat)	<ul style="list-style-type: none"> • HotCat: Best-balanced non-dominated configuration achieves 59% accuracy, 0.58 <i>NMI</i> in 129s; max accuracy 63% in 132s
	Chapter 7 (GA4GC)	<ul style="list-style-type: none"> • GA4GC: Up to 135-fold <i>HV</i> improvement and 37.7% runtime reduction over default under constrained evaluation budget • Temperature most critical hyperparameter for agent task effectiveness • Efficiency and effectiveness jointly optimizable via MOO, though limited budgets preclude convergence guarantees

8. GENERAL DISCUSSION

limitations invisible to automatic metrics, highlighting the continued importance of involving human evaluators in LLM assessment, especially for high-stakes domains.

For the text-to-SQL domain, we developed the QAS, a continuous metric that integrates semantic similarity and execution result similarity. The QAS addresses fundamental limitations of existing binary metrics such as EM and EX. We demonstrated that EM penalizes semantically equivalent queries that differ in syntactic details, while EX provides only binary feedback that obscures the degree of similarity between generated and reference queries. Our experimental evaluation across 11 text-to-SQL models showed that the QAS enables more sophisticated comparisons by revealing characteristics that binary metrics obscure. The granular nature of the metric allowed us to distinguish between models with similar mean performance based on median values and variability, differences that would be invisible to traditional binary assessments. For instance, we identified that DAILSQL demonstrated greater consistency and robustness than other models with similar average performance by maintaining higher minimum scores and narrower variability ranges. The QAS also provides diagnostic signals about error types. A consistent pattern emerged in which table similarity scores were significantly lower than semantic similarity scores, corresponding to specific error types such as missing aggregation operators, incorrect sorting orders, and differing filtering conditions. Although the metric cannot pinpoint exact errors, it effectively narrows the search space and offers actionable guidance for targeted improvements.

These contributions establish that meaningful LLM evaluation requires moving beyond surface-level metrics toward approaches that measure genuine utility and correctness. Human-centered evaluation remains essential for domains where understanding represents the ultimate success metric, while continuous metrics provide the granularity needed for nuanced model comparisons and error analysis.

RQ1 Answer

Meaningful LLM evaluation requires domain-specific approaches that capture genuine utility. In the legal domain, human-centered evaluation with 75 participants reveals that GPT-4o achieves only 38% comprehension accuracy versus 45–58% for expert-crafted *Massime*, with an inverse expertise-error relationship showing that knowledgeable participants are *more* misled by LLM outputs. In text-to-SQL, the QAS provides continuous assessment combining semantic and execution similarity, revealing quality distinctions—such as consistency differences between models with similar mean performance—that binary metrics like EM and EX systematically miss. These findings establish that evaluation must move beyond surface-level metrics toward human-centered and granular approaches.

8.1.2 RQ2: Democratization of Specialized Knowledge

To what extent can LLMs democratize access to specialized knowledge in the legal and SE domains, and what are the limitations and risks of current systems in these contexts? (The-sis RQ2)

Our research offers a nuanced assessment of the democratization potential of LLMs, highlighting both their promising capabilities and the significant limitations and risks that must be addressed before these systems can reliably function as effective bridges to specialized knowledge.

DEMOCRATIZATION POTENTIAL AND LIMITATIONS. In the legal domain (Chapter 3), we investigated whether LLM-generated summaries can enhance comprehension for individuals without legal expertise compared to reading original judgments. The evidence showed that GPT-4o summaries achieve modest comprehension improvements, primarily through length reduction rather than genuine conceptual simplification. Legal texts encode complex relationships, conditional logic, and precise terminology that resist simplification without loss of meaning. The model condenses judgments exceeding 10000 words to under 1000 words, but this compression introduces accuracy concerns that undermine the democratization promise. The comparison with human-crafted *Massime* revealed the sophisticated nature of expert legal summarization. *Massime* achieved superior performance across multiple dimensions: highest accuracy rates (45–58%), lowest error rates (27–29%), and optimal length-to-information ratios. This superiority re-

8. GENERAL DISCUSSION

flects human experts' ability to identify and extract genuinely essential legal principles while maintaining technical accuracy. Legal experts understand not just what text says but what it means within broader legal contexts, enabling them to preserve critical nuances while eliminating genuinely redundant information. However, **Massime** exhibited higher "I don't know" response rates, reflecting their specialized nature as summaries created for legal professionals. This finding highlights a fundamental tension in knowledge democratization, as content tailored for experts assumes background knowledge that general audiences may not have, whereas content simplified for broader audiences can sacrifice the precision necessary for accurate legal interpretation. Current LLMs have not resolved this tension, producing either expert-oriented summaries that remain inaccessible or simplified summaries that sacrifice accuracy.

Our code generation investigation (Chapter 4) revealed analogous limitations. We demonstrated that current LLMs, regardless of scale, cannot reliably generate correct code for diverse programming tasks. Even state-of-the-art models directly solve only approximately half of benchmark problems, with the remainder producing partially or completely incorrect solutions. Smaller models perform substantially worse, with success rates below 20%, indicating that model scale alone does not guarantee correctness. The failure modes vary systematically with model size. Smaller models produce syntactically invalid or semantically meaningless code, suggesting insufficient understanding of programming constructs. Larger models generate plausible but subtly incorrect solutions, handling common cases while failing on edge conditions or complex logic. Self-correction experiments provided additional evidence of LLM limitations. The inability of models to debug their own outputs, even with explicit test feedback, suggests that error correction requires capabilities beyond current architectures.

RISKS OF CURRENT SYSTEMS. Our empirical findings reveal several categories of risks that must be considered when deploying LLMs for knowledge democratization:

Risk of Misinformation and Misleading Content. The inverse relationship between legal knowledge and GPT-4o error rates represents a particularly concerning finding. Legally knowledgeable participants experienced higher error rates with GPT-4o summaries than participants without legal expertise, suggesting that the model generates plausible-sounding but potentially misleading content. This phenomenon indicates that LLMs can produce outputs that appear authoritative while containing subtle inaccuracies—a risk that is especially dangerous because users may trust these outputs precisely because of their fluent presentation. In legal contexts, such misinformation can lead to serious consequences: misunderstood rights, missed deadlines, inappro-

appropriate actions based on incorrect legal interpretation, or failure to seek professional counsel when needed.

Risk of Over-Reliance and Displacement of Expertise. The cost-benefit analysis adds another dimension to the risk assessment. While **Massive** require extensive expert time and resources, they deliver reliable, accurate summaries. GPT-4o offers dramatic cost reductions and instant generation, but at the expense of reliability. For legal systems considering AI adoption, this trade-off demands careful consideration of use cases and risk tolerance. There is a danger that the economic appeal of automated systems could lead to the displacement of human expertise, potentially degrading the overall quality of specialized knowledge translation over time. As organizations rely more heavily on automated summaries, the institutional knowledge required to produce high-quality human summaries may atrophy.

Risk of Propagating Subtle Errors. In the code generation domain, the pattern of larger models generating plausible but subtly incorrect solutions is particularly concerning for democratization efforts. Non-expert users—the primary beneficiaries of democratization—are least equipped to identify these subtle errors. A domain expert might recognize that a generated function handles edge cases incorrectly, but a novice programmer using LLM-generated code may deploy flawed implementations without awareness of their limitations. This risk is compounded by the fact that such errors may only manifest in production environments, potentially causing failures that are difficult to diagnose.

Risk of Calibration Mismatch. Our findings suggest a calibration mismatch between LLM confidence and accuracy. Models produce outputs with consistent stylistic confidence regardless of their actual correctness. This creates an epistemic hazard for users who cannot distinguish between correct and incorrect outputs based on surface-level presentation. Effective democratization requires not only accurate outputs but also appropriate uncertainty communication—a capability that current systems lack.

IMPLICATIONS FOR DEPLOYMENT. These findings collectively show that, while LLMs hold considerable potential for democratizing access to specialized knowledge, current systems have systematic limitations and risks that preclude reliable deployment without proper safeguards. The gap between apparent fluency and true correctness represents a critical concern that must be addressed through rigorous evaluation methodologies that prioritize correctness over fluency, post-processing techniques to enhance reliability (as discussed in RQ3), user interface designs that effectively communicate uncertainty, human oversight mechanisms for high-stakes applications, and clear documentation of

8. GENERAL DISCUSSION

system limitations for end users.

RQ2 Answer

LLMs offer significant democratization potential but exhibit systematic limitations and risks preventing reliable deployment without safeguards. In the legal domain, GPT-4o achieves modest comprehension gains through length reduction, but human-crafted Massime maintain superior accuracy (45–58% vs. 38%). In code generation, even GPT-4 solves only 52% of benchmark problems; smaller models achieve <20%. Critical risks include: (1) inverse expertise-error relationship where knowledgeable users are more misled by fluent inaccuracies, (2) calibration mismatch between confidence and correctness, and (3) subtle errors undetectable by non-expert users.

8.1.3 RQ3: Post-Processing and Hybrid Approaches

8

What post-processing and hybrid approaches can enhance the reliability of LLM outputs in high-stakes applications? (Thesis RQ3)

Our research demonstrated that hybrid approaches combining LLM generation with complementary optimization techniques can substantially improve the reliability of LLM outputs, addressing limitations that the models themselves cannot overcome through self-correction. We developed a GI framework that combines LLM code generation with evolutionary optimization. The approach uses GE with dynamically specialized grammars to refine generated code through test-driven fitness evaluation. Our empirical evaluation across 25 problems and 5 models demonstrated that GI is able to improve in a statistically significant way the code generated by LLMs, with some cases achieving perfect correctness from completely incorrect initial solutions. The success of GI can be attributed to several factors. The search process systematically explores variations that maintain syntactic validity while altering semantic behavior. Test-driven fitness offers objective guidance that LLMs do not have during generation. Additionally, the population-based approach preserves diversity, mitigating the commitment problem that limits autoregressive generation. Grammar specialization proved crucial to GI effectiveness. By incorporating problem-specific elements from both the natural language description and LLM output, the search space becomes tractable while retaining necessary expressiveness. This dynamic adaptation represents a key innovation, as manually crafted grammars would be impractical for real-world deployment. The correlation between grammar extraction success and GI performance highlights the

importance of leveraging LLM knowledge extraction alongside evolutionary search. The superiority of GI over self-correction approaches reveals fundamental limitations in current LLM architectures. While LLMs excel at pattern recognition and syntactic generation, they lack the systematic reasoning required for debugging and logical refinement. Evolutionary approaches complement these capabilities by providing objective fitness evaluation and exploratory search mechanisms that escape local optima in the solution space.

The effectiveness of GI varies systematically with both model quality and problem characteristics. Smaller models benefit more from enhancement, as their outputs typically contain correct high-level structure but flawed implementation details, which represent errors well-suited to evolutionary refinement. The outputs of larger models, which tend to be either correct or fundamentally flawed, provide limited opportunity for incremental improvement. The success of evolutionary enhancement is also strongly influenced by the problem domain: tasks involving string manipulation or numerical computation exhibit consistent improvement due to well-defined correctness criteria and modular solution structures, whereas complex algorithmic problems with interdependent logic are more resistant to evolutionary optimization.

These findings have substantial implications for SE practice. As LLMs become integrated into development workflows through tools such as GitHub Copilot, ensuring code correctness becomes paramount. Our approach provides a principled post-processing mechanism that enhances reliability without requiring model retraining or architectural modifications. This compatibility with existing infrastructure facilitates immediate adoption while research continues on more fundamental improvements to generative models. The grammar-based approach ensures transferability across programming languages, requiring only BNF specification for new language support. This generality is particularly valuable for underrepresented languages where LLMs exhibit weaker performance due to limited training data. As programming languages continue to proliferate and evolve, approaches that gracefully handle linguistic diversity become increasingly important.

Our contributions demonstrate that the gap between LLM capabilities and the reliability required for deployment in consequential domains can be addressed through hybrid approaches that combine neural generation with systematic optimization. These techniques complement LLM generation with verification, refinement, and optimization steps that improve output quality without requiring changes to the underlying models.

RQ3 Answer

Hybrid approaches combining LLM generation with evolutionary optimization substantially improve reliability. GI with dynamically specialized grammars significantly improves LLM-generated code across 25 problems and 5 models, with some cases reaching perfect correctness from completely incorrect initial solutions. Success factors include: test-driven fitness providing objective guidance, population-based diversity preventing premature convergence, and dynamic grammar specialization enabling domain adaptation. The superiority of GI over self-correction confirms that LLMs lack systematic debugging capabilities, validating hybrid approaches as essential for high-stakes applications.

8.1.4 RQ4: Environmental Sustainability and Resource Optimization

How can we develop and deploy LLM-based systems while minimizing their environmental footprint and ensuring sustainable resource utilization? (Thesis RQ4)

This thesis provides evidence that computational efficiency and task effectiveness need not be mutually exclusive, contributing practical frameworks for sustainable LLM deployment that align with **green AI** principles, while acknowledging that the limited evaluation budgets in both studies constrain the degree to which convergence to globally optimal trade-offs can be claimed.

We developed HotCat, a pipeline that employs MOO using NSGA-II to identify feature subsets for bug classification that balance predictive accuracy with computational efficiency. The experimental evaluation revealed that accuracy improvements can be achieved without corresponding increases in resource consumption. The configuration achieving the highest accuracy (63%) required only 132 seconds, merely three seconds more than the best-balanced configuration. This finding supports the **green AI** principle that computational efficiency and model effectiveness can be jointly optimized. The feature selection mechanism in HotCat serves dual purposes: it reduces the dimensionality of input data, thereby decreasing LLM prompt sizes and inference costs, while simultaneously improving classification quality by eliminating noisy or redundant features. Data augmentation improved generalization performance from 55% to 72% accuracy, suggesting that expanded training data can address current limitations without necessarily increasing computational demands during inference.

For coding agents, we introduced GA4GC, a framework that identifies non-dominated agent configurations through MOO. The experimental evaluation on the SWE-Perf

benchmark demonstrated that GA4GC achieves up to 135-fold HV improvement over default configurations, with runtime reductions of up to 37.7% while simultaneously improving correctness. Four out of five discovered non-dominated configurations dominate the default baseline in multiple objectives, addressing both “greener agent” (reduced computational cost) and “greener code” (improved performance) requirements. These results were obtained under a constrained budget of 25 evaluations, meaning that the discovered solutions represent a promising approximation to the Pareto front rather than a guarantee of global optimality; nevertheless, they substantially outperform both default and random baselines. The comprehensive influence analysis revealed that temperature is the most critical hyperparameter, with moderate values (0.65–0.73) enabling the exploration necessary for discovering effective optimizations. This finding has practical implications for default configurations: the common practice of using temperature 0.0 for code generation tasks may be suboptimal when the goal involves discovering non-obvious optimizations rather than simply producing correct code. LLM hyperparameters primarily impact task effectiveness, while agent constraints predominantly affect resource consumption. This separation confirms the necessity of multi-objective rather than single-objective optimization for sustainable agent deployment. Single-objective optimization that focuses exclusively on any one dimension may compromise others, whereas multi-objective approaches provide practitioners with non-dominated configurations that make trade-offs explicit. GA4GC addresses the “break-even problem” identified in prior work, wherein energy consumed by optimization agents can require hundreds of thousands of code executions to offset. By reducing agent runtime while improving correctness, GA4GC effectively lowers the break-even threshold. For frequently-executed code, even modest performance improvements become environmentally beneficial when optimization overhead is minimized. Configuration optimization represents an underexplored sustainability lever. Traditional approaches to sustainable AI focus on model architecture, hardware efficiency, or carbon-aware scheduling. Our research demonstrates that substantial efficiency gains are achievable without model retraining, hardware upgrades, or infrastructure changes. This finding suggests that sustainability audits should include configuration analysis alongside traditional efficiency measures.

These contributions align with the GSE principles and Net Zero targets established by international climate initiatives. The MOO approaches developed in this thesis provide practical strategies for reducing environmental impact while maintaining effectiveness, contributing to a future where AI development can balance capability with environmental responsibility.

RQ4 Answer

Computational efficiency and task effectiveness are jointly optimizable through MOO. HotCat demonstrates that maximum accuracy (63%) requires only marginally more runtime than balanced configurations (132s vs. 129s), while data augmentation improves generalization from 55% to 72% accuracy. GA4GC achieves 135-fold HV improvement with 37.7% runtime reduction while improving correctness; temperature emerges as the most critical hyperparameter. Configuration optimization represents an underexplored sustainability lever: substantial efficiency gains are achievable without model retraining, hardware upgrades, or infrastructure changes, aligning with green AI principles.

8.2 Implications and Broader Impact

The research presented in this thesis has implications that extend beyond the specific technical contributions to broader considerations about the responsible development and deployment of LLM technology.

The human-centered evaluation methodology we employed provides a template for assessing AI systems in domains where human understanding represents the ultimate success metric. This approach reveals limitations invisible to automatic metrics, highlighting that evaluation methodologies must evolve alongside the systems they assess. As LLMs become increasingly capable of producing fluent, plausible-sounding outputs, the risk of mistaking surface-level quality for genuine utility grows correspondingly. Our findings underscore the continued importance of involving human evaluators in LLM assessment, especially for high-stakes applications.

The democratization potential of LLMs represents a significant opportunity with profound societal implications, yet our research reveals that this potential is currently constrained by systematic limitations. Legal texts, code, and other specialized content remain challenging for LLMs to process reliably. The gap between apparent fluency and genuine correctness has direct consequences for individuals who might rely on LLM outputs for consequential decisions. In legal contexts, misunderstood rights can lead to violated protections; in SE, subtle bugs can cause failures in production systems. These findings argue for careful consideration of deployment contexts and appropriate safeguards when LLMs serve as intermediaries between specialized knowledge and general audiences.

The hybrid approaches developed in this thesis demonstrate that the limitations of LLMs need not be absolute constraints. By combining neural generation with systematic optimization, we can achieve reliability levels that neither approach would attain independently. This finding has implications for system design: rather than waiting for LLMs to become perfect, practitioners can deploy hybrid systems that leverage the strengths of multiple approaches while compensating for their respective weaknesses.

The sustainability contributions of this thesis address an increasingly pressing concern. As LLMs continue to scale in size and capability, their environmental footprint grows correspondingly. Our preliminary evidence that efficiency and effectiveness can be jointly optimized provides a promising path forward that does not require sacrificing one for the other, though further validation with larger evaluation budgets and across diverse deployment scenarios is needed to confirm the generality of these findings. The frameworks developed here can be extended to other domains and applications, contributing to a broader agenda of sustainable AI development.

For the scientific community, our research provides empirical evidence and methodological templates that can inform future investigations. The detailed analysis of model capabilities and limitations across diverse domains contributes to a more nuanced understanding of what LLMs can and cannot accomplish. The MOO frameworks demonstrate how search-based approaches can address emerging challenges in AI system configuration. For practitioners, our research provides actionable insights for LLM deployment. The evaluation methodologies can guide assessment strategies; the hybrid approaches can enhance reliability; the configuration optimization frameworks can improve efficiency. These contributions bridge the gap between research findings and practical application, enabling organizations to deploy LLM technology more responsibly.

8.3 Reflections on the Present and Future of LLMs

The findings of this thesis, viewed collectively, invite broader reflection on the role of LLMs in society and the directions that research should pursue.

THE ACCESSIBILITY-ACCURACY PARADOX. Our work reveals a recurring pattern across domains: LLMs excel at producing fluent, accessible outputs but struggle to maintain accuracy, particularly in high-stakes contexts. In the legal domain, GPT-4o summaries improve accessibility while simultaneously increasing error rates. In code generation, LLM

8. GENERAL DISCUSSION

outputs appear structurally sound but contain subtle logical errors. This paradox represents a fundamental challenge: the very fluency that makes LLMs useful for democratization also makes their errors harder to detect, creating a “trust gap” between perceived and actual reliability. Looking forward, this suggests that LLMs should not be viewed as replacements for domain expertise but as tools that augment human capabilities. In the legal domain, our findings suggest a promising hybrid approach: rather than replacing *Massime* with LLM-generated summaries, LLMs could assist legal experts in producing *Massime* more efficiently, combining the scalability of automation with the precision of expert judgment. Similarly, in software engineering, our GI results demonstrate that the most effective approach combines LLM generation with systematic optimization—neither alone achieves what the hybrid can.

GENETIC IMPROVEMENT AS A COMPLEMENTARY PARADIGM. This work on GI raises a provocative question: should the research community focus on making LLMs better at code generation, or should it invest more in post-processing approaches that can systematically correct LLM output? The evidence from this thesis strongly favors hybridization over either approach in isolation. LLMs excel at generating plausible initial solutions and extracting domain-specific building blocks (constants, library functions, structural patterns), while GI excels at systematic, test-driven refinement that ensures functional correctness. Indeed, when provided with appropriate building blocks extracted from LLM output, GI improves code more reliably than LLMs can improve their own output: the self-correction experiments in Chapter 4 demonstrated that LLMs struggle to improve through iterative prompting, validating that a fundamentally different search mechanism—such as evolutionary search—is needed for post-hoc improvement. This finding argues for a hybrid paradigm where LLMs provide initial solutions and domain knowledge, while evolutionary methods provide the systematic search capability needed for correctness guarantees. Future research should explore deeper integration of these paradigms through co-evolutionary approaches where LLMs and evolutionary methods jointly evolve both the code and the grammars used for improvement, creating self-improving systems that leverage the complementary strengths of neural and evolutionary computation.

SOCIETAL IMPLICATIONS. The democratization potential documented in this thesis carries significant societal implications. If LLMs can genuinely improve access to legal, medical, and technical knowledge, they could reduce information asymmetries that perpetuate inequality. However, our finding that knowledgeable participants are *more*

misled by LLM errors than novices raises concerns about a new form of “automated misinformation”—plausible-sounding but incorrect content that erodes rather than enhances understanding. Responsible deployment requires not just technical improvements but also user education about LLM limitations and appropriate calibration of trust.

LESSONS LEARNED AND FUTURE OUTLOOK. Reflecting on the entirety of this doctoral work, several overarching lessons emerge about the current state and future trajectory of LLMs. First, the gap between perceived and actual capability remains the single most dangerous characteristic of these systems: their fluency creates an illusion of understanding that can be more harmful than obvious incompetence, as demonstrated by the inverse expertise-error relationship in our legal study. Second, the democratization promise of LLMs is real but conditional—current systems can broaden access to specialized knowledge only when complemented by appropriate safeguards, human oversight, and user education about limitations. Third, sustainability is not merely a desirable property but a prerequisite for responsible deployment at scale, and our multi-objective optimization results suggest that the field has barely begun to explore the configuration space available for efficiency gains.

Looking forward, the rapid pace of LLM advancement raises both optimism and concern. On the positive side, larger context windows, improved reasoning capabilities, and better multilingual support will likely address some of the limitations documented in this thesis. On the negative side, the increasing opacity of commercial models, the growing computational requirements, and the risk of automation bias—where users defer to LLM outputs without critical evaluation—pose challenges that technical improvements alone cannot resolve.

8.4 Limitations

Several limitations constrain the interpretation and generalization of our findings.

In the legal domain, our focus on Italian Constitutional Court judgments, while providing consistency, may not generalize to other legal document types with different structures and complexity levels. The Italian civil law context may yield different results than common law systems with their emphasis on precedent and case law. Our participant pool, while including individuals with varying legal knowledge, may not fully represent the broader population’s diversity in education, language proficiency, and motivation for

8. GENERAL DISCUSSION

understanding legal texts.

For code generation and GI, our evaluation relies on the PSB2 benchmark, which represents a particular class of algorithmic challenges. Performance on other domains, such as web development, systems programming, or ML, may differ substantially. The relatively small problem size and clear specifications may not reflect real-world SD complexity. Additionally, our evaluation includes specific model versions that may not represent current best practices given the rapid advancement in LLM capabilities.

The text-to-SQL evaluation faces computational limitations regarding table similarity calculations for queries producing very large result tables. The linear relationship between computation time and cell count means that some practical scenarios may be challenging to evaluate with current implementations.

For the sustainability optimization studies in Chapters 6 and 7, a fundamental constraint is the limited number of NSGA-II evaluations imposed by the high computational cost of each fitness evaluation. In HotCat, 20 individuals evolved over 20 generations; in GA4GC, only 25 configurations were evaluated (population of 5 across 5 generations), with each evaluation requiring 15–60 minutes. Under such constrained budgets, NSGA-II cannot be reasonably expected to converge to the true Pareto front [37]. Consequently, the non-dominated solutions reported in both chapters should be interpreted as promising configurations that substantially improve over default settings, rather than globally optimal trade-offs. Alternative optimization strategies specifically designed for expensive evaluations—such as surrogate-assisted evolutionary algorithms [81] or Bayesian multi-objective optimization [139]—may achieve better convergence within the same evaluation budget and represent a natural direction for future work. Additionally, our evaluation of GA4GC focuses on a subset of the SWE-Perf benchmark (the astropy project), which may limit generalizability across different domains, programming languages, and project types. Runtime served as a proxy for computational efficiency; direct energy and emissions measurement would provide a more accurate sustainability assessment.

These limitations define the scope within which our findings should be interpreted and point toward directions for future research.

8.5 Concluding Remarks

The emergence of LLM represents both an opportunity and a challenge for the research community and society at large. These systems demonstrate capabilities that seemed unattainable mere years ago, offering the potential to transform how humans interact

with complex information and automated systems. Yet this transformative potential is accompanied by significant concerns, which this thesis has addressed: evaluation methodologies that inadequately capture true system capabilities, limitations that compromise reliability in high-stakes applications, and environmental costs that pose challenges to sustainability.

This thesis has contributed a comprehensive framework for evaluating, improving, and sustainably deploying LLM-based systems. We developed evaluation methodologies that provide meaningful assessments of LLM capabilities across diverse domains, demonstrating that human-centered evaluation and continuous metrics reveal insights invisible to traditional approaches. We investigated the extent to which current systems can democratize access to specialized knowledge, finding both promising capabilities and significant limitations that must be acknowledged. We proposed hybrid approaches that enhance reliability through complementary optimization techniques, demonstrating that evolutionary methods can address limitations that LLMs cannot overcome through self-correction. We contributed to sustainable deployment through MOO frameworks that balance performance against resource consumption, providing evidence that efficiency and effectiveness can be jointly optimized even under constrained evaluation budgets.

This thesis offers concrete tools: human-centered evaluation for legal summarization, GI for code reliability, QAS for query assessment, and multi-objective optimization for sustainable deployment. Each contribution addresses a specific limitation of current LLM systems with methods that practitioners can apply directly. As we advance toward increasingly sophisticated AI systems, maintaining focus on human-centered outcomes remains essential. The path forward requires continued collaboration between AI researchers, domain experts, and the communities these systems aim to serve, ensuring that technological progress translates into genuine benefit for society.

A

Appendix

A.1 Example Legal Materials from Chapter 3

This appendix provides two concrete examples of the materials used in the human-centered evaluation presented in Chapter 3. All materials are in Italian, as the study focused on Italian Constitutional Court judgments. Throughout this appendix, the notation [. . .] indicates that the original text continues beyond the excerpt shown. In the comprehension questions, correct answers are highlighted in **bold** and marked with **(CORRECT)**. The examples include:

A.1.1 Sentenza n. 213/2021

ORIGINAL JUDGMENT

1- Con ordinanza depositata il 24 aprile 2021, iscritta al n. 107 del registro ordinanze 2021, il Tribunale ordinario di Trieste, in funzione di giudice dell'esecuzione, ha sollevato questioni di legittimità costituzionale dell'art. 103, comma 6, del decreto-legge 17 marzo 2020, n. 18 (Misure di potenziamento del Servizio sanitario nazionale e di sostegno economico per famiglie, lavoratori e imprese connesse all'emergenza epidemiologica da COVID-19), convertito, con modificazioni, nella legge 24 aprile 2020, n. 27, con cui è stata disposta la sospensione dell'esecuzione dei provvedimenti di rilascio degli immobili sino alla data del 1 settembre 2020; dell'art. 17-bis del decreto-legge 19 maggio 2020, n. 34 (Misure urgenti in materia di salute, sostegno al lavoro e all'economia, nonché di politiche sociali connesse all'emergenza epidemiologica da COVID-19), convertito, con modificazioni, nella legge 17 luglio 2020, n. 77, con cui è stata disposta la proroga della suddetta sospensione sino alla data del 31 dicembre 2020; nonché dell'art. 13, comma 13, del decreto-legge 31 dicembre 2020, n. 183, recante Disposizioni urgenti in materia di termini legislativi, di realizzazione di collegamenti digitali, di esecuzione della decisione (UE, EURATOM) 2020/2053 del Consiglio, del 14 dicembre 2020, nonché in materia di recesso del Regno Unito dall'Unione europea, convertito, con modificazioni, nella legge 26 febbraio 2021, n. 21, che ha ulteriormente prorogato la sospensione sino alla data del 30 giugno 2021, in riferimento agli artt. 3, 24, 42, 47, 77 e 117, primo comma, della Costituzione, quest'ultimo in relazione all'art. 6 della Convenzione per la salvaguardia dei diritti dell'uomo e delle libertà fondamentali (CEDU), firmata a Roma il 4 novembre 1950, ratificata e resa esecutiva con legge 4 agosto 1955, n. 848, e all'art. 1 del Protocollo addizionale alla CEDU, firmato a Parigi il 20 marzo 1952. [...]

MASSIMA

Sebbene l'esecuzione coattiva delle decisioni giudiziarie rientri nell'alveo dell'art. 24 Cost., in quanto essa è fondamentale per una tutela effettiva dei diritti accertati in sede cognitiva, tuttavia, il legislatore dispone di un'ampia discrezionalità nella conformazione degli istituti processuali, incontrando il solo limite della manifesta irragionevolezza o arbitrarietà delle scelte compiute, limite che, con riferimento specifico all'art. 24 Cost., viene superato solo qualora emerga un'ingiustificabile compressione del diritto di azione. (Precedenti: S. 128/2021 - mass. 43959; S. 80/2020 - mass. 42556; S. 58/2020 - mass. 42158; S. 47/2020 - mass. 42301; O. 3/2020 - mass. 42222). (Nel caso di specie, sono dichiarate non fondate le questioni di legittimità costituzionale, sollevate dai Tribunali di Trieste e di Savona in riferimento all'art. 24 Cost., dell'art. 13, comma 13, del d.l. n. 183 del 2020, come conv., e dell'art. 40-quater del d.l. n. 41 del 2021, come conv., che hanno previsto, rispettivamente al 30 giugno 2021 e al 31 dicembre 2021, la proroga della sospensione delle procedure esecutive per rilascio degli immobili locati. Le norme censurate non impediscono in via definitiva all'avente diritto al rilascio di promuovere un'azione esecutiva per rientrare in possesso dell'immobile, limitandosi, in presenza di una situazione di carattere eccezionale e imprevedibile come l'emergenza pandemica, a differire temporaneamente la possibilità di agire in executivis. Resta ferma in capo al legislatore, ove l'evolversi dell'emergenza epidemiologica lo richieda, la possibilità di adottare altre misure più idonee per realizzare un diverso bilanciamento, ragionevole e proporzionato).

GPT-4O SUMMARY

La sentenza del Tribunale ordinario di Trieste del 24 aprile 2021 ha sollevato questioni di legittimità costituzionale relative a diverse disposizioni normative che avevano prorogato la sospensione dell'esecuzione dei provvedimenti di rilascio degli immobili durante l'emergenza epidemiologica da COVID-19. In particolare, sono state impugnate le seguenti norme: Art. 103, comma 6, del decreto-legge n. 18 del 2020 (convertito nella legge n. 27 del 2020), che sospendeva l'esecuzione dei provvedimenti di rilascio degli immobili fino al 1° settembre 2020. Art. 17-bis del decreto-legge n. 34 del 2020 (convertito nella legge n. 77 del 2020), che prorogava questa sospensione fino al 31 dicembre 2020. Art. 13, comma 13, del decreto-legge n. 183 del 2020 (convertito nella legge n. 21 del 2021), che ulteriormente prorogava la sospensione fino al 30 giugno 2021. Il giudice rimettente ha dubitato della compatibilità di queste disposizioni con diversi articoli della Costituzione italiana (artt. 3, 24, 42, 47, 77 e 117) e con la Convenzione Europea dei Diritti dell'Uomo (CEDU), sostenendo che: Le proroghe della sospensione mancavano dei presupposti di necessità e urgenza richiesti dall'art. 77 della Costituzione. La sospensione generalizzata dell'esecuzione, senza considerare la correlazione causale tra morosità e pandemia, violava il principio di uguaglianza sancito dall'art. 3 della Costituzione. La misura poteva costituire un'espropriazione sostanziale senza indennizzo, in contrasto con la tutela della proprietà (art. 42 Cost.) e del risparmio (art. 47 Cost.). La sospensione temporanea comprometteva il diritto del creditore a ottenere soddisfazione, essenziale per la tutela giurisdizionale (art. 24 Cost.). Le norme impugnate violavano anche l'art. 117, primo comma, della Costituzione in relazione agli artt. 6 CEDU e 1 Prot. addiz. CEDU, incidendo negativamente sulla durata ragionevole del processo e sul diritto di proprietà. Nel procedimento si sono costituiti l'esecutante e il Presidente del Consiglio dei Ministri, difeso dall'Avvocatura generale dello Stato, chiedendo rispettivamente l'accoglimento e il rigetto delle questioni di legittimità costituzionale. Il giudice dell'esecuzione del Tribunale di Savona ha sollevato questioni simili di legittimità costituzionale con riferimento agli artt. 3, 11, 24, 41, 42, 111 e 117 della Costituzione, alla CEDU e alla Carta dei diritti fondamentali dell'Unione Europea (CDFUE), proclamata a Nizza il 7 dicembre 2000 e adattata a Strasburgo il 12 dicembre 2007. La Corte Costituzionale ha riunito i due giudizi incidentali per essere definiti con un'unica decisione. Ha ritenuto non fondate le questioni di legittimità costituzionale, affermando che le misure impugnate erano giustificate dall'eccezionalità dell'emergenza pandemica e avevano rispettato i principi di temporaneità e proporzionalità. Inoltre, ha riconosciuto la discrezionalità del legislatore nel bilanciare i diritti coinvolti, tenendo conto della grave crisi sanitaria ed economica. In conclusione, la Corte ha dichiarato non fondate le questioni di legittimità costituzionale sollevate dai Tribunali di Trieste e Savona, stabilendo che la proroga della sospensione dell'esecuzione dei provvedimenti di rilascio degli immobili era conforme alla Costituzione e alle norme sovranazionali richiamate.

FINE-TUNED LLAMA 2 SUMMARY

Il Tribunale di Trieste dubita, in riferimento agli artt. 03, 24, 42, 47, 77 e 117, primo comma, della Costituzione, quest'ultimo in relazione all'art. 06 della Convenzione per la salvaguardia dei diritti dell'uomo e delle libertà fondamentali (CEDU), firmata a Roma il 4 novembre 1950, ratificata e resa esecutiva con legge 4 agosto 1955, n. 848, e all'art. 1 del Protocollo addizionale alla CEDU, firmato a Parigi il 20 marzo 1952, di alcune norme statali - tra cui, in particolare, l'art. 017-bis del decreto-legge 19 maggio 2020, n. 034 (Misure urgenti in materia di salute, sostegno al lavoro e all'economia, nonché di politiche sociali connesse all'emergenza epidemiologica da COVID-19), convertito, con modificazioni, nella legge 17 luglio 2020, n. 077, e l'art. 013, comma 13, del decreto-legge 31 dicembre 2020, n. 0183 (Disposizioni urgenti in materia di termini legislativi, di realizzazione di collegamenti digitali, di esecuzione della decisione (UE, EURATOM) 2020/2053 del Consiglio, del 14 dicembre 2020, nonché in materia di recesso del Regno Unito dall'Unione europea), convertito, con modificazioni, nella legge 26 febbraio 2021, n. 021, che hanno disposto la proroga della sospensione fino alla data del 31 dicembre 2020 e poi fino al 30 giugno 2021 - che sarebbero lesive del citato precetto costituzionale di ragionevole durata del processo, in quanto si tratterebbe di misure che vanificherebbero, in sostanza, ogni possibilità di accertamento della tempestività della morosità nei confronti degli acquirenti, impedendone altresì la verifica attraverso l'accertamento del pagamento effettuale nell'arco temporale del periodo di tempo stabilito dai commi 3 e seguenti dello stesso art. 017-bis, così violando anche l'art. 047 della Carta dei diritti fondamentali dell'Unione europea (CDFUE). A tale conclusione indurrebbe, ancora, la circostanza che la stessa Corte di cassazione avrebbe affermato, con sentenze rese nei giorni 15 e 19 ottobre 2020, che la disciplina dettata dalle predette disposizioni inciderebbe direttamente sulla tutela giurisdizionale del diritto di proprietà, in forza del quale l'acquisto di beni immobiliari deve essere considerato strumentale ad assicurare il godimento di quel diritto, così rendendo possibile l'adozione delle misure censurate, ponendoli in contrasto con l'art. 42 della Costituzione. Al riguardo, il giudice a quo evidenzia che sia la natura meramente eventuale del meccanismo di proroga della sospensione prevista dagli artt. 17-bis e 13 del d.l. n. 183 del 2020, sia il suo ambito applicativo limitatissimo - relativo alla sola fase esecutiva del rapporto immobiliare e alla sola ipotesi di morosità -, farebbero emergere l'evidente arbitrarietà delle medesime disposizioni, tanto da determinare la loro irragionevolezza sotto il profilo della disparità di trattamento tra cittadini privi e coloro che ne sono dotati. Da ciò deriverebbe, pertanto, secondo il rimettente, anche il contrasto con l'art. 3 Cost. e con i parametri interposti invocati. Infine, l'art. 17-bis del d.l. n. 34 del 2020 sarebbe in contrasto con l'art. 117, primo comma, Cost., in relazione all'art. 6 della CEDU, poiché il meccanismo sopra descritto sarebbe incompatibile con il principio della garanzia del diritto di proprietà, atteso che, sebbene il diritto di proprietà debba subire il sacrificio necessario alle esigenze di tutela pubblica in caso di emergenza, tuttavia lo stesso dovrebbe comunque trovare adeguata protezione presso l'autorità giudiziaria, in base al combinato disposto degli artt. 610 e 614 cod. proc. civ.; il richiamo all'art. 6 della CEDU sarebbe, infatti, introdotto proprio dal denunciato art. 17-bis del d.l. n. 34 del 2020, che precisa, in proposito, che «Nell'applicazione delle disposizioni qui previste [...] si tiene presente il diritto internazionale riconosciuto a tutti gli uomini di avere la propria vita privata protetta contro interferenze ingiustificate della potenza pubblica. In tal guisa, il giudice a quo ritiene che il sistema normativo vigente viola il principio di proporzionalità, posto a fondamento della tutela del diritto di proprietà, in quanto il divieto di eseguire gli atti esecutivi in tema di morosità, già originatosi in virtù della proroga operativa dall'art. 13 del d.l. n. 183 del 2020, viene ulteriormente esteso ai primi dieci mesi dopo la fine della proroga, con conseguente pregiudizio della certezza del diritto e della celerità del processo, quali requisiti posti a base della tutela giurisdizionale del diritto di proprietà, così violandone la finalità, oltre che l'art. 42 della Costituzione. Sussisterebbe, infine, la dedotta lesione dell'art. 24 Cost., in quanto il sistema normativo denunciato, limitando la facoltà di agire in ordine all'esecuzione dei provvedimenti di rilascio in causa di morosità, determinerebbe uno specifico ostacolo ai mezzi di difesa delle parti private nel giudizio esecutivo, con compromissione del diritto di azionarsi in giudizio e la possibilità di ottenere la rideterminazione dell'ammontare del prezzo convenuto.

COMPREHENSION QUESTIONS

Q1: Quale tra queste norme è indicato come parametro costituzionale nella sentenza in esame?

- **Art. 11 Costituzione (CORRECT)**
- Art. 6 Convenzione per la salvaguardia dei diritti dell'uomo
- Art. 53 Costituzione
- Art. 13 comma 13 Decreto Legge n. 183 del 31/12/2020
- Non lo so

A. APPENDIX

Q2: Qual è la causa di connessione?

- Stessi soggetti nei due procedimenti
- **Stesso oggetto dei procedimenti (CORRECT)**
- Impossibilità di decidere su uno dei procedimenti senza aver preventivamente deciso sull'altro
- Stessa fattispecie costitutiva
- Non lo so

Q3: A quale data si è esaurita l'efficacia della sospensione dell'esecuzione dei provvedimenti di rilascio degli immobili?

- 31 dicembre 2020
- 28 febbraio 2020
- 30 giugno 2021
- **31 dicembre 2021 (CORRECT)**
- Non lo so

A.1.2 Sentenza n. 320/2000

ORIGINAL JUDGMENT

1.1. Nel corso di un procedimento penale concernente, tra l'altro, il reato di diffamazione a mezzo stampa (artt. 595, commi secondo e terzo, cod. pen., e 13 e 21 della legge 8 febbraio 1948, n. 47) addebitato al deputato Amedeo Maticena, il giudice per le indagini preliminari presso il Tribunale di Reggio Calabria ha promosso, con ordinanza del 19 febbraio 1999, conflitto di attribuzione tra poteri dello Stato nei confronti della Camera dei deputati, in relazione alla deliberazione, adottata il 9 dicembre 1998, con la quale la Camera, accogliendo la proposta della Giunta per le autorizzazioni a procedere, ha dichiarato che i fatti per i quali è in corso il procedimento penale concernono opinioni espresse dal deputato nell'esercizio delle sue funzioni, con conseguente insindacabilità a norma dell'art. 68, primo comma, della Costituzione. 1.2. Il deputato Maticena premette, in fatto, il giudice che solleva il conflitto è stato imputato di diffamazione in relazione a un articolo, non firmato, apparso sul giornale *Il quotidiano della Calabria* in data 26 marzo 1997; in detto articolo si dava conto della presentazione di una denuncia da parte di Amedeo Maticena senior padre del deputato, e imputato anch'egli nel medesimo procedimento penale, con l'imputazione di calunnia contro Vincenzo Macrò, magistrato addetto alla Direzione nazionale antimafia; la denuncia, secondo il testo dell'articolo, era volta ad accertare se il PM nazionale Vincenzo Macrò avesse avuto o no titolo a percepire la diaria che normalmente compete ai funzionari dello Stato che vengono distaccati in luoghi dove non hanno alcuna dimora. Maticena senior - prosegue il testo dell'articolo - fa riferimento al fatto che il magistrato alla data del 17 aprile 1993 era residente a Reggio Calabria. E per dimostrare che la residenza in città dura tuttora, tocca il tasto del servizio di vigilanza attorno all'abitazione del magistrato, argomento che per le forti critiche fatte in passato sono costate all'anziano armatore una condanna per diffamazione del giudice Macrò. Nell'articolo, poi, veniva riferito che la denuncia riportava la risposta scritta del Ministro di grazia e giustizia a un'interrogazione parlamentare del deputato Amedeo Maticena, figlio del denunciante, sul medesimo argomento; in detta risposta del Ministro, sintetizzata anch'essa nell'articolo, si chiariva ... che i magistrati di quell'ufficio [la Direzione nazionale antimafia] dimorano di fatto a Roma e percepiscono l'indennità di missione allorché svolgono attività fuori dell'ordinaria sede di servizio e del luogo di abituale dimora. Avviate le indagini sulla base di denuncia-querela proposta dal magistrato nei confronti di entrambi i Maticena, e formulata dal pubblico ministero richiesta di rinvio a giudizio degli imputati, la Camera dei deputati, su richiesta del suo componente, ha dichiarato, con deliberazione del 9 dicembre 1998, su conforme proposta della Giunta per le autorizzazioni a procedere, che i fatti per i quali è in corso il procedimento penale concernono opinioni espresse dal deputato nell'esercizio delle sue funzioni, con conseguente insindacabilità a norma dell'art. 68, primo comma, della Costituzione. La deliberazione della Camera prosegue l'ordinanza che ha promosso il conflitto si incentra sul rilievo, formulato dalla Giunta e fatto proprio dall'Assemblea, secondo il quale l'antecedente logico di tutta la vicenda è costituito da un atto di sindacato ispettivo presentato dall'on. Maticena, "cioè dall'interrogazione parlamentare al Ministro di grazia e giustizia n. 4-02698 del 31 luglio 1996, cui indiscibilmente si connetterebbero le vicende successive, dall'iniziativa del padre del deputato, all'articolo pubblicato sul giornale; quest'ultimo pertanto rappresenterebbe, per la Camera, divulgazione di un'attività parlamentare e, per ciò stesso, un comporta-

mento scriminato dall'art. 68, primo comma, della Costituzione. 1.3. Ciò premesso, il ricorrente osserva che, alla stregua delle risultanze processuali e dello stesso articolo (il quale esordisce enunciando che il deputato di Forza Italia on. Amedeo Matacena ha reso noto che il padre ...); non sussistono dubbi né circa lo svolgimento dei fatti né circa la riferibilità di essi al deputato; si tratta dunque, prosegue il ricorrente, soltanto di verificare se all'attività posta in essere dal deputato possa dirsi applicabile la prerogativa costituzionale e se, in particolare, sia ravvisabile nel caso in esame il nesso funzionale tra l'attività tipica del parlamentare (l'interrogazione) e quella successiva ed esterna (la pubblicazione), nesso che giustifica l'estensione alla seconda della garanzia posta per la prima. A tale riguardo, il giudice critica la qualificazione di attività parlamentare che la Camera ha attribuito alla pubblicazione nel caso di specie, poiché il taglio complessivo dell'informazione, ispirata dal deputato, non consentirebbe questa qualificazione: l'articolo esordisce annunciando una iniziativa giudiziaria di chi non è deputato - il padre del parlamentare - e prosegue menzionando precedenti controversie giudiziarie tra la stessa persona e il magistrato, risoltesi peraltro a favore di quest'ultimo; solo in chiusura dell'articolo, si dà conto di stralci della risposta (in data 3 aprile 1997) del Ministro all'interrogazione del deputato Matacena. Nel suo assieme e per il suo divulgare iniziative assunte nei confronti di un magistrato da una persona, non parlamentare, che con il primo coltiva una remota conflittualità, l'articolo pubblicato non potrebbe dunque ricollegarsi alla funzione parlamentare svolta dal deputato Matacena, funzione che ha natura personalissima e che non è estensibile a terzi. Anzi, la pubblicazione in discorso riveste, per il ricorrente, il carattere di uno sfogo pubblico di iniziative giudiziarie assunte dal genitore del deputato, e lo stesso sintetico accenno al contenuto della risposta del Ministro appare, in questo quadro, un espediente, strumentale alla cattura dell'interesse del pubblico su una vicenda che rimane tutta privata. [...]

MASSIMA

Al fine dell'applicabilità della speciale garanzia prevista dal primo comma dell'art. 68 Cost. alle dichiarazioni rilasciate da membri del Parlamento al di fuori di attività parlamentari tipiche - nella specie, da un deputato a un organo di stampa - e ciononostante riconducibili o inerenti alla funzione parlamentare, distinguendole così da quelle che ricadono nel diritto comune a tutti i cittadini, non basta la semplice comunanza di argomenti, oggetto di attività parlamentari tipiche e di dichiarazioni fatte al di fuori di esse, né basta la riconducibilità di queste ultime dichiarazioni a un medesimo contesto politico. Occorre invece che la dichiarazione possa essere qualificata come espressione di attività parlamentare, il che normalmente accade se e in quanto sussista una sostanziale corrispondenza di significati tra le dichiarazioni rese al di fuori dell'esercizio delle attività parlamentari tipiche svolte in Parlamento e le opinioni già espresse nell'ambito di queste ultime. E dal momento che sussiste sostanziale corrispondenza tra il contenuto delle dichiarazioni affidate alla stampa dal deputato Matacena e il contenuto di atti parlamentari precedenti - in particolare, considerando l'interrogazione da lui presentata il 31 luglio 1996 -, spetta alla Camera dei deputati affermare l'insindacabilità, ai sensi dell'art. 68, primo comma, della Costituzione, delle dichiarazioni espresse dal deputato Matacena, secondo quanto deliberato dall'Assemblea della Camera il 9 dicembre 1998 in relazione al procedimento penale per diffamazione a mezzo stampa, pendente presso l'ufficio giudiziario ricorrente. - Per l'ammissibilità del conflitto, v. ordinanza n. 319/1999. - In ordine a dichiarazioni identificabili come espressione di attività parlamentare, cfr. anche sentenze nn. 10 e 11/2000 e 56 e 58/2000; 329/1999, nonché n. 375/1997.

GPT-4O SUMMARY

Contesto del Procedimento Procedimento Penale: Il giudice per le indagini preliminari presso il Tribunale di Reggio Calabria ha promosso un conflitto di attribuzione nei confronti della Camera dei deputati. Il caso riguarda il reato di diffamazione a mezzo stampa, imputato al deputato Amedeo Matacena, derivante da un articolo pubblicato sul giornale Il quotidiano della Calabria. Delibera della Camera: La Camera dei deputati ha deliberato che i fatti per cui Matacena è in corso di procedimento penale concernono opinioni espresse nell'esercizio delle sue funzioni parlamentari, dichiarando così l'insindacabilità a norma dell'art. 68, primo comma, della Costituzione. Argomenti del Giudice Nesso Funzionale: Il giudice contesta l'affermazione della Camera, sostenendo che la pubblicazione dell'articolo non può essere qualificata come attività parlamentare. Ritiene che la pubblicazione non sia strettamente collegata all'interrogazione parlamentare di Matacena, ma piuttosto a iniziative giudiziarie private del padre del deputato, rendendo quindi arbitraria l'insindacabilità. Difesa della Camera Insindacabilità

delle Opinioni: La Camera sostiene che le dichiarazioni di Maticena siano coperte dall'insindacabilità poiché strettamente collegate a un'interrogazione parlamentare. L'articolo di stampa riprenderebbe il contenuto dell'interrogazione, rientrando così nella garanzia costituzionale. Procedura e Merito: La Camera eccepisce l'irricevibilità e l'inammissibilità del ricorso per questioni formali, ma anche nel merito, afferma che vi sia una chiara connessione funzionale tra l'attività parlamentare e le dichiarazioni pubblicate. Decisione della Corte Costituzionale Ammissibilità del Conflitto: La Corte costituzionale ha dichiarato ammissibile il conflitto e rigettato le eccezioni preliminari sollevate dalla Camera, riconoscendo la validità formale dell'atto introduttivo del giudizio. Nesso Funzionale Confermato: La Corte ha stabilito che le dichiarazioni di Maticena sono sostanzialmente corrispondenti all'interrogazione parlamentare, quindi rientrano nella sfera delle attività parlamentari coperte dalla garanzia costituzionale dell'art. 68, primo comma. Conclusione: Il giudizio sul conflitto di attribuzione è stato risolto a favore della Camera dei deputati, riconoscendo l'insindacabilità delle dichiarazioni di Maticena.

FINE-TUNED LLAMA 2 SUMMARY

La questione sottoposta all'esame della Corte riguarda: se gli atti di cui all'impugnato articolo debbano essere ricondotti alle opinioni espresse dall'Ambiente Maticena nell'esercizio delle funzioni parlamentari. L'ordinanza di rimessione afferma che tale questione sarebbe rilevante ed in subordine non manifestamente infondata. Sul punto, infatti, il relatore osserva che nella specie la difesa degli imputati si fonda sull'assunto che l'articolo censurato sia frutto di mero squalificamento, ma senza specificarne i profili; ciò posto, l'attuale questione, poiché considera come 'principale', e pertanto prioritaria, quella delle funzioni parlamentari in ordine alle quali si assume che vi sia stata la loro esplicitazione, determina la necessità di esaminare la relativa questione di competenza. La rilevanza della stessa viene, altresì, motivata nel senso che l'eventuale pronuncia favorevole sulla natura costituzionale della dichiarazione, ne determinerebbe l'influenza sugli eventuali successivi provvedimenti processuali. Invero, qualora il riconoscimento dell'insindacabilità fosse negativo, dovrebbero essere annullate tutte le decisioni prese in merito alla questione di insindacabilità. A sostegno della fondatezza della questione di competenza, il giudice remittente pone il seguente argomento: a) la valutazione oggettiva della condotta attribuita all'imputato presupporrebbe un 'giusto apprezzamento' della sua portata politica e del contesto politico e sociale in cui si inserisce; b) solo l'appartenenza alla categoria dei membri del Parlamento consente di ravvisare una connessione causale tra l'atto e la posizione occupata; c) il collegamento tra il fatto extra moenia e lo status deve risultare evidente e concreto; d) la diversa interpretazione posta dalla Procura regionale e dalla Camera, non consentirebbe di individuare il requisito soggettivo della funzione parlamentare; e) il fatto che, al momento del discorso, l'onorevole Maticena fosse già candidato per la carica di Presidente della Provincia di Reggio Calabria sarebbe evocabile anche quando la dedotta estraneità alla funzione parlamentare non possa trovare ragionevole applicazione. Quando, quindi, la delibera della Camera di appartenenza sia restata sconosciuta, e sopravvenga l'ipotesi che la Corte ritenga che il fatto stigmatizzato sia contestualmente configurabile come atto rivolto ad esprimere opinioni in seno al Parlamento, occorrerebbe che la stessa Camera si riconsideri sulle modalità attraverso cui ha disposto la declaratoria di insindacabilità; in tal caso, non mancherebbe altro che la decisione di revoca. Considerato che il giudice per le indagini preliminari presso il tribunale di Reggio Calabria dubita della competenza della Camera dei deputati, per membra dell'Assemblea Parlamentare europea, ad emanare la deliberazione della quale è causa, e chiede al nostro giudizio se questa Corte possa, per quanto attiene alla questione di competenza, dichiarare che la Camera aveva omesso il proprio obbligo di decidere, e, conseguentemente, annullare la citata deliberazione. Circa la prima questione (quella di competenza), va anzitutto tenuto presente che, in materie come quelle concernenti la cognizione di fatti, l'autonomia legislativa e amministrativa, la tutela dei diritti e interessi di singoli cittadini, la regolamentazione delle elezioni e delle istanze politiche, la disciplina delle associazioni civili private, oltre a tutte quelle altre materie nelle quali intervento può avere lo Stato, in virtù della delega operata al Governo dagli artt. 70 e segg. della Costituzione, la Camera dei deputati - ove sia chiamata a pronunciare sulla responsabilizzazione di membri suoi propri - non gode della competenza, né può goderne alcuno tipo di potestà di carattere generale o speciale. Tuttavia, nella specie, il giudice a quo deduce che il riconoscimento dell'insindacabilità prevista dall'art. 68, primo comma, Cost. (che attribuisce allo stesso organo la facoltà di dichiarare non punibili certi comportamenti dei membri del Parlamento) potrebbe influire sulla sentenza definitiva di proscioglimento, modificandola positivamente o negativamente, e, conseguenzialmente, influenzare gli effetti connessi alla sentenza, come quello della revoca. Né tale possibilità di influenzare la sentenza finale può ritenersi priva di importanza, considerato che, se pure i fatti contestati fossero stati ritenuti colpevoli, la loro grav-

ità e la violazione della libertà di pensiero e di parola da parte di un rappresentante del potere legislativo avrebbero dovuto far supporre l'adozione di una pronuncia rigida e severa, che avrebbe comunque determinato la revoca della immunità. Tanto premesso, va ora precisato che la questione di competenza è, per sé, irrilevante, perché l'organo parlamentare, nell'emergere in tal formazione di una controversia di valore politico, non ha mai potuto intervenire, né in sede istruttoria né in sede decisoria, tanto meno con una pronuncia di carattere vincolante. Infatti, mentre i magistrati di sorveglianza avevano chiesto alla Camera di approvarsi una nota di presentazione di querela ex art. 595 c.p., in quanto l'operato del magistrato Macrò rientrava nella tipologia di falsa testimonianza, l'Ufficio parlamentare della P.M. aveva chiesto invece, in sede di analisi della autenticità del documento allegato all'atto impugnato, il parere della Camera, secondo la quale si trattasse di un semplice elenco di dipendenti della polizia, non necessario per la formulazione del capo di imputazione, e non pertinente per la verifica della identità personale e professionale dell'autore del documento. Neppure il Consiglio superiore della magistratura aveva chiesto l'approvazione del proprio parere sugli elementi probatori, e ancora meno la Camera aveva chiesto la sentenza del tribunale di sorveglianza.

COMPREHENSION QUESTIONS

Q1: Cosa chiede la Camera dei deputati?

- Ricorso venga dichiarato irricevibile
- Ricorso venga dichiarato inammissibile
- Ricorso venga dichiarato irricevibile ed inammissibile
- **Ricorso venga dichiarato irricevibile ed inammissibile e venga riconosciuta la sussistenza degli elementi dell'insindacabilità (CORRECT)**
- Non lo so

Q2: La garanzia di insindacabilità copre:

- Le opinioni contenute negli atti tipici della funzione
- **Le opinioni contenute negli atti tipici della funzione e le opinioni per cui sia stata accertata l'esistenza del nesso causale tra opinioni e mandato parlamentare (CORRECT)**
- Tutte le opinioni espresse all'interno dell'aula parlamentare
- Anche le opinioni personali estranee alla sfera politica espresse al di fuori dell'aula parlamentare
- Non lo so

Q3: La Corte ritiene le eccezioni della Camera fondate?

- È fondata la richiesta di dichiarazione di irricevibilità, ma non quella di dichiarazione di inammissibilità
- È fondata la richiesta di dichiarazione di inammissibilità, ma non quella di dichiarazione di irricevibilità
- Sono entrambe ritenute fondate
- **Sono entrambe ritenute infondate (CORRECT)**
- Non lo so

B

Appendix

In this section, we present all the SQL queries used to evaluate the proposed metric. These queries belong to groups 1, 2, 3, 4, 5 and 6, as introduced in Section 5.4. In group 1, we incremented the number of columns and left the number of rows unchanged.

```
SELECT mailstreet
FROM schools
WHERE cdscode IN (
  SELECT cdscode
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 1
);
```

Query 1: Group 1 - query 1: one column, one row, and descending order.

```
SELECT mailstreet, mailstrabr
FROM schools
WHERE cdscode IN (
  SELECT cdscode
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 1
);
```

Query 2: Group 1 - query 2: two columns, one row, and descending order.

B. APPENDIX

B

```
SELECT mailstreet, mailstrabr, mailcity
FROM schools
WHERE cdscod IN (
  SELECT cdscod
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 1
);
```

Query 3: Group 1 - query 3: three columns, one row, and descending order.

```
SELECT mailstreet, mailstrabr, mailcity, mailzip
FROM schools
WHERE cdscod IN (
  SELECT cdscod
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 1
);
```

Query 4: Group 1 - query 4: four columns, one row, and descending order.

In group 2, we incremented the number of rows and left the number of columns unchanged.

```
SELECT mailstreet
FROM schools
WHERE cdscod IN (
  SELECT cdscod
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 1
);
```

Query 5: Group 2 - query 1: one column, one row, and descending order.

```
SELECT mailstreet
FROM schools
WHERE cdscod IN (
  SELECT cdscod
```

```

FROM frpm
ORDER BY "frpm count (k-12)" DESC
LIMIT 2
);

```

Query 6: Group 2 - query 2: one column, two rows, and descending order.

```

SELECT mailstreet
FROM schools
WHERE cdscode IN (
  SELECT cdscode
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 3
);

```

Query 7: Group 2 - query 3: one column, three rows, and descending order.

```

SELECT mailstreet
FROM schools
WHERE cdscode IN (
  SELECT cdscode
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 4
);

```

Query 8: Group 2 - query 4: one column, four rows, and descending order.

In group 3, we examine how the metric behaves as the number of columns increases from 1 to 4, while maintaining a constant number of rows at 4.

```

SELECT mailstreet
FROM schools
WHERE cdscode IN (
  SELECT cdscode
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 4
);

```

Query 9: Group 3 - query 1: one column, four rows, and descending order.

```
SELECT mailstreet, mailstrabr
FROM schools
WHERE cdscod IN (
  SELECT cdscod
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 4
);
```

Query 10: Group 3 - query 2: two columns, four rows, and descending order.

```
SELECT mailstreet, mailstrabr, mailcity
FROM schools
WHERE cdscod IN (
  SELECT cdscod
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 4
);
```

Query 11: Group 3 - query 3: three columns, four rows, and descending order.

```
SELECT mailstreet, mailstrabr, mailcity, mailzip
FROM schools
WHERE cdscod IN (
  SELECT cdscod
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 4
);
```

Query 12: Group 3 - query 4: four columns, four rows, and descending order.

In group 4, we examine how the metric behaves as the number of rows increases from 1 to 4, while maintaining a constant number of columns at 4.

```
SELECT mailstreet, mailstrabr, mailcity, mailzip
FROM schools
WHERE cdscod IN (
  SELECT cdscod
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 1
);
```

Query 13: Group 4 - query 1: one column, four rows, and descending order.

```
SELECT mailstreet, mailstrabr, mailcity, mailzip
FROM schools
WHERE cdscod IN (
  SELECT cdscod
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 2
);
```

Query 14: Group 4 - query 2: two columns, four rows, and descending order.

```
SELECT mailstreet, mailstrabr, mailcity, mailzip
FROM schools
WHERE cdscod IN (
  SELECT cdscod
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 3
);
```

Query 15: Group 4 - query 3: three columns, four rows, and descending order.

```
SELECT mailstreet, mailstrabr, mailcity, mailzip
FROM schools
WHERE cdscod IN (
  SELECT cdscod
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
```

B. APPENDIX

B

```
LIMIT 4
);
```

Query 16: Group 4 - query 4: four columns, four rows, and descending order.

In group 5, we simultaneously increased both rows and columns.

```
SELECT mailstreet
FROM schools
WHERE cdscod IN (
  SELECT cdscod
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 1
);
```

Query 17: Group 5 - query 1: one column, one row, and descending order.

```
SELECT mailstreet, mailstrabr
FROM schools
WHERE cdscod IN (
  SELECT cdscod
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 2
);
```

Query 18: Group 5 - query 2: two columns, two rows, and descending order.

```
SELECT mailstreet, mailstrabr, mailcity
FROM schools
WHERE cdscod IN (
  SELECT cdscod
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 3
);
```

Query 19: Group 5 - query 3: three columns, three rows, and descending order.

```

SELECT mailstreet, mailstrabr, mailcity, mailzip
FROM schools
WHERE cdscod IN (
  SELECT cdscod
  FROM frpm
  ORDER BY "frpm count (k-12)" DESC
  LIMIT 4
);

```

Query 20: Group 5 - query 4: four columns, four rows, and descending order.

In group 6, we switched the order from descending (DESC) to ascending (ASC) and progressively increased the number of rows.

```

SELECT mailstreet
FROM schools
WHERE cdscod IN (
  SELECT cdscod
  FROM frpm
  ORDER BY "frpm count (k-12)" ASC
  LIMIT 1
);

```

Query 21: Group 6 - query 1: one column, one row, and ascending order.

```

SELECT mailstreet
FROM schools
WHERE cdscod IN (
  SELECT cdscod
  FROM frpm
  ORDER BY "frpm count (k-12)" ASC
  LIMIT 2
);

```

Query 22: Group 6 - query 2: one column, two rows, and ascending order.

```

SELECT mailstreet
FROM schools
WHERE cdscod IN (
  SELECT cdscod

```

B. APPENDIX

B

```
FROM frpm
ORDER BY "frpm count (k-12)" ASC
LIMIT 3
);
```

Query 23: Group 6 - query 3: one column, three rows, and ascending order.

```
SELECT mailstreet
FROM schools
WHERE cdscode IN (
  SELECT cdscode
  FROM frpm
  ORDER BY "frpm count (k-12)" ASC
  LIMIT 4
);
```

Query 24: Group 6 - query 4: one column, four rows, and ascending order.

Bibliography

- [1] Marah Abdin et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.
- [2] Flavia Aचना, David Preti, Davide Venditti, Leonardo Ranaldi, Cristina Giannone, Fabio Massimo Zanzotto, Andrea Favalli, and Raniero Romagnoli. Legal summarization: to each court its own model. In *Proceedings of the 9th Italian Conference on Computational Linguistics (CLiC-it 2023)*, pages 10–16, 2023.
- [3] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [4] Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. A comparative survey of recent natural language interfaces for databases. *The VLDB Journal*, 28(5):793–819, 2019.
- [5] Anthropic. Claude 3 model card. 2024. URL <https://www.anthropic.com/claude>.
- [6] Anthropic. Claude’s performance on SWE-bench. 2025. URL <https://www.anthropic.com>.
- [7] Mari Ashiga, Vardan Voskanyan, Fateme Dinmohammadi, Jingzhi Gong, Paul Brookes, Matthew Truscott, Rafail Giavrimis, Mike Basios, Leslie Kanthan, and Wei Jie. Industrial llm-based code optimization under regulation: A mixture-of-agents approach. *arXiv preprint arXiv:2508.03329*, 2025.
- [8] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.
- [9] Thomas Bäck and Hans-Paul Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.
- [10] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.
- [11] Pierpaolo Basile, Elio Musacchio, Marco Polignano, Lucia Siciliani, Giuseppe Fiameni, and Giovanni Semeraro. Llamantino: Llama 2 models for effective text generation in italian language. *arXiv preprint arXiv:2312.09993*, 2023.

- [12] Peter Belcak et al. Small language models are the future of agentic AI. *arXiv preprint arXiv:2506.02153*, 2025.
- [13] Julian Blank and Kalyanmoy Deb. pymoo: Multi-objective optimization in Python. *IEEE Access*, 8:89497–89509, 2020. doi: 10.1109/ACCESS.2020.2990567.
- [14] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.
- [15] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [16] Bobby R Bruce, Justyna Petke, and Mark Harman. Reducing energy consumption using genetic improvement. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1327–1334, 2015.
- [17] Bruce G Buchanan and Thomas E Headrick. Some speculation about artificial intelligence and legal reasoning. *Stan. L. Rev.*, 23:40–62, 1970.
- [18] Ruisheng Cao et al. Lgesql: Line graph enhanced text-to-sql model with mixed local and non-local relations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2541–2555, 2021.
- [19] Gemma Catolino, Fabio Palomba, Andy Zaidman, and Filomena Ferrucci. Not all bugs are the same: Understanding, characterizing, and classifying bug types. *Journal of Systems and Software*, 152:165–181, 2019. doi: 10.1016/j.jss.2019.03.002.
- [20] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45, 2024.
- [21] Dikshit Chauhan, Bapi Dutta, Indu Bala, Niki Stein, Thomas Bäck, and Anupam Yadav. Evolutionary computation and large language models: A survey of methods, synergies, and applications. *arXiv preprint arXiv:2505.15741*, 2025.
- [22] Angelica Chen, David Dohan, and David So. Evoprompting: Language models for code-level neural architecture search. *Advances in neural information processing systems*, 36:7787–7817, 2023.
- [23] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- [24] Ram Chillarege, Inderpal S Bhandari, Jarir K Chaar, Michael J Halliday, Diane S Moebus, Bonnie K Ray, and Man-Yuen Wong. Orthogonal defect classification—a concept for in-process measurements. *IEEE Transactions on software Engineering*, 18(11):943–956, 1992.

- [25] DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *Computational Linguistics*, 47(2):309–332, 2021.
- [26] Agnieszka Ciborowska et al. Data augmentation for software engineering tasks. *Information and Software Technology*, 2023.
- [27] Tristan Coignon, Clément Quinton, and Romain Rouvoy. When faster isn't greener: The hidden costs of LLM-based code optimization. In *Proceedings of the 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2025.
- [28] Gheorghe Comanici et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.
- [29] Benoit Courty, Victor Schmidt, Goyal-Kamal, Marion Coutarel, Boris Feld, Jérémy Lecourt, and Lex Loft. CodeCarbon: Estimate and track carbon emissions from machine learning computing. Zenodo, <https://doi.org/10.5281/zenodo.11171501>, 2024.
- [30] Noel Cressie and Timothy RC Read. Multinomial goodness-of-fit tests. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 46(3):440–464, 1984.
- [31] Deborah A. Dahl et al. Expanding the scope of the atis task: The atis-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.
- [32] Giordano d'Aloisio, Tosin Fadahunsi, Jay Choy, Rebecca Moussa, and Federica Sarro. Sustain-Diffusion: Optimising the social and environmental sustainability of Stable Diffusion models. In *Proceedings of the 47th IEEE/ACM International Conference on Software Engineering (ICSE)*, 2026.
- [33] Luis De la Cal, Yazhuo Cao, Ayse Irmak Ercevik, Giovanni Pinna, Lukas Twist, David Williams, Karine Even-Mendoza, William B. Langdon, Hector D. Menendez, and Federica Sarro. Hotcat: Green and effective feature selection toward hotfix bug taxonomy. In *Proceedings of the 17th Symposium on Search-Based Software Engineering*, Cham, Switzerland, 2025. Springer. In press.
- [34] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001.
- [35] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014. doi: 10.1109/TEVC.2013.2281535.
- [36] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017.

- [37] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2): 182–197, 2002.
- [38] Aniket Deroy, Kripabandhu Ghosh, and Saptarshi Ghosh. How ready are pre-trained abstractive models and llms for legal case judgement summarization? *arXiv preprint arXiv:2306.01248*, 2023.
- [39] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient fine-tuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.
- [40] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, volume 1 (long and short papers)*, pages 4171–4186, 2019.
- [41] Xuemei Dong et al. C3: Zero-shot text-to-sql with chatgpt. *arXiv preprint arXiv:2307.07306*, 2023.
- [42] Abhimanyu Dubey et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- [43] Karine Even-Mendoza et al. PatchCat: Automatic categorization of software patches. *Automated Software Engineering*, 2025.
- [44] David Fagan, Michael O’Neill, Edgar Galván-López, Anthony Brabazon, and Seán McGarraghy. A survey of grammatical evolution. *Handbook of Natural Computing*, pages 1209–1238, 2013.
- [45] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. CodeBERT: A pre-trained model for programming and natural languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings (EMNLP)*, pages 1536–1547. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.139.
- [46] Dawei Gao et al. Text-to-sql empowered by large language models: A benchmark evaluation. *Proceedings of the VLDB Endowment*, 17(5):1132–1145, 2024.
- [47] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.
- [48] Gemma Team et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.
- [49] GitHub. mini-swe-agent. <https://github.com/pppyb/mini-swe-agent>, 2024.
- [50] Jingzhi Gong, Sisi Li, Giordano d’Aloisio, Zishuo Ding, Yulong Ye, William B. Langdon, and Federica Sarro. GreenStableYolo: Optimizing inference time and image quality of text-to-image generation. In *Proceedings of the 13th Symposium on Search-Based Software Engineering (SS-BSE)*, volume 14767 of LNCS, pages 70–76. Springer, 2024. doi: 10.1007/978-3-031-64573-0_7.

- [51] Jingzhi Gong, Yixin Bian, Luis De la Cal, Giovanni Pinna, Anisha Uteem, David Williams, Mar Zamorano, Karine Even-Mendoza, William B. Langdon, Hector D. Menendez, and Federica Sarro. Ga4gc: Greener agent for greener code via multi-objective configuration optimization. In *Proceedings of the 17th Symposium on Search-Based Software Engineering*. Springer, 2025. arXiv:2510.04135.
- [52] Jingzhi Gong, Giovanni Pinna, Yixin Bian, and Jie M Zhang. Analyzing message-code inconsistency in ai coding agent-authored pull requests. 2026.
- [53] Lin Gong et al. Tuning large language models for code optimization. *Empirical Software Engineering*, 2025.
- [54] Maarten Grootendorst. Keybert: Minimal keyword extraction with bert, 2020.
- [55] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. Towards complex text-to-sql in cross-domain database with intermediate representation. pages 4524–4535, 2019.
- [56] Jiaqi Guo et al. Towards complex text-to-sql in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, 2019.
- [57] Christoph Hanna, Federica Sarro, Mark Harman, and Justyna Petke. HotBugs.jar: A benchmark of hot fixes for time-critical bugs. *arXiv preprint arXiv:2510.07529*, 2025.
- [58] Mark Harman, S. Afshin Mansouri, and Yuanyuan Zhang. Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys*, 45(1):1–61, 2012. doi: 10.1145/2379776.2379787.
- [59] Mark Harman, Yue Jia, and William B Langdon. Babel pidgin: Sbse can grow and graft entirely new functionality into a real world system. *International Symposium on Search Based Software Engineering*, pages 247–252, 2014.
- [60] Moshe Hazoom, Vibhor Malik, and Ben Bogin. Text-to-sql in the wild: A naturally-occurring dataset based on stack exchange data. *arXiv preprint arXiv:2106.05006*, 2021.
- [61] Ziming He et al. SWE-Perf: Benchmarking code performance optimization via LLM agents. *arXiv preprint*, 2025.
- [62] Thomas Helmuth and Peter Kelly. Psb2: the second program synthesis benchmark suite. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 785–794, 2021.
- [63] Thomas Helmuth, Lee Spector, and James Matheson. Solving uncompromising problems with lexicase selection. *IEEE Transactions on Evolutionary Computation*, 19(5):630–643, 2015.
- [64] Martin Hemberg, Una-May O’Reilly, Achim Menges, Katrin Jonas, Michel da Costa Gonçalves, and Steven Fuchs. Genr8: Architects’ experience with an emergent design tool. In *The Art of Artificial Evolution*, pages 167–188. Springer, 2008.

- [65] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 21(248):1–43, 2020.
- [66] Dan Hendrycks, Steven Basart, Saurabh Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, et al. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*, 2021.
- [67] John H Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. MIT Press, 1992.
- [68] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.
- [69] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.
- [70] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- [71] Dong Huang, Jie M. Zhang, Yuhao Qing, and Heming Cui. EffiBench: Benchmarking the efficiency of automatically generated code. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. Datasets and Benchmarks Track.
- [72] Nargiz Humbatova, Gunel Jahangirova, Gabriele Bavota, Vincenzo Riccio, Andrea Stocco, and Paolo Tonella. Taxonomy of real faults in deep learning systems. In *Proceedings of the 42nd International Conference on Software Engineering (ICSE)*, pages 1110–1121. ACM, 2020. doi: 10.1145/3377811.3380395.
- [73] Shashikant Ilager, Lukas Florian Briem, and Ivona Brandic. Green-Code: Learning to optimize energy efficiency in LLM-based code generation. In *Proceedings of the 25th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 559–569. IEEE, 2025.
- [74] International Energy Agency. Data centres and data transmission networks. Technical report, IEA, 2020. URL <https://www.iea.org/reports/data-centres-and-data-transmission-networks>.
- [75] Md Rakibul Islam et al. Runtime bug detection and hotfix deployment in production systems. *IEEE Transactions on Software Engineering*, 2023.
- [76] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, 2017.

- [77] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.
- [78] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.
- [79] Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation. *ACM Transactions on Software Engineering and Methodology*, 2024.
- [80] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=VTF8yNQM66>.
- [81] Yaochu Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011.
- [82] George Katsogiannis-Meimarakis, Mike Xydias, and Georgia Koutrika. Natural language interfaces for databases with deep learning. *Proceedings of the VLDB Endowment*, 16(12):3878–3881, 2023.
- [83] Maurice G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.
- [84] Eva Kern, Markus Dick, Stefan Naumann, Achim Guldner, and Timo Johann. Green software and green software engineering—definitions, measurements, and quality aspects. pages 87–91, 2013.
- [85] John R Koza. *Genetic programming: On the programming of computers by means of natural selection*. MIT Press, 1992.
- [86] Wuwei Lan et al. Unite: A unified benchmark for text-to-sql evaluation. *arXiv preprint arXiv:2305.16265*, 2023.
- [87] William B Langdon. Improving software with genetic programming. *IEEE Transactions on Evolutionary Computation*, 19(1):118–135, 2015.
- [88] William B Langdon and Mark Harman. Optimizing existing software with genetic programming. *IEEE Transactions on Evolutionary Computation*, 19(1):118–135, 2015.
- [89] William B Langdon and Gabriela Ochoa. Genetic improvement: A key challenge for evolutionary computation. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 3068–3075. IEEE, 2016.
- [90] Claire Le Goues, ThanhVu Nguyen, Stephanie Forrest, and Westley Weimer. Genprog: A generic method for automatic software repair. *IEEE Transactions on Software Engineering*, 38(1):54–72, 2012.

- [91] Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. Kaggledbqa: Realistic evaluation of text-to-sql parsers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2261–2273, 2021.
- [92] Gyubok Lee et al. Ehrsql: A practical text-to-sql benchmark for electronic health records. *Advances in Neural Information Processing Systems*, 35:15589–15601, 2022.
- [93] Boyan Li, Yuyu Luo, Chengliang Chai, Guoliang Li, and Nan Tang. The dawn of natural language to sql: Are we fully ready? *arXiv preprint arXiv:2406.01265*, 2024.
- [94] Hao Li, Haoxiang Zhang, and Ahmed E. Hassan. The rise of AI teammates in software engineering (SE) 3.0: How autonomous coding agents are reshaping software engineering. *arXiv preprint arXiv:2507.15003*, 2025.
- [95] Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. Resdsq: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence*. AAAI Press, 2023.
- [96] Haoyang Li et al. Codes: Towards building open-source language models for text-to-sql. *Proceedings of the ACM on Management of Data*, 2(3):1–28, 2024.
- [97] Jinyang Li et al. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36, 2024.
- [98] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023.
- [99] Xianming Li and Jing Li. Angle-optimized text embeddings. *arXiv preprint arXiv:2309.12871*, 2023.
- [100] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.
- [101] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 510–520, 2011.
- [102] Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S. Yu. A comprehensive evaluation of chatgpt’s zero-shot text-to-sql capability. *arXiv preprint arXiv:2303.13547*, 2023.
- [103] Jiachang Liu et al. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.
- [104] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36, 2023.

- [105] Zhiwei Liu et al. LLM agents: A comprehensive survey. *arXiv preprint*, 2024.
- [106] Vadim Liventsev, Anastasiia Grishina, Aki Härmä, and Leon Moonen. Fully autonomous programming with large language models. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1146–1155, 2023.
- [107] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.
- [108] Lauren Martin, Nick Whitehouse, Stephanie Yiu, Lizzie Catterson, and Rivindu Perera. Better call gpt: Comparing large language models against lawyers. *arXiv preprint arXiv:2401.16212*, 2024.
- [109] Robert I McKay, Nguyen Xuan Hoai, Peter Alexander Whigham, Yin Shan, and Michael O’Neill. Grammar-based genetic programming: A survey. *Genetic Programming and Evolvable Machines*, 11(3-4):365–396, 2010.
- [110] Brad L Miller, David E Goldberg, et al. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3):193–212, 1995.
- [111] Sewon Min et al. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.
- [112] Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.
- [113] Hoang Duong Thien Nguyen, Dawei Qi, Abhik Roychoudhury, and Satish Chandra. SemFix: Program repair via semantic analysis. In *Proceedings of the 35th International Conference on Software Engineering (ICSE)*, pages 772–781. IEEE, 2013. doi: 10.1109/ICSE.2013.6606623.
- [114] Michael O’Neill and Conor Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358, 2001.
- [115] Michael O’Neill and Conor Ryan. Grammatical evolution: Evolutionary automatic programming in an arbitrary language. *Genetic Programming Series*, 4, 2003.
- [116] Alena Orlenko and Jason H. Moore. A comparison of methods for interpreting random forest models of genetic association in the presence of non-additive interactions. *BioData Mining*, 14(1):9, 2021. doi: 10.1186/s13040-021-00243-0.
- [117] Fatma Özcan et al. State of the union: Natural language interfaces for databases. *ACM SIGMOD Record*, 49(1):33–42, 2020.
- [118] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.
- [119] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refined-web dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023.

- [120] Justyna Petke, Saemundur O Haraldsson, Mark Harman, William B Langdon, David R White, and John R Woodward. Genetic improvement of software: A comprehensive survey. *IEEE Transactions on Evolutionary Computation*, 22(3):415–432, 2018.
- [121] Giovanni Pinna, Damiano Ravalico, Luigi Rovito, Luca Manzoni, and Andrea De Lorenzo. Enhancing large language models-based code generation by leveraging genetic improvement. In *European Conference on Genetic Programming*, pages 108–124. Springer, 2024.
- [122] Giovanni Pinna, Damiano Ravalico, Luigi Rovito, Luca Manzoni, and Andrea De Lorenzo. Improving llm-generated code via genetic improvement: A summary of recent advances. In *5th National Conference on Artificial Intelligence*. CINI, 2025.
- [123] Giovanni Pinna, Damiano Ravalico, Luigi Rovito, Luca Manzoni, and Andrea De Lorenzo. Exploring the effect of genetic improvement for large language models-generated code. *SN Computer Science*, 6(7):760, 2025. doi: 10.1007/s42979-025-04281-x.
- [124] Giovanni Pinna, Jingzhi Gong, David Williams, and Federica Sarro. Comparing ai coding agents: A task-stratified analysis of pull request acceptance. In *Proceedings of the 23rd International Conference on Mining Software Repositories (MSR '26)*, Rio de Janeiro, Brazil, 2026. ACM. doi: 10.1145/3793302.3793583.
- [125] Riccardo Poli, William B Langdon, and Nicholas F McPhee. *A field guide to genetic programming*. Lulu.com, 2008.
- [126] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.
- [127] Colin Raffel et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.
- [128] Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498*, 2022.
- [129] Md Rezaur Haque Rakib, Norbert Zeh, Magdalena Jankowska, and Evangelos Milios. Enhancement of short text clustering by iterative classification. In *Proceedings of the 25th International Conference on Natural Language and Information Systems (NLDB)*, pages 105–117. Springer, 2020. doi: 10.1007/978-3-030-51310-8_10.
- [130] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.
- [131] Siyu Ren, Bill Tomlinson, R.W. Black, et al. Reconciling the contrasting narratives on the environmental impact of large language models. *Scientific Reports*, 14:26310, 2024. doi: 10.1038/s41598-024-76682-6.
- [132] Franz Rothlauf. Representations for genetic and evolutionary algorithms. *Studies in Computational Intelligence*, 2006.

- [133] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.
- [134] Conor Ryan, Michael O’Neill, and JJ Collins. *Handbook of grammatical evolution*. Springer, 2018.
- [135] Diptikalyan Saha et al. Athena: An ontology-driven system for natural language querying over relational data stores. *Proceedings of the VLDB Endowment*, 9(12):1209–1220, 2016.
- [136] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, 2021.
- [137] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.
- [138] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL <https://aclanthology.org/P16-1162/>.
- [139] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2016.
- [140] Alex Shypula et al. Learning to optimize code with genetic improvement. *IEEE Transactions on Software Engineering*, 2024.
- [141] Niki van Stein and Thomas Bäck. Llamea: A large language model evolutionary algorithm for automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation*, 29(2):331–345, 2025. doi: 10.1109/TEVC.2024.3497793.
- [142] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 3645–3650, 2019.
- [143] Richard Susskind and Daniel Susskind. *The Future of the Professions: How Technology Will Transform the Work of Human Experts*. Oxford University Press, 2015.
- [144] Gilbert Syswerda. Uniform crossover in genetic algorithms. *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 2–9, 1989.
- [145] Ning Tao, Anthony Ventresque, Vivek Nallur, and Takfarinas Saber. Grammar-obeying program synthesis: A novel approach using large language models and many-objective genetic programming. *Computer Standards & Interfaces*, 92:103938, 2025.

- [146] Manuel Tapia et al. Evaluation of LLM-based code generation systems. *Journal of Systems and Software*, 2025.
- [147] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [148] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [149] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. volume 30, 2017.
- [150] Roberto Verdecchia, June Sallou, and Luís Cruz. A systematic review of green AI. *WIREs Data Mining and Knowledge Discovery*, 13(4), 2023. doi: 10.1002/widm.1507.
- [151] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, 2020.
- [152] Ping Wang, Tian Shi, and Chandan K. Reddy. Text-to-sql generation for question answering on electronic medical records. In *Proceedings of The Web Conference 2020*, pages 350–361, 2020.
- [153] Song Wang, Taiyue Liu, and Lin Tan. Automatically learning semantic features for defect prediction. In *Proceedings of the 38th International Conference on Software Engineering (ICSE)*, pages 297–308. ACM, 2016. doi: 10.1145/2884781.2884804.
- [154] Zheng Wang and Michael O’Boyle. Machine learning in compiler optimization. *Proceedings of the IEEE*, 106(11):1879–1901, 2018.
- [155] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.
- [156] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.
- [157] Grant Wilkins, Srinivasan Keshav, and Richard Mortier. Offline energy-optimal LLM serving. In *Proceedings of the 3rd Workshop on Sustainable Computer Systems (HotCarbon)*, 2024.
- [158] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of machine learning and systems*, 4:795–813, 2022.
- [159] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

- [160] Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying LLM-based software engineering agents. *arXiv preprint arXiv:2407.01489*, 2024.
- [161] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- [162] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3911–3921, 2018.
- [163] David M Zajic, Bonnie Dorr, Jimmy Lin, and Richard Schwartz. Sentence compression as a component of a multi-document summarization system. In *Proceedings of the 2006 document understanding workshop*, 2006.
- [164] G Zames. Genetic algorithms in search, optimization and machine learning. *Inf Tech J*, 3(1): 301, 1981.
- [165] John M. Zelle and Raymond J. Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1050–1055, 1996.
- [166] Bo Zhang et al. Benchmarking the text-to-sql capability of large language models: A comprehensive evaluation. *arXiv preprint arXiv:2403.02951*, 2024.
- [167] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020.
- [168] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020.
- [169] Yi Zhang et al. Sciencebenchmark: A complex real-world benchmark for evaluating natural language to sql systems. *arXiv preprint arXiv:2306.04743*, 2023.
- [170] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.
- [171] Lei Zhou, Fengwei Zhang, Jinghui Liao, Zhengyu Ning, Jidong Xiao, Kevin Leach, Westley Weimer, and Guojun Wang. KSHOT: Live kernel patching with SMM and SGX. In *Proceedings of the 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 596–607. IEEE, 2020. doi: 10.1109/DSN48063.2020.00073.
- [172] Qinglin Zhu et al. Talk to your data: Natural language interfaces for business intelligence. *Proceedings of the VLDB Endowment*, 16(11):2985–2997, 2023.
- [173] Albert Ziegler, Eirini Kalliamvakou, X Alice Li, Andrew Rice, Devon Rifkin, Shawn Simister, Ganesh Sittampalam, and Edward Aftandilian. Productivity assessment of neural code completion. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*, pages 21–29, 2022.

- [174] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003. doi: 10.1109/TEVC.2003.810758.
- [175] Zsolt Zódi. The limits of plain legal language: Understanding the comprehensible style in law. *International Journal of Law in Context*, 15(3):246–262, 2019.

Publications

Giovanni Pinna*, Damiano Ravalico*, Luigi Rovito, Luca Manzoni, Andrea De Lorenzo, “Enhancing large language models-based code generation by leveraging genetic improvement ,” *European Conference on Genetic Programming (Part of EvoStar)*. Cham: Springer Nature Switzerland, 2024. p. 108-124.

Giovanni Pinna*, Damiano Ravalico*, Luigi Rovito*, Luca Manzoni, Andrea De Lorenzo, “Exploring the Effect of Genetic Improvement for Large Language Models-Generated Code ,” *SN Computer Science*, 2025, 6.7: 1-21.

Giovanni Pinna*, Damiano Ravalico*, Luigi Rovito*, Luca Manzoni, Andrea De Lorenzo, “Improving LLM-Generated Code via Genetic Improvement: A Summary of Recent Advances,” *Ital-IA, Workshop on Generative AI: Generative AI and Education*.

Giovanni Pinna, Davide Tugnoli, Mara Bartole, Luca Manzoni, Andrea De Lorenzo, “From Courts to Comprehension: Can LLMs Make Judgments More Accessible? ,” *2024 IEEE/WIC International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. IEEE, 2024. p. 297-304.

Giovanni Pinna, Yuriy Perezhohin, Luca Manzoni, Mauro Castelli, Andrea De Lorenzo, “Redefining text-to-SQL metrics by incorporating semantic and structural similarity,” *Scientific Reports*, 2025, 15.1: 22357

Jingzhi Gong, Yixin Bian, Luis de la Cal, **Giovanni Pinna**, Anisha Uteem, David Williams, Mar Zamorano, Karine Even-Mendoza, William B. Langdon, Hector D. Menendez, Federica Sarro, “GA4GC: Greener Agent for Greener

Code via Multi-Objective Configuration Optimization” *The 17th Symposium on Search-Based Software Engineering 2025 (SSBSE 2025)*, SSBSE 2025 Challenge Case: Green SBSE.

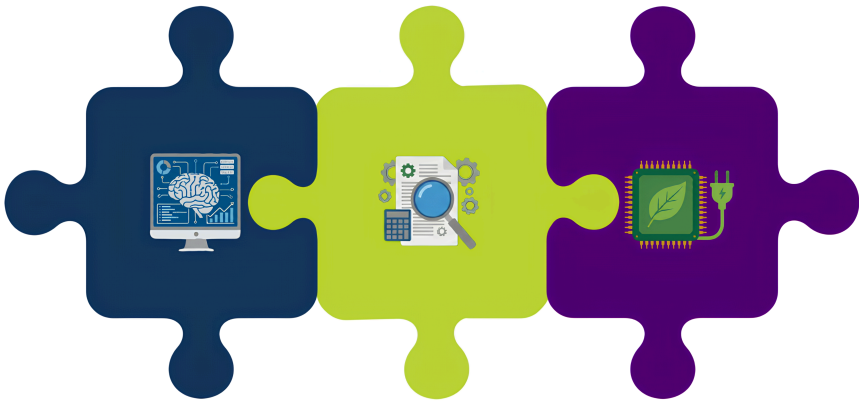
Luis de la Cal, Yazhuo Cao, Irmak Ercevik, **Giovanni Pinna**, Luke Twist, David Williams, Karine Even-Mendoza, William B. Langdon, Hector D. Menendez, Federica Sarro, “HotCat: Green and Effective Feature Selection for HotFix Bug Taxonomy,” *The 17th Symposium on Search-Based Software Engineering 2025 (SSBSE 2025)*: SSBSE 2025 Challenge Case: Hot Fixing Benchmark. 2025.

Jingzhi Gong, **Giovanni Pinna**, Yixin Bian, Jie M Zhang, “Analyzing Message-Code Inconsistency in AI Coding Agent-Authored Pull Requests” *The 23rd International Conference on Mining Software Repositories 2026 (MSR 2026)*, MSR 2026 Mining Challenge.

Giovanni Pinna, Jingzhi Gong, David Williams, Federica Sarro, “Comparing AI Coding Agents: A Task-Stratified Analysis of Pull Request Acceptance” *The 23rd International Conference on Mining Software Repositories 2026 (MSR 2026)*, MSR 2026 Mining Challenge.

*equal contribution.

Logo Description



This logo holds a deep meaning for me. It was designed with a minimalist and playful aesthetic because I believe that research, beyond being a job, should also be a source of pleasure and enjoyment.

Scientific research is by its very nature in constant evolution, and the works I have published—collected in this thesis—are nothing but pieces of an ever-expanding puzzle. My hope is that other researchers will be able to build upon what I have done, adding new pieces to the ones I have already placed.

Each puzzle piece contains an image generated with artificial intelligence: the technology I have studied over these three years, and that continues to fascinate me. Each image is the visual synthesis of one of the three parts of the thesis,

just as three are the puzzle pieces and the years of my PhD.

The first piece is blue, the colour of the University of Trieste and of PLUS, which have accompanied me throughout the entire journey. The image represents the first part of the thesis, *Output Quality and Improvement*, and is a visual synthesis of the outputs of large language models: the brain symbolises the model, while what appears on the screen represents its output. These works were developed primarily at the University of Trieste during the early years of the PhD.

The second piece recalls the months spent at NOVA IMS in Lisbon, and its colour is inspired by the 2024 logo of the Portuguese university. The image evokes the concepts of evaluation and measurement, in reference to the second part of the thesis, *Metrics and Evaluation*, whose main contribution was developed at NOVA IMS.

The third piece is dedicated to the period at UCL in London, and its colour draws from the logo of the London university. The image depicts a chip with a leaf inside, symbolising green AI and sustainability in the field of software engineering and artificial intelligence—themes at the heart of the work carried out at UCL, which form the third part of the thesis *Sustainability and Green AI*.

For me, this is not just a logo: it is a visual representation of my entire PhD journey, a way of telling, in a single image, who I have been and what I have done over these three years.

Curriculum Vitae



Giovanni Pinna obtained his Bachelor's and Master's degrees in Computer Engineering from the University of Trieste. During his Master's studies, he increasingly focused on data science and artificial intelligence, developing a strong expertise in computer vision and natural language processing (NLP). He spent a six-month Erasmus exchange period in Austria, broadening his academic and cultural perspective. His Master's thesis in Computer Vision provided his first experience with academic research and motivated his decision to pursue a doctoral degree.

In November 2022, Giovanni began his Ph.D. in Data Science and Artificial Intelligence at the University of Trieste, under the supervision of Prof. Luca Manzoni and Prof. Andrea De Lorenzo. His doctoral research, funded by the industry partner PLUS s.r.l., investigates the evaluation, improvement, and sustainable deployment of Large Language Models (LLMs) at the intersection of NLP, Software Engineering, and evolutionary computation.

Throughout his Ph.D., Giovanni combined theoretical research with applied innovation, developing Proof of Concepts and delivering solutions in collaboration with the industry partner.

Giovanni's research interests lie at the intersection of natural language processing, evolutionary computation, and software engineering, with a focus on evaluating and improving the outputs of Large Language Models across diverse application domains. His doctoral work addressed the reliability of LLM-generated content in both the legal and software engineering domains, developing novel evaluation metrics for text-to-SQL systems, and investigating multi-objective optimization strategies for sustainable AI deployment in the context of Green AI and green software engineering.

His academic journey has been marked by significant international mobility and collaboration with leading experts. In 2024 and 2025, he spent research periods at the NOVA Information Management School (NOVA IMS) in Lisbon, working with Prof. Mauro Castelli on the development of evaluation metrics for text-to-SQL systems, resulting in a publication in *Scientific Reports*. In the final quarter of 2025, he joined the SOLAR research group at University College London (UCL) as a visiting researcher. There, hosted by Prof. Federica Sarro, he deepened his expertise in search-based software engineering, focusing on sustainable coding agent optimization, AI4Code, and LLM4Code.

These research activities resulted in nine peer-reviewed publications at international venues including *Scientific Reports* (Nature), the European Conference on Genetic Programming (EuroGP), IEEE/WIC WI-IAT, the Symposium on Search-Based Software Engineering (SSBSE), the International Conference on Mining Software Repositories (MSR), and *SN Computer Science*.

Beyond his research activities, Giovanni has been deeply committed to the academic community. He served as a Teaching Assistant for the Databases and SQL course at the University of Trieste and actively mentored and co-supervised several students during the completion of their theses. He also served as a reviewer for international conferences and workshops, contributing to the peer review process in his areas of expertise. Furthermore, he contributed to the broader research ecosystem by volunteering at and attending various summer and winter schools, consistently striving to support and engage with the scientific community.

Acknowledgements

My deepest thanks go to my supervisors, Luca Manzoni and Andrea De Lorenzo, for their invaluable guidance, patience, and unwavering support throughout this journey. You taught me what research truly means.

I gratefully acknowledge *PLUS s.r.l.* for funding this research, with special thanks to Federico and Fulvio for their constant support.

I am indebted to the students I had the privilege of supervising—Damiano and Davide— and work with Luigi, whose dedication and talent continually inspired me. You taught me as much as I hope I taught you.

My sincere appreciation goes to my lab colleagues—Lorenzo, Andrea, Mauro, Gabriele, Federico, Fred, Berfin, Michel, and others—for making this journey lighter.

I am grateful for the enriching international collaborations that shaped my research. Special thanks to Prof. Mauro Castelli at NOVA IMS Lisbon for his generous mentorship and warm hospitality, and to Yuriy for the fruitful collaboration. Thanks also to Prof. Federica Sarro for the opportunity to join the SOLAR Group at UCL London. Thanks also to Prof. Bill Langdon, Dave, and the entire UCL team for their welcome.

I also thank the PhD Secretariat and Prof. Pauli for their administrative support.

Finally, my heartfelt gratitude goes to my friends and family for their unconditional support, and to Ilaria, for walking beside me every step of the way.

Thank you all for making this journey possible.

Acronyms

AI

A broad field of computer science dedicated to creating systems capable of performing tasks that typically require human intelligence, such as reasoning, problem-solving, and perception. 1, 4–6, 8, 19, 20, 22, 23, 25, 31–33, 35, 45–47, 51, 60, 76–79, 117, 135, 136, 148, 159, 163–165, 169

AST

A tree representation of the abstract syntactic structure of source code written in a programming language, where each node of the tree denotes a construct occurring in the source code. 57, 59

BNF

A notation technique for context-free grammars, used to describe the syntax of programming languages, document formats, and instruction sets. 17, 52, 54, 75, 161

CNN

A class of deep neural networks, most commonly applied to analyzing visual imagery, utilizing convolutional layers to automatically learn spatial hierarchies of features. 88

CoT

A prompting technique that encourages the model to generate intermediate logical reasoning steps before providing the final answer. 1, 22

DL

A specialized subset of machine learning based on artificial neural networks with multiple layers, capable of learning complex representations of data through hierarchical processing. 1, 26, 119

DNN

An artificial neural network comprising multiple hidden layers between the input and output layers, capable of modeling complex non-linear relationships and learning hierarchical feature representations. 20

EA

A subset of artificial intelligence utilizing mechanisms inspired by biological evolution (selection, reproduction, mutation) for optimization problems. 7, 18, 52, 55, 79

EM

An evaluation metric that considers a prediction correct only if it matches the reference string exactly (character-by-character or token-by-token). 2, 25, 52, 85, 86, 89, 92, 95, 106, 109, 110, 153, 156, 157

EX

A metric that evaluates the correctness of a generated SQL query by executing it on the database and comparing the result with that of the reference query. 2, 25, 83, 85, 86, 89, 92, 95, 100, 101, 106, 107, 109, 110, 156, 157

FT

The process of re-training a pre-trained model on a specific, smaller dataset to specialize it for a particular domain or task. 22, 36–38, 46, 72, 85, 88

GA

A heuristic search algorithm inspired by Darwin's theory of natural selection, used to solve optimization and search problems. 13, 16–18

GBGI

An approach to genetic improvement where the search for improved software variants is constrained by a formal grammar (such as BNF) to ensure that modifications preserve the syntactic validity of the source code. 54

GBGP

A variation of genetic programming where the generation and evolution of programs are constrained by a formal grammar (typically BNF) to ensure that all solutions are syntactically valid and adhere to a specific structure. 17, 18

GE

A grammar-based form of genetic programming that evolves a linear string of integers (genotype), which are mapped to an executable program (phenotype) using a context-free grammar (usually BNF). 18, 49, 52, 55–58, 62, 160

GI

The use of genetic programming techniques to automatically improve existing software by optimizing functional or non-functional properties (e.g., speed, energy consumption). 7–9, 18, 19, 25, 26, 49, 51, 52, 54, 61, 62, 65–74, 77, 79, 155, 160–162, 166, 168, 169

GNN

A class of neural networks designed to process data represented as graphs, leveraging message-passing mechanisms to capture dependencies between nodes. 88

GP

An evolutionary technique where solutions to problems are represented as computer programs that are evolved automatically. 13, 17–19, 24, 26, 54

green AI

Research and practices in artificial intelligence that prioritize energy efficiency and the reduction of the carbon footprint associated with model training and inference, aiming to make AI systems environmentally sustainable. 5, 7, 9, 19, 115, 117, 119, 120, 126, 129, 133, 135, 162, 164

GSE

An emerging discipline at the intersection of climate science, software practices, and hardware architecture focused on building, running, and maintaining sustainable software systems with minimal environmental impact. 5, 133, 163

HV

A unary quality indicator that measures the volume of the objective space dominated by a set of non-dominated solutions, bounded by a reference point. 135, 139–142, 144, 145, 147, 148, 163, 164

ICL

The ability of a language model to learn a task based on examples provided within the prompt, without updating the model weights. 22, 85, 88, 94

LLM

A deep learning model trained on vast textual datasets, capable of understanding, generating, and manipulating natural language with high proficiency. 1–9, 11, 13, 18–20, 22–26, 29, 31–36, 38, 45, 47, 49, 51–58, 60–66, 68–76, 78, 79, 85, 87, 88, 90, 115, 117–123, 126, 127, 129, 131, 133–138, 143, 145–148, 153–169

LM

A probabilistic model that learns the statistical distribution of words in a language to predict the next sequence of tokens. 18, 26, 88

LoRA

A specific PEFT technique that freezes pre-trained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture. 21

LSTM

A type of Recurrent Neural Network architecture designed to address the vanishing gradient problem, capable of learning long-term dependencies in sequence data. 88

ML

A subset of artificial intelligence that involves the development of algorithms enabling computers to learn patterns from data and make predictions or decisions without explicit programming. 5, 168

MOO

The process of simultaneously optimizing two or more conflicting objective functions, typically resulting in a set of trade-off solutions known as the Pareto front. 16, 19, 25, 117–121, 128, 134–138, 146, 148, 155, 162–165, 169

NLG

A subfield of artificial intelligence and natural language processing that focuses on generating natural language text from non-linguistic data or structured representations. 6

NLP

A multidisciplinary field of computer science and artificial intelligence concerned with the interactions between computers and human languages, focusing on enabling machines to understand, interpret, and generate human text. 1, 2, 31, 33, 85, 88, 153

NMI

A normalization of the Mutual Information score used to compare clustering results against ground truth, scaling the value between 0 (no mutual information) and 1 (perfect correlation). 121–127, 129

PEFT

A set of techniques used to fine-tune large pre-trained models by updating only a small subset of parameters, thereby reducing computational cost and memory usage. 21

QAS

A score used to measure the relevance or semantic compatibility between a user query and database schema elements or retrieved documents. 8, 9, 83, 86, 87, 90, 93–95, 101–105, 107–109, 111, 155–157, 169

RL

A machine learning paradigm where an agent learns to make decisions by interacting with an environment to maximize cumulative reward. 137, 147

SD

The set of computer science activities dedicated to the process of creating, designing, deploying, and supporting software. 51, 73, 76, 78, 117, 168

SE

The application of a systematic, disciplined, and quantifiable approach to the development, operation, and maintenance of software. 5, 19, 22, 26, 49, 51, 76, 119, 129, 133, 136, 148, 157, 161, 164

VES

A metric combining execution correctness with query efficiency, penalizing correct but computationally expensive solutions. 89

Glossary

Massima

A concise summary distilling the essential legal principle from an Italian court decision, crafted by specialized legal offices. Plural: *Massime*. 34, 36, 37

Massime

Plural of *Massima*. Concise expert-crafted summaries of Italian court decisions. 29, 31, 32, 34, 35, 37, 40, 41, 44–47, 155, 157–160, 166