State-of-the-Art Pre-Trained LLMs for Construction Cost Estimation Tasks: A Conceptual Estimation Scenario Using a Modular Chain of Thought (CoT) Approach

Prashnna Ghimire *, Kyungki Kim, Terry Stentz, Tirthankar Roy

Posted Date: 15 October 2025

doi: 10.20944/preprints202510.1060.v1

Keywords: construction cost estimation; AI; large language models (LLMs); modular framework; chain of thought (CoT); prompting

 doi:10.20944/preprints202510.1060.v1

Article

State-of-the-Art Pre-Trained LLMs for Construction Cost Estimation Tasks: A Conceptual Estimation Scenario Using a Modular Chain of Thought (CoT) Approach

Prashnna Ghimire [1,*], Kyungki Kim [1], Terry Stentz [1] and Tirthankar Roy [2]

[1] Durham School of Architectural Engineering & Construction, University of Nebraska-Lincoln, USA

[2] Civil & Environmental Engineering, University of Nebraska-Lincoln, USA

* Correspondence: pghimire3@huskers.unl.edu

Abstract

The traditional cost estimation process in construction involves extracting information from diverse data sources in various data forms and often relying on human intuition and judgment. This makes the estimation process time-intensive, subjective, and susceptible to human errors. The advancement of large language models (LLMs) offers a promising avenue to address these inefficiencies; however, their effectiveness in cost estimation tasks remains unexplored. There are previous studies that have explored LLM applications in different construction problems. However, no study has evaluated how existing pre-trained LLMs perform in cost estimation or provided a structured framework for enhancing their accuracy and reliability through prompt engineering to automate estimation workflows. In our previous study, we identified the key estimation burdens and categorized them into three distinct areas of estimation: (1) conceptual estimation, (2) evaluating subcontractor estimates, and (3) change management, version control, and data recycling. This study specifically evaluates the performance of LLMs, including GPT-4o, LLaMA 3.2, Gemini 2.0, and Claude 3.5 Sonnet, on cost estimation tasks under a conceptual estimation scenario to compare zero-shot prompting with a modular chain-of-thought (CoT) framework. The results of this study indicate that zero-shot prompting produced incomplete and inconsistent responses, with an average confidence score of 1.906 (64%). The CoT framework significantly improved accuracy, achieving a confidence score of 2.52 (84%) and significant improvements across quantitative measurements such as BLEU, ROUGE-L, METEOR, content overlap, and semantic similarity metrics. This study makes three key contributions: (i) develops a cost estimation scenario for LLMs, (ii) evaluates the baseline performance of LLMs in construction cost estimation, and (iii) develops a modular CoT framework that improves estimation performance, which offers a scalable, adaptable approach for industry application. The findings serve as a guideline for integrating AI into construction estimation workflows, bridging the gap between pre-trained LLMs and construction industry applications.

Keywords: construction cost estimation; AI; large language models (LLMs); modular framework; chain of thought (CoT); prompting

1. Introduction

1.1. Background

Construction cost estimation is a systematic process to project the total expenses required to complete a project for its defined scope. For contractors, precise estimates are essential to remain competitive in bidding, whereas, for owners, these estimates inform critical budgeting, scheduling, and resource allocation decisions [1,2]. An accurate estimation involves a clear understanding of

 doi:10.20944/preprints202510.1060.v1

quantities, technical specifications, and prices. At its core process, cost estimation involves identifying necessary materials, labor, and equipment, and then assigning suitable prices to the tasks outlined in the project documentation. Through the alignment of financial expectations with primary project goals, this process acts as a important element of effective planning and decision-making, ensuring that stakeholders have a clear understanding of budgetary demands [3]. One of the most significant components of cost estimation is the quantity takeoff- often referred to as the material takeoff- where estimators compile a detailed inventory of all required materials, labor activities, and equipment [4]. This procedure can follow either two-dimensional or three-dimensional approaches, and it generally benefits from standardized practices such as breaking projects into smaller, more manageable segments to improve both speed and accuracy. However, in large projects or complicated projects, it is easy to overlook even minor tasks or items, and this could lead to significant cost overruns or delays [5,6]. Consequently, frequent cross-checking of measurements and calculations is essential. After determining the necessary quantities, estimators move on to assigning prices. Although this step can be time-intensive, it can draw upon several resources: supplier quotations, subcontractor bids, historical data, industry databases, and professional experience. And, by extracting cost information from multiple sources, estimators can develop comprehensive and reliable forecasts, ultimately helping stakeholders make informed financial decisions and supporting more predictable project outcomes [4,7].

Currently, estimators manually extract cost data from various sources. This not only demands extensive human effort and time but also risks introducing errors that can propagate through estimates [8]. Despite advancements in digital construction tools and AI in construction, estimators still lack an integrated, modular pipeline that can systematically process and extract insights from diverse data inputs such as project specifications, quantities, historical cost data, and historic data, which often exist in varied formats and structures. This lack of standardization and automation forces estimators to rely on manual, intuition-based, and subjective approaches, making cost estimation slow, inconsistent, and error-prone. LLMs have already demonstrated immense potential in extracting, synthesizing, and interpreting complex, heterogeneous data across various domains [9,10]. However, the construction industry still lacks a framework that integrates LLM in cost estimation process, preventing opportunity for AI-driven estimation solutions. This is a gap to limit the industry's ability to integrate AI to automate preconstruction cost planning. To address these challenges, this study evaluated existing LLMs performance and proposes a modular CoT prompting framework that improves LLM performance for cost estimation tasks. By structuring LLM interactions into task-specific modules, this framework allows estimators to ingest, analyze, and cross-reference data from multiple sources. This study provides general contractor companies with a flexible and scalable LLM-based framework, supporting their existing workflow automation while improving productivity and accuracy in construction cost estimation.

1.2. Problem Statement

Construction cost estimation is highly dependent on expert judgment and intuitive processes[11]. Although these dependencies have been practiced for decades, they remain time-consuming, highly subjective, and prone to human error [12]. Also, estimators often contend with multiple software platforms and data formats, adding inefficiency layers. The construction projects are time-sensitive in nature, and even minor delays in producing estimates negatively impact overall schedules and budgets. A major challenge arises in quantity takeoffs, where methods can differ substantially between 2D and 3D models. Estimators often face inconsistent naming conventions, document formats, and file structures, which increase error risks. In current industry practice, historical cost data is underutilized, and estimators generally rely on personal judgment instead of centralized databases, slowing the estimation process and reducing accuracy [13]. Without a systematic data integration approach, aligning estimates with benchmarks is difficult, and hampering competitiveness. When internal data falls short, estimators manually search external cost databases like RSMeans or Sage, extending timelines. The verification of project specifications requires manual

 doi:10.20944/preprints202510.1060.v1

cross-referencing, while Work Breakdown Structure (WBS) variations add complexity, delaying conceptual estimates and increasing errors. In addition to this, the subcontractors' estimate evaluation is a time intensive process as it requires manual verification for completeness, unit rates, and template consistency. The selection often relies on subjective factors like past relationships rather than standardized criteria. The compilation of bids and preparing final estimates take weeks, with additional delays. The lack of standard version control practice makes it difficult to track changes or maintain a history of costs, which often leads to repeated rework. These estimation challenges highlight the lack of consistency in cost estimation practice, which can result in overall process inefficiencies, which ultimately lead to delays and budget overruns [14]. The rise of LLMs presents an opportunity to automate processes, integrate data more effectively, and improve accuracy, offering a scalable solution to longstanding industry inefficiencies. Generative Pre-trained Transformers (GPT), among the most prominent LLM architectures, demonstrate advanced natural language processing and content generation capabilities. While initial studies have applied LLMs to various construction activities, there is no investigation on how existing pre-trained models perform under zero-shot conditions in cost estimation or whether their outputs can be improved through prompt engineering. A significant gap remains in providing industry professionals with a clear, evidence-based framework for integrating this generative AI technology into current cost estimation practices.

1.3. Research Main Objective

To investigate the detailed tasks associated with identified construction cost estimation burdens, evaluate the ability of state-of-the-art general-purpose LLMs to perform these tasks in a zero-shot learning setting, and explore the potential of modular CoT prompting to enhance their performance in cost estimation workflows. To achieve the main objective, the research is structured around several guiding Research Questions (RQs):

1.4. Research Questions

- RQ1: Can the state-of-the-art general-purpose LLMs perform construction cost estimation workflow tasks in zero-shot learning, without additional instructions and data?

The objective of RQ1 is to investigate the zero-shot generalization of existing LLMs on construction cost estimation workflow tasks in the pre-construction phase.

- RQ2: Can we change the behavior of LLM to better perform construction cost estimation tasks with modular CoT prompting?

The objective of RQ2 is to investigate the change in the performance of LLM with modular CoT, with additional instructions and examples, for construction cost estimation tasks.

1.5. Contributions

This study makes noteworthy contributions across theoretical, practical, and methodological dimensions. From a theoretical perspective, it explores how pre-trained LLMs can be leveraged for construction cost estimation tasks, presenting innovative ways in which careful prompt engineering can guide these models to produce targeted outcomes. The developed scenario, detailed tasks, instructions, and CoT prompting framework serve as a practical guide for estimators, offering the flexibility to be adapted as needed. On the methodological front, the study proposes an LLM-integrated modular framework aimed at standardizing and scaling the cost estimation process. This framework is designed to reduce the reliance on manual efforts through task automation, thereby making estimation processes faster and more accurate compared to traditional methods. This chapter is organized as follows: Section 2 reviews the relevant literature, and Section 3 outlines the study's methodology. Section 4 details the case study, followed by results and discussions in Section 5.

 doi:10.20944/preprints202510.1060.v1

Section 6 addresses the study's limitations and future research directions, and Section 7 summarizes the conclusion.

2. Literature Review

2.1. Existing Cost Estimation Approaches in Commercial Construction

Construction cost estimation methods are broadly categorized into four general groups, which vary in detail, accuracy, and pertinence relative to the project development stage. The rough order of magnitude (RoM) estimates is the most preliminary form of cost estimate that relies on past cost information and analogous estimating techniques to provide a general cost estimate. These are employed when project details are still uncertain and are utilized to perform a crucial function in initial feasibility estimates, with an accuracy margin of around ±25%. Square Footage Estimating expands on this method by employing cost-per-square-foot benchmarks derived through similar completed projects. While more detailed than RoM estimates, this is nonetheless a high-order estimate that offers cost insight into labor, materials, and professional services without selection of individual materials. Typically, estimates of square meterage are ±20% precise. Assemblies Estimating presents a more sophisticated level with its methodology for separating cost information about around specific building systems and assemblies- lowly formed components such as plumbing devices all the way through to higher-level mechanical installations[4,7]. This activity utilizes classification methods such as Uniformat II, with cost estimations up to ±15% precision[15]. Lastly, Unit Cost Estimating is the most detailed and precise cost estimating method, whereby a project is broken down to its lowest quantifiable elements, factoring in material needs, labor, and equipment costs. The practice follows the MasterFormat system of classification, to be able to supply detailed cost reports, and to achieve an accuracy rate of between -5% and +10%. Although unit cost estimating is very time- and data-consuming, its precision makes it the favored method at the final design or bidding phase[16]. The level of an estimation method employed should be guided by the level of a project's completion, availability of data, and cost certainty desired so that stakeholders are able to make proper financial decisions at all stages of development.

Construction cost estimation is an important and considerably complex activity that demands accuracy, speed, and flexibility at different phases of a project. However, there are a number of challenges to its effectiveness, including time-consuming procedures, lack of consistency in data, manual errors, and integration of diverse sources of cost information[17]. These difficulties are added to by the need to aggregate quantities from diverse formats (2D drawings and 3D models), refer to past data and outside pricing databases, cross-check estimates with project specifications, and review subcontractor bids. Lack of standardization and automation often results in duplication of effort, omissions, and unmatched units of measure, which can equate to large cost variances and budget issues [18]. The majority of estimators still rely on spreadsheets and manual input, which renders the process slower and more error-prone[19]. For example, on large commercial projects, estimators must extract cost data from architectural, structural, and mechanical drawings, input quantities into cost estimating software, and apply labor and material costs from external sources. If there is a change or revision in the design, these calculations must be redone, delaying project timelines significantly. While electronic estimating programs such as Sage Estimating and RSMeans offer automation, a majority of companies continue using outdated methods that are inefficient[20]. Moreover, project teams tend to recreate similar estimates time and time again for the same kind of projects without recycling historical data, thereby producing inefficiencies that could otherwise be minimized with the automatic recycling of data. Another critical issue is the integration of quantities from different sources, including 2D and 3D models. Modern construction projects involve various formats of cost estimation, and estimators need to extract quantities from traditional 2D drawings as well as BIM (Building Information Modeling) models[21]. The issue arises when the two sources do not align due to differences in software outputs or levels of detail. For instance, a structural estimator may pick up rebar needs from a 2D drawing through manual measurement, while a BIM model provides a

 doi:10.20944/preprints202510.1060.v1

volumetric calculation automatically for the same product. Without cross-checking these figures, material quantities that are non-matching may lead to extreme cost fluctuations. Additionally, certain aspects of the project, such as complex mechanical and electrical systems, are more precisely measured with 3D models, whereas simpler items such as finishes and flooring can continue to utilize traditional 2D takeoffs[22]. This hybrid approach generates inconsistencies that require additional verification and manual adjustments, increasing the work of estimators. Also, the inconsistency in measurement units adds to challenges in the cost estimation process. Suppliers, contractors, and project stakeholders use different measurement units, which call for conversions that may be prone to error. A vendor might quote concrete in cubic meters, while other quotes in cubic yards, demanding accurate conversions to prevent miscalculations. Similarly, flooring material can be estimated per square foot by one subcontractor but per square meter by another, bringing confusion if uncorrected. Additionally, there are differences in classification systems for different estimation methodologies; assemblies' estimation is based on Uniformat II, which structures costs by functional systems, and unit cost estimating is based on MasterFormat, structuring costs by construction divisions. To keep these formats consistent demands precise adjustments, particularly when rolling up subcontractor bids that may have different classification systems.

Another significant challenge is referencing cost data from different sources, including enterprise historical databases and external pricing guides such as RSMeans and Sage[20]. While historical project data are a good benchmark for cost estimation, inflation, market fluctuations, and regional differences in labor costs must be considered to maintain accuracy[23]. For instance, if a contractor applies two-year-old project material prices without adjusting for inflation, the estimate can be significantly lower than the current market rate. RSMeans also provides national average cost data, which may vary from local price fluctuations. A contractor operating in Los Angeles, where labor and material costs are higher than the national average, must employ location-based cost multipliers to avoid underestimation. Yet, it is still a difficult and time-consuming process to guarantee that all the data referenced meet project-specific conditions, particularly when manually comparing multiple sources. The cross-verification against project specifications is also a problem since the estimators must check their assumptions of cost against detailed material and construction method requirements[24]. Errors occur when the estimators assume standard materials without confirming that they match the project's technical specifications. For example, when the estimate explicitly states ordinary drywall installation, but the job requires a fire-rated gypsum board of a specific thickness, the estimate will prove to be deceptive, which leads to surprise cost escalation at procurement. Similarly, the mechanical and electrical components typically contain performance specifications that impact cost, such as energy efficiency ratings or code compliance specifications. Non-verification of these elements with the cost data will result in big budget mistakes and even potential delays in the projects.

2.2. Application of Generative Pre-Trained LLMs in Construction Cost Estimation

In recent years, substantial progress in artificial intelligence research has facilitated the development of powerful Large Language Models (LLMs), such as GPT, Gemini, and Llama. These advancements are a subset of generative AI (GenAI), which relies on deep learning architectures- particularly neural networks- to handle both labeled and unlabeled data through supervised, unsupervised, and semi-supervised techniques[9,25]. LLMs are engineered to comprehend and produce human-like text by analyzing extensive datasets and leveraging patterns derived from large-scale pre-training[26,27]. Once a user provides a prompt, these generative systems employ learned representations to create new, contextually appropriate content. This process is driven by transformer-based architectures that utilize encoders to interpret inputs and decoders to formulate coherent, relevant responses (Ghimire et al., 2024).

A variety of GenAI models exist, typically categorized by the type of output they generate. Four prominent examples are text-to-text, text-to-image, text-to-video/3D, and text-to-task models. Text-to-text models learn mappings between pairs of textual data; given natural language inputs, they

 doi:10.20944/preprints202510.1060.v1

generate linguistic outputs aligned with user queries[28]. In contrast, text-to-image models- often employing diffusion-based methods- are trained on image datasets associated with textual captions, enabling them to produce images in response to textual prompts [29] . Similarly, text-to-video models stitch together video pieces from text inputs, ranging from brief descriptions to extensive scripts[30]. Text-to-3D models also work in the same manner, depending on text inputs to create three-dimensional objects corresponding to user needs. Text-to-task models, on the other hand, focus on accomplishing specific tasks described in textual commands, including responding to questions, conducting searches, predicting, and executing other requested operations [31] . Although all of these generative approaches may be utilized across various tasks, model choice often relies on the pursued application. Large-scale pre-trained models such as GPT are designed with versatility in mind, capable of fine-tuning for a wide array of purposes ranging from question answering and sentiment analysis to image captioning and instruction following. Based on their generative mechanisms, GenAI models can also be classified into at least five major types as shown in Figure 1 (GANs, [9,30,32] .

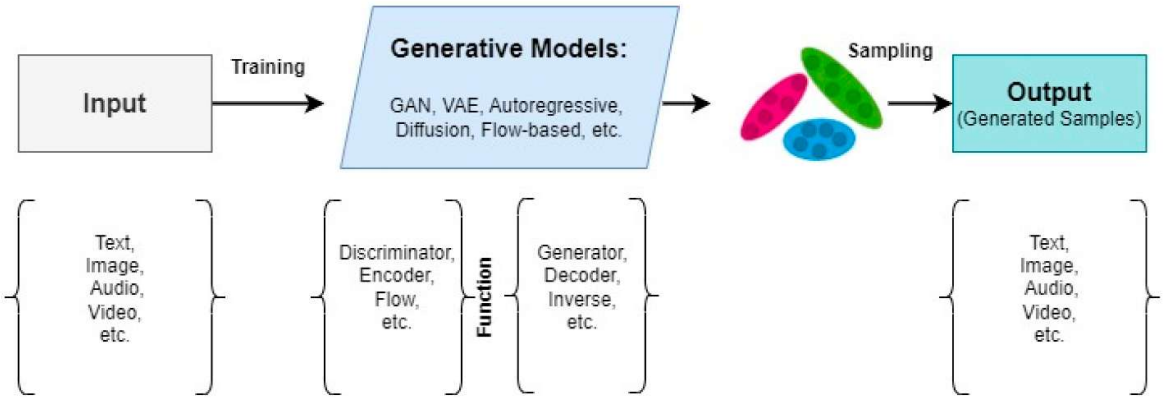


Figure 1. GenAI models.

Generative Adversarial Networks (GANs; Goodfellow et al., 2014) are commonly employed for high-fidelity image generation, leveraging a competitive framework in which a generator strives to create realistic images while a discriminator attempts to distinguish them from real examples. In contrast, Variational AutoEncoders Generative Adversarial Networks [33] are commonly employed for high-fidelity image generation, leveraging a competitive framework in which a generator strives to create realistic images while a discriminator attempts to distinguish them from real examples. In contrast, Variational AutoEncoders [34] are frequently utilized for text generation because they learn a latent representation of the underlying data distribution. This latent-space modeling helps produce coherent, grammatically sound outputs. Autoregressive models- exemplified by architectures such as GPT (Radford et al., 2019)- excel at generating text that closely resembles their training data, as they process one token at a time while conditioning each subsequent token on previously generated content. Diffusion models [36] take a different approach by starting with noise and iteratively reversing a diffusion process, thus crafting smooth and natural outputs, particularly for image synthesis. Finally, flow-based models [37,38] employ invertible transformations that map data to latent variables and back again, facilitating versatile and creative image generation through precise control over the generative process. Altogether, these methods illustrate the variety of current generative modeling techniques, each bringing novel strengths for uses ranging from photorealistic image generation to text coherence.

Within the construction sector, emerging exploratory studies highlight the potential of LLMs across various project stages- including feasibility evaluation, design, procurement, on-site execution, and operations/maintenance- by capitalizing on zero-shot learning, few-shot learning, chain-of-

 doi:10.20944/preprints202510.1060.v1

thought reasoning, retrieval augmented generation, and fine-tuning [9,10,39]. For example, researchers have proposed frameworks that use prompts to extract detailed information from complex BIM models [27], integrated GPT-based approaches for material selection through few-shot or zero-shot prompting [40], and experimented with ChatGPT-enabled project scheduling [41]. Other applications include identifying potential safety risks via a BERT sentence-pair model tuned on the OSHA database [42] and employing LLMs to plan tasks for construction robots [43,44]. Despite these advancements, only a few published works directly address cost-related tasks. For instance, [45] examined budget forecasting and bill of quantities generation using LLMs, while [46] evaluated a prompt-driven framework employing the Mistral-7b language model to query cost-oriented data in IFC format. However, to the best of the authors' knowledge, no comprehensive studies have yet explored how pre-trained LLMs might be harnessed to tackle construction estimation in real-world scenarios or integrated seamlessly into existing workflows. Further, there is a lack of literature offering practical guidelines and frameworks on how to use prompt-based techniques efficiently for routine estimation tasks in industry practice. Filling this void is critical to translating LLMs' potential on paper into actual productivity gains and improved cost estimation practices for construction projects.

2.3. Chain of Thoughts (CoT) Prompting

The recent advances in large language models have given rise to several prompting techniques to improve output responses' quality, correctness, and relevance. Among the most widely used are zero-shot, few-shot, chain-of-thought (CoT), and tree-of-thought (ToT) prompting [47–49]. Zero-shot prompting with LLMs involves asking a model to perform a task without providing it with any examples in advance, and this helps effectively test the model's internal ability to generalize the output. The few-shot prompting, by contrast, provides the model with a few examples and then asks it to perform a similar task. The chain-of-thought prompting focuses on reducing a tricky problem to a step-by-step sequence of rational processes so that the model can tackle multi-step questions better. By contrast, tree-of-thought prompting hierarchically organizes task instructions so general goals are at higher levels and more detailed prompts at lower levels [47–49]. Chain-of-thought (CoT) prompting is a very sophisticated technique that simulates human thinking by breaking down complex tasks into middle-level steps [50]. Unlike simpler prompting methods that rely primarily on straightforward question-and-answer formats, CoT prompting systematically guides the model through a logical sequence, significantly improving its ability to tackle multi-step challenges such as mathematical reasoning, commonsense inference, and intricate decision-making [47,51] . And, these studies demonstrated that the large-scale language models often exhibit emergent reasoning capabilities when prompted with carefully curated CoT exemplars, surpassing the performance traditionally achieved through simpler prompt structures. The incremental, step-by-step nature of CoT, not only improves the accuracy of outputs but also guides models to generate responses that more closely follow human thinking processes[52]. A critical advantage of CoT prompting lies in its contributions to interpretability and transparency in AI-driven outputs. By making each step of the reasoning process explicit, users can more easily discern how final conclusions are reached, thereby addressing common "black-box" concerns in artificial intelligence [53]. Moreover, multiple studies have demonstrated that CoT prompting leads to marked performance improvements across a spectrum of complex tasks- ranging from mathematical problem-solving to programming and medical diagnostics [50,51]. Its adaptability is another key benefit, as CoT can be implemented across various domains requiring arithmetic, logic-based problem-solving, or multi-step workflows [49]. An additional strength is CoT's potential to reduce hallucinations in language models; by compelling AI to construct a clear chain of reasoning, spurious or illogical outputs become less likely.

Designing effective CoT prompts generally involves a systematic procedure to ensure logical coherence and completeness. The first task is to define the overarching goal and pinpoint the structural logic necessary for a correct solution [50]. Next, one must decompose the larger problem into smaller, interconnected steps, making sure each intermediate stage is explicitly laid out so the

 doi:10.20944/preprints202510.1060.v1

model does not have to infer these steps on its own [54]. Providing detailed exemplars is crucial here, as showcasing the correct reasoning flow helps the model internalize more accurate patterns[55].A recent study also suggests that multiple, carefully chosen examples in a single prompt boost generalization, enhancing both the accuracy and consistency of CoT-based solutions [56]. Finally, iterative refinement- guided by prompt engineering and feedback loops- enables domain-specific tuning that can further improve structured reasoning [56].

When implementing CoT prompting, users often begin by selecting an appropriately large language model, such as GPT-4, Llama, Claude, or any LLM, given that higher parameter counts generally correlate with more successful CoT outcomes[57]. After choosing the model, practitioners develop a tailored CoT framework that outlines step-by-step prompts relevant to their particular application domain. The mutiple testings and iterative updating are then employed to assure that the model consistently adheres to the prescribed logical structure [58]. Methods such as programmatic prompt generation and self-consistency sampling can be applied to enhance the reliability of CoT output such that models are able to provide multiple reasoning pathways and select the most coherent response [50,51]. By taking these implementation steps, businesses can leverage CoT prompting to significantly improve AI-based decision-making, reasoning, and problem-solving across various industries. Therefore, there is a high potential for CoT prompting in construction cost estimation due to tasks made up of numerous interdependent elements, including quantities, cost references, specifications, calculations, and formatting. By dividing each phase of the estimation into discrete, unequivocal steps, a CoT-based system closely mimics the logical process of a human estimator. This step-by-step process not only maintains accuracy and transparency of explanation but also generates a traceable record of all intermediate actions. As such, it addresses fundamental challenges in construction management by facilitating more reliable and auditable AI-driven cost estimation assistance, which improves the automation and application of CoT prompting in real-world industry applications.

3. Methodology

This study followed a structured methodology to develop and evaluate the application of LLMs in modular data extraction for construction cost estimation. The methodology, as shown in Figure 2, was divided into three sequential steps: (i) scenario development, (ii) zero-shot testing for the scenario with existing pre-trained LLMs, and (iii) an experiment applying modular CoT approach. The first step involved designing a conceptual estimation scenario to test the performance of existing LLMs in addressing the identified tasks. The detailed tasks for the scenario were based on the detailed burdens of current estimation process, tasks, and sub-tasks identified by our previous study, Ghimire, (2025). The detailed table is presented in Appendix 1. The experimental scenario was then used to test LLMs' existing capabilities, specifically their ability to learn and respond accurately without prior training (zero-shot learning). This step provided insights into the current strengths and limitations of LLMs in executing cost estimation tasks and identified areas requiring refinement or additional prompting strategies. The final step applied an experimental approach to evaluate the effectiveness of modular CoT prompting. This involved structuring LLM interactions using a systematic reasoning framework to enhance their ability to generate accurate, context-aware responses. The outcome of this experiment provided proof of concept for the implementation of modular sequential CoT prompting in construction cost estimation, demonstrating its potential to reduce estimation burdens and improve the existing process, ultimately enhancing efficiency. The findings from each step contributed to refining LLM-driven approaches, demonstrating their practical applicability in real-world cost estimation workflows within the construction industry. This study ultimately aimed to validate the potential of LLMs to support estimation process automation and improve efficiency in the construction sector.

 doi:10.20944/preprints202510.1060.v1

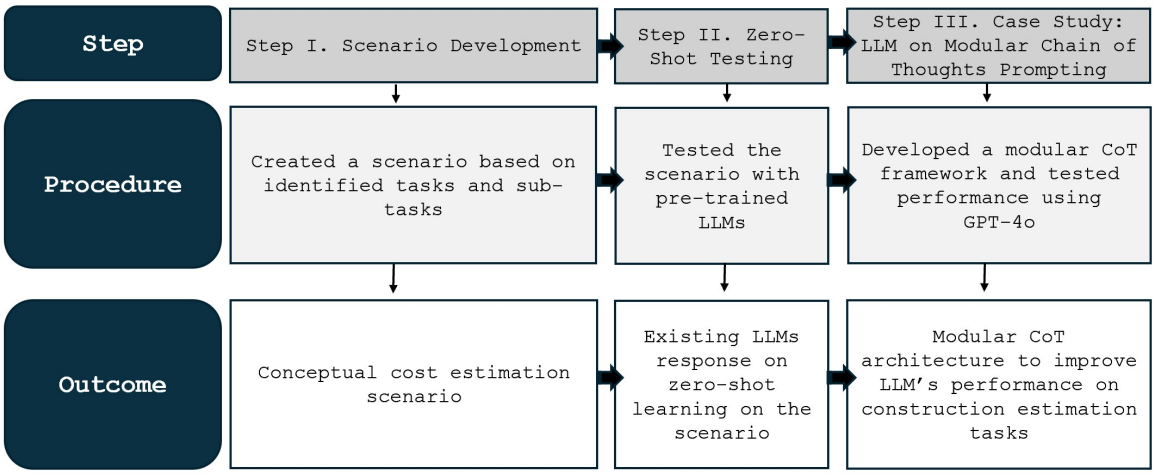


Figure 2. Research methodology.

3.1. Proposed LLM Framework

The proposed framework streamlines construction cost estimation process by integrating Large Language Models (LLMs) into a structured workflow, enhancing automation, accuracy, and adaptability. As illustrated in Figure 3, the proposed framework serves as an intelligent intermediary between various construction data sources, file types, and estimators, helping to reduce the burdens associated with manual estimation methods. Traditional cost estimation often involves labor-intensive data extraction, analysis, and interpretation, leading to inconsistencies and potential errors. By leveraging AI-driven automation, this framework processes vast and complex datasets with improved efficiency, ensuring standardization and precision in cost estimation. At the framework's core lies the LLM, which dynamically interacts with multiple data sources, including Building Information Modeling (BIM)-generated quantity takeoffs, multiple cost databases, estimation templates, LLM instructions, and project specifications. The multiple cost databases provide options to select preferred material and labor rates, while templates and specifications define consistent output and project-specific requirements. By aggregating and interpreting these diverse inputs, the system generates accurate estimates per requirements. The interactive capability allows estimators to engage directly with the LLM through prompts. Estimators can tune prompt parameters, set priorities, request detailed explanations, show tables, generate files, and request insights during the estimation process. The LLM automates not only cost-related calculations but also allows for interactive Q&A, context-aware recommendations, and instant adjustments. The two-way interaction is dynamic and adapts well in keeping the process transparent, enhancing confidence in decisions with reduced error possibilities. This is one of the strong points in the ease of adaptation to an ever-evolving landscape of LLM models. Because it's built with a modular prompt structure, this framework will include more advanced AI models as they emerge, enhancing reasoning, contextual understanding, and accuracy. Because generative AI gets better with time, its framework is open to adaptation due to model updates and fine-tuning. At the end of the system are structured outputs completed estimation tasks, subtasks, and visualized cost insights to make decisions easier and more efficient. It also minimizes manual efforts, expensive errors, and accelerates the process as a whole by embedding AI in an estimation workflow. In the end, this finally leads to a fast, dependable, and well-scalable cost estimation system that easily adapts to any future change in AI technology.

 doi:10.20944/preprints202510.1060.v1

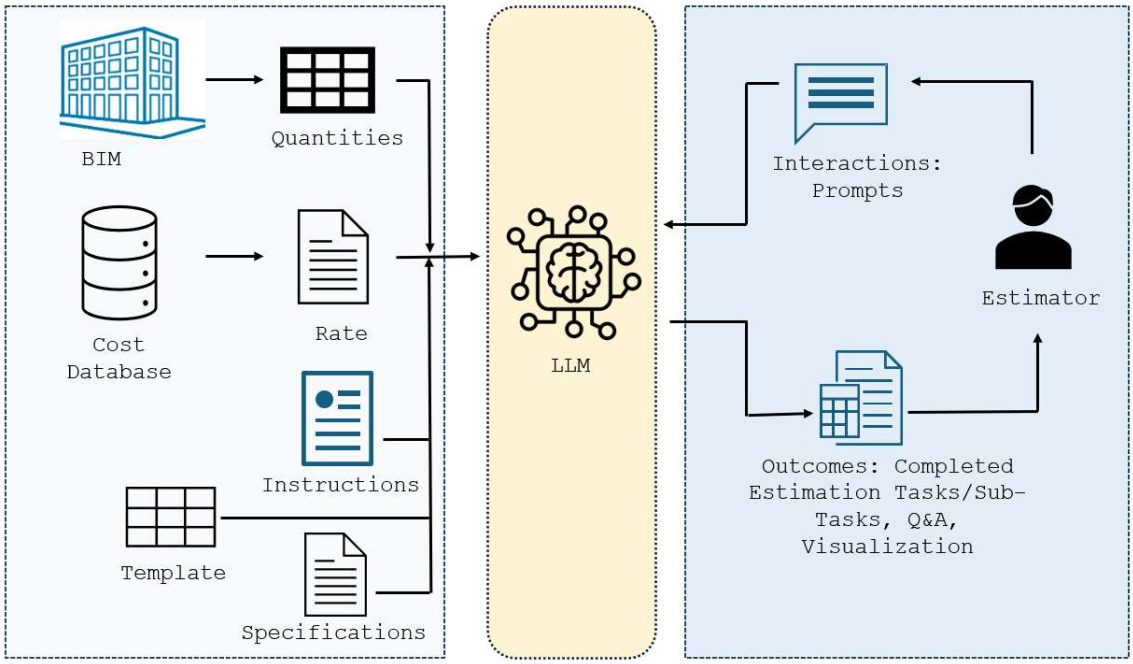


Figure 3. Proposed LLM integrated framework for construction cost estimation.

3.2. Cost Estimation Scenario for Existing LLMs

To evaluate the effectiveness of pre-trained LLMs in construction cost estimation, conceptual cost estimation scenario was developed, based on detailed tasks developed in previous study by Ghimire (2025). The previous study identified cost estimation burdens in three categories, each representing a key aspect of the construction estimation workflow: (1) Conceptual Estimation, (2) Evaluating Subcontractor Estimates, and (3) Change Management, Version Control, and Data Recycling. [59] then mapped burden-to-task in transforming abstract challenges into smaller tasks and processes that could be analyzed for automation potential. Following this, the study further decomposed all identified tasks into granular sub-tasks, refining the workflow into smaller, manageable components that LLMs could understand. This hierarchical breakdown- from broad burdens to actionable tasks and detailed sub-tasks- provided a structured foundation for integrating LLMs into cost estimation workflows. [59] utilized the construction industry insights through direct engagement with industry practitioners, ensured the identified tasks & sub-tasks were not only theoretically relevant but also practically applicable to real-world construction operations. This involved digging deeper to understand the mechanics behind each burden and identifying where processes break down, as well as the specific details that make certain tasks more cumbersome. In this study, we developed Scenario 1- conceptual estimation- focuses on initial cost estimation for general contractors, where estimators must work with limited project information to generate a reliable cost projection. This scenario includes tasks such as Aggregating Quantities, Referencing Enterprise Historic Cost, Referencing External Cost Databases, and Cross-Verification with Project Specifications. Figure 4 illustrates how each task and sub-tasks are mapped to specific user queries and expected LLM responses (Table 1). The structured sequential dialogue between the user and LLM simulates realistic estimator interactions, ensuring that the evaluation measures logical consistency, task interconnections, and accuracy of AI-generated responses.

 doi:10.20944/preprints202510.1060.v1

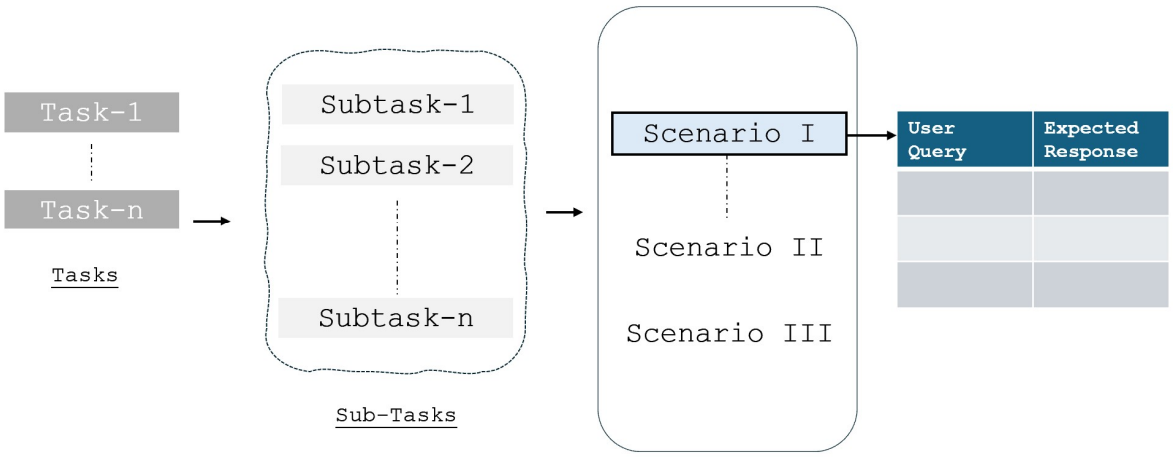


Figure 4. Scenario creating steps.

To evaluate the LLMs' performance, this study tested Scenario 1 using zero-shot prompting and then implemented a Chain-of-Thought (CoT) framework to assess performance improvements. By developing task-driven scenarios for evaluation, this study replicates real-world estimation processes, allowing for analysis of existing pre-trained LLMs' effectiveness in construction cost estimation.

Table 1. Scenario 1- Conceptual Cost Estimation Scenario.

ID	*Estimation Sub-Task*	*Query from the "User"*	*Expected Answer from "LLM"*
1	Task Initiation	I have a new project for which I need to create an estimate. Can you help me?	*Yes, I can. Could you provide some details about the project, such as the type of construction, location of the project, and any specific type of estimation- Assembly or Item based- that you want to follow?*
2	Specify Estimation Type	Yes. This is a new commercial building construction in Omaha, Nebraska. I want to do an Assembly cost estimation.	*Ok. For assembly estimation, you need to know the assembly breakdown in Uniformat II before starting the estimation. Do you know this?*
3	Confirm Work Breakdown Structure (WBS)	No, can you explain the Assembly breakdown in Uniformat II down structure for me?	*Of course! Uniformat II divides the building into major functional areas or "assemblies" based on the work they perform.* *Here's an overview of the Uniformat II structure, broken down into its primary categories:* *A. Substructure*

 doi:10.20944/preprints202510.1060.v1

This section includes all the elements related to the building's foundation and below-grade construction:

- *A10 Foundations: Includes standard footings, foundation walls, slab-on-grade, and pile systems.*
- *A20 Basement Construction: Covers items like basement excavation, retaining walls, and waterproofing.*

B. Shell

The building's exterior envelope or shell:

- *B10 Superstructure: Includes vertical elements (columns, beams) and floor/roof framing systems.*
- *B20 Exterior Enclosure: Covers walls, windows, doors, and exterior cladding.*
- *B30 Roofing: Includes roofing materials, insulation, and any associated components like flashing and drainage systems.*

C. Interiors

This section covers everything inside the building:

- *C10 Interior Construction: Partitions, interior walls, and doors.*
- *C20 Stairs: Includes staircases, railings, and related structures.*
- *C30 Interior Finishes: Flooring, ceilings, paint, and other wall finishes.*

D. Services

Systems that provide utilities and services to the building:

 doi:10.20944/preprints202510.1060.v1

- *D10 Conveying Systems: Elevators, escalators, and lifts.*
- *D20 Plumbing: Water supply, drainage, and fixtures.*
- *D30 HVAC: Heating, ventilation, and air conditioning systems.*
- *D40 Fire Protection: Sprinkler systems, fire alarms, and smoke detectors.*
- *D50 Electrical: Lighting, power distribution, and wiring.*

E. Equipment & Furnishings

This category covers the built-in equipment and furnishings in the building:

- *E10 Equipment: Major fixed equipment such as kitchen appliances or medical equipment in a hospital.*
- *E20 Furnishings: Built-in furniture and seating systems.*

F. Special Construction & Demolition

Non-standard or specialized construction:

- *F10 Special Construction: Includes items like greenhouses, clean rooms, or pools.*
- *F20 Selective Demolition: Any necessary demolition work.*

G. Building Sitework

Anything outside the building footprint:

- *G10 Site Preparation: Grading, clearing, and demolition of structures.*
- *G20 Site Improvements: Pavements, curbs, gutters, and landscaping.*
- *G30 Site Mechanical Utilities: Storm drainage, water distribution, and gas supply.*

 doi:10.20944/preprints202510.1060.v1

			• *G40 Site Electrical Utilities: Exterior lighting, power distribution, and communications.*
4	Collect Quantities	Ok, understand. Now how can I start?	*Now, you need to provide quantities, cost data, project specifications, and a particular format or template for the final output of the estimate.*
5	Collect Quantities	OK. let's go one by one with the data	*Great. Provide me quantities.*
6	Determine Scope	OK. I only have quantities for the shell category. And want to estimate only that portion.	*I got it! I will focus on the Shell (B) category for now. Please provide the quantities.*
7	Summarize and Display Collected Quantities	Here's the quantity excel file I have uploaded. Please show me a list of items and quantities in tabular format.	*Ok. Here is the list of your items and respective quantities. Do you want to know more about, or do you want to move ahead to the next steps?*
8	Check Duplications	Can you check if there are any duplicate items based on the description and assembly code?	*These are items that have exactly the same description and assembly code. Do you want to add them to a single item?*
9	Check Duplications	Yes, please add. Also, can you find potential duplication that doesn't match exactly but potentially?	*Here are these group items that look potentially similar based on size and descriptions.*
10	Finalize Quantities	No, they are not the same items. Keep separate.	*Ok. We have now total 25 items. What do you want to do next with quantities? Should we move to cost data?*
11	Format Quantities	Now, I want you to help formatting before starting estimate. Can you do that?	*Yes, I can. Do you have any template of format or you want me to suggest some format?*
12	Suggest Template	Please suggest a template format suitable for this estimation	*Here is the suggested format suitable for the assembly estimate… Do you want to use this? Or do you want me to create other templates?*

 doi:10.20944/preprints202510.1060.v1

13	Upload Template and Map Quantities	I have one standard template format for quantities. Here I have uploaded. Please put quantities into this new format and follow this instructions to do mapping- i)Read Instructions inside Template file, each description under every title is the description for you understand what it means ii) Look at example output and understand iii) map all quantities to suitable section of this template's yellow highlighted area, and you are free to add remove rows as required for items. Show the table and save.	*I have arranged the original quantities into the new template that you provided as per instructions i, ii, and iii.* *See the table below. Let me know if you want to modify or change anything.*
14	Provide download link	Also, can you export new table in excel format.	*You can click on the link below to download the new quantities table.*
15	Referencing Multiple Cost Data Sources	Now, we are good for quantities. What's our next task?	*We will work on cost data in the next step. Please provide me with cost data.*
16	Collect Cost Data	Ok. I have uploaded two datasets for cost data that I want to use. 1) Historic cost-enterprise database 2) External cost database	*Great! I have stored both datasets. How do you want to set up the estimation to use both your historic cost and external cost?*

 doi:10.20944/preprints202510.1060.v1

17	Define Priority and calculations	I want to use the historical database as much as possible.	*Ok. I can structure the estimation to prioritize historic cost and only use the database cost if the historic cost for an item is not available.*
18	Define Priority and calculations	In the historic cost data add a column and get the average of all available commercial projects, for all cost items	*Ok. The new column is added to historic dataset, I have saved the updated data. I have calculated the average for items in past commercial projects Here is the table:*
19	Set Conditions	I want to use historic cost for all items if it is available. If not , then i want to use database cost.	*Ok. I can structure the estimation to prioritize historic cost and only use the database cost if the historic cost for an item is not available.*
20	Set Common Identifier	How would you prioritize?	*Are there any common identifiers in two cost data sets? On what basis you want to find items?*
21	Set Common Identifier	We have the same Assembly Code but slight variations in descriptions. So, consider Assembly Code as the common Identifier.	*I will first load both the historic and database cost data from separate files. Let's assume both files contain Assembly Code as a common identifier that can be used to match items.* *I first check the historic cost data. If a historic cost is available, I use that.* *If no historic cost is found, I search for the cost in the database cost file.*
22	Handle Missing Items	If an item is missing from both datasets, what would you do?	*I will ask you with closest options based on description.*
23	Summarize Work Done	Briefly Summarize in bullet points: what you have done so far, what data you have, what data you need now	*Summary of What Has Been Done:* • *loaded quantities, formatted to match provided quantity template* • *loaded historic and added average cost column for all items* • *loaded database cost.* *Data I Have:* • *Quantities, Historic cost data (with average commercial costs), Database cost*

 doi:10.20944/preprints202510.1060.v1

			Data I Need: • *Project Specification* • *Final output format or template*
24	Explain Next Steps	What's the next steps?	*Next Steps:* 1. *Reading output template* 2. *Calculate total costs using provided quantities and cost data sets* 3. *Verify with specification* 4. *Export the final estimate*
25	Output Template Recall	Here, I have uploaded the output template file and specification file. Please read and store both files.	*Thank you! Now that I have the output template and project specification files. Can I go ahead start cost calculations?*
26	Define Priority for External Cost & Define Condition to use External Cost Data	How would you utilize the template and start calculation? Explain to me before doing it.	*First, I'll carefully review the output template to understand its structure- column names and section layout. There may be sections based on Uniformat II categories (e.g., B10 Superstructure, B20 Exterior Enclosure).* *Second, I will map the quantities (description, unit, and quantity) to the output template.* *Third, based on the description of quantities, I will search for each item to match in cost data, first to 'historical cost', where available, and then to the 'database cost' if needed.* *Fourth, multiply the quantity by the unit cost to calculate the total cost* *Fifth, I will keep the cost data assembly code as the unique identifier in the 'Assembly Code' column*
27	Handling No Match Items	What if you don't find an exact match?	*If I don't find each match in the cost data, I will ask you to select with the closest options (maximum 5 options).*

 doi:10.20944/preprints202510.1060.v1

28	Handling Multiple Matched Items	What if you find multiple matches?	*If I find multiple matching items in the cost data, I will provide you with options and ask you to select one out of those options.*
29	Follow Output Format	Can you export the estimate in excel format with the file name 'conceptual estimation _today's date)?	*Yes, I can export the estimate with the file name "conceptual estimation_[today's date]."*
30	Cross-Verification with Specification	Ok. If I want to verify anything with specifications, how would you help?	*I can help you in multiple ways:* *I can provide any specific information that you want to look at.* *I can also provide a summary of any discrepancies in item or installation description, between specification and items in estimation, that need clarification.* *And, If adjustments are required, I can incorporate them into the estimate.*
31	Handle Changes	If something needs to be modified, added, removed, or corrected, would you do it for me?	*Absolutely! If any modifications, additions, removals, or corrections are needed, just let me know the specific details, and I'll make the necessary changes for you. Whether it's adjusting costs, updating quantities, or correcting formatting, I'm happy to assist with all aspects of the estimate.*
32	Work Resuming	If I stop here, and come back to continue work, could you remind me with summary of what we have done, and what's next step, and providing me the last file?	*Yes. When you come back, I'll provide a quick summary of the work we've completed so far, including details like the data we've loaded, the calculations performed, and any adjustments made. I'll outline the next steps based on where we left off, so you can pick up exactly where you need to. I'll also share the last file we were working on.*

3.3. Cost Estimation Scenario for Existing LLMs

To evaluate the performance of pre-trained LLMs in construction cost estimation tasks, using Scenario 1, this study employed three widely used natural language (NL) evaluation metrics- BLEU, ROUGE-L, and METEOR- and compared four leading models available as of January 2025: GPT-4o, LLaMA 3.2, Gemini 2.0, and Claude 3.5 Sonnet. The selection of GPT-4o, LLaMA 3.2, Gemini 2.0, and Claude 3.5 Sonnet was based on their state-of-the-art architecture and strong performance across

 doi:10.20944/preprints202510.1060.v1

various benchmarks. GPT-4o, developed by OpenAI, is recognized for its superior multimodal reasoning and domain adaptation capabilities [60]. LLaMA 3.2, an open-source model from Meta, is optimized for efficient fine-tuning and domain-specific applications [61]. Gemini 2.0 from Google DeepMind leverages retrieval-augmented generation (RAG) for complex knowledge-intensive tasks, while Claude 3.5 Sonnet is designed for long-context reasoning and structured interpretability [62,63]. These models were selected due to their prominence in AI research and their demonstrated capabilities in technical reasoning, retrieval-augmented generation, and structured output generation. BLEU, originally developed for machine translation, measures n-gram precision between generated and reference texts, making it suitable for assessing word-level accuracy [64]. However, BLEU has been criticized for not considering semantic meaning or fluency, which is why recent research integrates additional contextual similarity measures alongside it [65,66]. To address BLEU's limitations, ROUGE-L was incorporated to evaluate phrase-level recall and longest common subsequence (LCS) matches, making it particularly relevant for cost estimation, where key terms and structured phrases must align with expected outputs [67,68]. Additionally, METEOR was used, as it improves upon BLEU by incorporating synonym matching, stemming, and precision-recall balancing, offering a more linguistically robust assessment of generated text [69]. The application of several metrics ensures comprehensive evaluation, capturing both lexical accuracy and semantic accuracy, which is essential for AI-generated responses in structured domains like construction cost estimation. The results , as shown in Table 2, indicate that GPT-4o performed better than the other models in all metrics across the board, with a BLEU of 0.023, a ROUGE-L of 0.185, and a METEOR of 0.196, indicating improved performance in word correctness, structural coherence, and semantic accuracy.

Table 2. LLMs' Performance on Estimation Tasks with Zero- Shot Learning.

Model	BLEU	ROUGE-L	METEOR
GPT4o	0.023*	0.185*	0.196*
Llama 3.2	0.0126	0.112	0.157
Gemini 2.0	0.010	0.095	0.122
Claude 3.5 Sonnet	0.0135	0.170	0.168

In order to assess how comprehensive, accurate, and uniform GPT-4o-generated answers are in terms of tasks related to construction cost estimation, evaluation was undertaken by human means. Compared to metrics based evaluation measures, human evaluation includes more detailed information regarding contextual appropriateness and logical cohesion for LLM-written answers such that the domain-based processes in the model can be adequately traced [50,70]. This study conducted the qualitative evaluation, with confidence scores- of 1,2, and 3, reflecting the quality of the response and how close responses are to the expected output outlined in the given scenario. A score of 1 (low confidence) was assigned if the response was incomplete or inaccurate. A score of 2 (medium confidence) reflected a partially correct response but lacked full accuracy or completeness. In addition, a score of 3 (high confidence) was given when the response was fully accurate, complete, and aligned with the expected response. The completeness criteria for human evaluation are shown in Table 3.

Table 3. Human evaluation criteria.

Task ID	Estimation Sub-Task	Completeness Criteria for LLM Response	User Confidence Level on Response

 doi:10.20944/preprints202510.1060.v1

			(1=Low 2=Medium 3= High)
1	Task Initiation	▪ *Readiness to help* ▪ *Asking for project details* ▪ *Asking for Estimation Type*	3
2	Specify Estimation Type	▪ *Providing workflow or process understanding of the estimation process for the specified specification type* ▪ *Confirm with the user if the user knows the Assembly estimation.*	2
3	Confirm Work Breakdown Structure (WBS)	▪ *Providing Work Breakdown Structure (WBS)*	3
4	Collect Quantities	▪ *Asking for required information- quantities, project specifications* ▪ *Asking for the final outcome and template format*	1
5	Collect Quantities	▪ *Asking for quantities first*	1
6	Determine Scope	▪ *Knowing the scope for quantities* ▪ *Asking again for the dataset*	3
7	Summarize and Display Collected Quantities	▪ *Providing a list of items and their summary* ▪ *Confirming before moving to the next step*	3
8	Check Duplications	▪ *Looking at the exact match items* ▪ *Asking criteria matched items*	1

 doi:10.20944/preprints202510.1060.v1

9	Check Duplications	▪ *Presenting, if found, the potential duplications list*	1
10	Finalize Quantities	▪ *Informing about total items* ▪ *Asking what's next with quantities now* ▪ *Follow up if the user wants to move to work with the next dataset*	1
11	Format Quantities	▪ *Asking for a template* ▪ *Confirming if users don't have a template*	3
12	Suggest Template	▪ *Suggesting formatting* ▪ *Confirming with the user*	3
13	Upload Template and Map Quantities	▪ *Mapping out quantities in the provided template* ▪ *Showing table* ▪ *Confirming if modifications are needed* ▪ *Saving table*	2
14	Provide download link	▪ *Providing the link to download*	2
15	Referencing Multiple Cost Data Sources	▪ *Informing the next step* ▪ *Asking for cost data for the next step*	3

 doi:10.20944/preprints202510.1060.v1

16	Collect Cost Data	▪ *Clarifying with users how they want to use multiple cost datasets*	2
17	Define Priority and calculations	▪ *Understanding user's priority order* ▪ *Understanding user's calculation rule*	3
18	Define Priority and calculations	▪ *Executing priority-based calculations*	2
19	Set Conditions	▪ *Structuring the estimation reference with historic cost as the first priority for referencing*	2
20	Set Common Identifier	• *Asking the user about a common identifier in different data sets* • *Asking the user on what basis they want to find the cost items*	2
21	Set Common Identifier	▪ *Explaining to the user step-by-step on how the system prioritizes and use the historic and other cost database*	1
22	Handle Missing Items	▪ *Informing the user about handling the missing items* ▪ *Providing the user with the closest options in the cost databases*	1
23	Summarize Work Done	▪ *Presenting summary of work done so far in bullet points* ▪ *Presenting separate sections for what has been done, data system has, data system need*	1
24	Explain Next Steps	▪ *Explaining next steps*	1
25	Output Template Recall	▪ *Confirming that the remaining files or data from the user have been received* ▪ *Recalling estimation template*	2

 doi:10.20944/preprints202510.1060.v1

		▪ *Confirming for the next step*	
26	Define Priority for External Cost & Define Condition to use External Cost Data	▪ *Understanding output template structure and section layout for Uniformat II mapping* ▪ *Mapping quantities to output template* ▪ *Searching items in the cost database, with historic cost data as the first priority* ▪ *Calculating total cost by multiplying unit cost with quantity* ▪ *Asking user if they wants to incorporate the inflation percentage in calculation* ▪ *Keeping the cost database's assembly code in the final output*	2
27	Handling No Match Items	▪ *Providing user with closest options in case of no exact match*	1
28	Handling Multiple Matched Items	▪ *Providing the user with a matched items list and asking to select suitable cost item*	1
29	Follow Output Format	▪ *Executing file export in the provided format and naming convention*	1
30	Cross-Verification with Specification	▪ *Providing specific information from the specification that the user wants to look at* ▪ *Providing a summary of discrepancies for any, some, or all items that need clarification* ▪ *Incorporating adjustments to estimate if requested*	2
31	Handle Changes	▪ *Explaining how the system handles modifications- adding, removing, and*	3

 doi:10.20944/preprints202510.1060.v1

		changing- for cost, quantities, and even formatting.	
32	Work Resuming	▪ *Providing a summary of past work,* ▪ *Providing past data,* ▪ *Outlining the next steps based on the point the work stopped last time*	2

As shown in Figure 5 & 6, the human evaluation highlighted notable deficiencies in zero-shot prompting. The figure compares an expected structured response against GPT-4o's actual zero-shot output. The expected response follows a logical approach, prioritizing historical cost data before falling back on database cost information. However, the LLM's zero-shot response lacks predefined priority logic, failing to specify how costs should be matched when multiple sources are available. Such shortcomings reduce the reliability of outcomes in cost estimation workflow, indicating the necessity of structured prompting strategies like Chain-of-Thought (CoT)[47,50]. The human evaluation shows an average confidence score of 1.906 (64%), indicating that responses frequently lacked completeness and correctness. This aligns with a previous study that emphasizes the limitations of LLMs in domain-specific reasoning without explicit reasoning prompts [71].

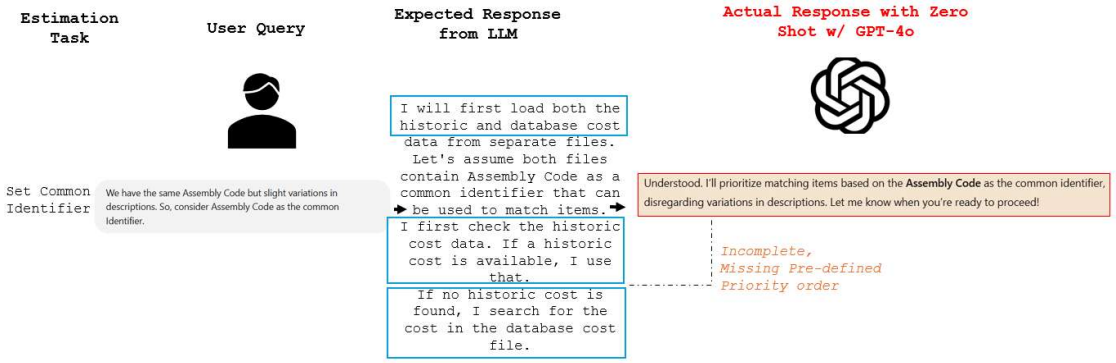


Figure 5. Example 1 of expected vs. actual response in zero-shot.

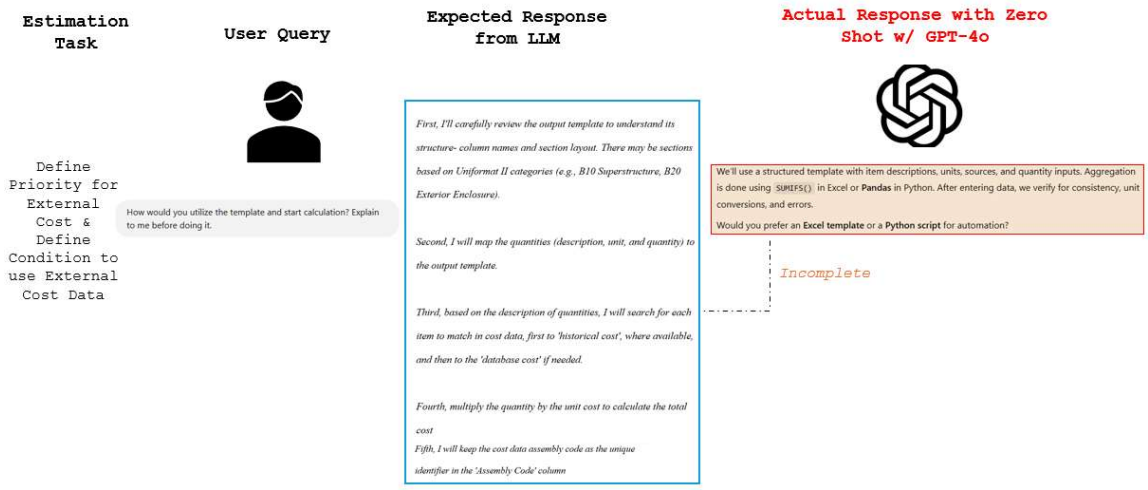


Figure 6. Example 2 of expected vs. actual response in zero-shot.

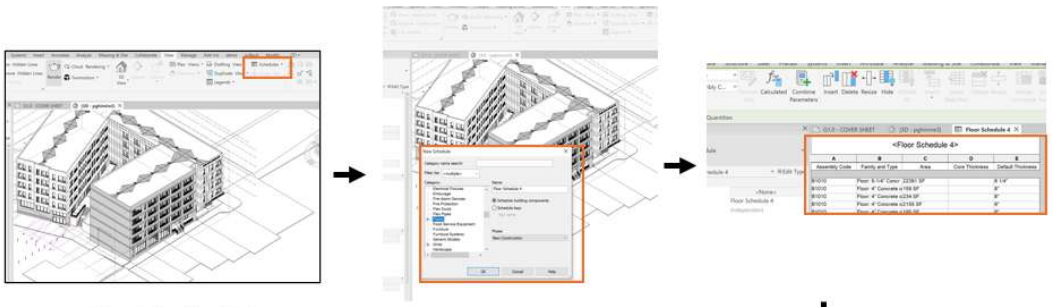
 doi:10.20944/preprints202510.1060.v1

4. Case Study: Modular Chain of Thoughts Prompting for Conceptual Estimation

To investigate the practical applicability of the proposed framework, a case study was conducted by implementing a modular chain-of-thought prompting approach within a commercial construction project in Omaha, Nebraska. This experiment sought to evaluate the effectiveness of a large language model (LLM) in facilitating structured reasoning and task execution within conceptual cost estimation. In practice, cost estimators adhere to standard classification systems such as Uniformat II and proceed through a sequence of integrated activities, i.e., data extraction, structuring, and computational calculation of material quantities and costs. This research suggested an NL query-based modular prompting methodology to optimize workflow accuracy and efficiency. By combining modular components in a systematic manner, this study suggested an NL query-based modular prompting methodology to automate workflow, standardize estimation procedures, reduce cognitive overload, and improve decision-making in construction cost management.

4.1. Data

The Revit model of a commercial construction project in Omaha, Nebraska, which contained all building components with their associated parameters, was utilized to extract quantity information. Using Revit's Schedule/Quantities tool, a structured dataset was generated for Level 3 individual elements classified under Level 1 major group elements of UniFormat II, specifically B-Shell components such as B1010 Floors, B1020 Roofs, B2010 Exterior Walls, B2020 Exterior Windows, and B2030 Exterior Doors, as illustrated in the figure. This process involved selecting relevant element categories, specifying essential parameters- including assembly code, family & type, unit, and area- and formatting the schedule accordingly. Once the schedule was generated, it was exported into a tabular dataset containing detailed quantity takeoffs. The final quantities file, comprising 46 distinct items, organized the extracted data into key attributes: element type, assembly code, family & type, unit, and quantity. Floors and roofs were quantified in square feet, such as concrete slabs and truss framing, whereas doors and windows were recorded as individual units with detailed specifications.



element type	assembly code	family & type	unit	area
floor	B1010	Floor: 5-1/4" Concrete on 3" Metal Deck	sf	22391
floor	B1010	Floor: 4" Concrete on Grade	sf	159
floor	B1010	Floor: 4" Concrete on Grade	sf	234
floor	B1010	Floor: 24" Truss Floor Framing	sf	18979
floor	B1010	Floor: 3/4" Plywd on 10" Wood Joist	sf	1536
floor	B1010	Floor: 24" Truss Floor Framing	sf	18979
floor	B1010	Floor: 3" Concrete on 3" Metal Deck	sf	6419
floor	B1010	Floor: 5" Concrete on Grade	sf	1177
floor	B1010	Floor: 3/4" Plywd on 10" Wood Joist	sf	1536

A section of quantities

Figure 7. Quantities from the Revit model of a commercial building.

To support cost estimation scenario, two distinct datasets were developed: (1) an external cost dataset extracted from RS Means and (2) a historic cost dataset representing an enterprise cost database. These datasets were created to introduce variations and complexities commonly faced in

 doi:10.20944/preprints202510.1060.v1

real-world construction cost estimation process, and evaluate the LLM's interaction with diverse data sources and address challenges similar to those faced by professional estimators.

The external cost dataset was generated by extracting cost information from RS Means for "Commercial New Construction", specifically for the "Assembly" cost category, in "Omaha", using the "2023 Quarter 4" release. The dataset consisted of 100 items corresponding to Level 3 individual components under the Level 1 major group elements of B-Shell, including B1010 Floors, B1020 Roofs, B2010 Exterior Walls, B2020 Exterior Windows, and B2030 Exterior Doors. However, this dataset did not include all items listed in the quantities file but covered the majority of them. Additionally, variations were introduced by incorporating (i) items with exact descriptions matching those in the quantities dataset, (ii) missing items, (iii) completely different items not present in the quantities dataset, and (iv) items with descriptions that were similar but not identical. In addition, to introduce additional complexity, column names in the external cost dataset were intentionally modified to differ slightly from those in the quantities dataset. For example, the column containing item descriptions was labeled as "Description" in the external dataset, whereas it was named "family & type" in the quantities dataset.

A historic cost dataset was created to simulate a construction enterprise cost database, which is commonly used in the construction industry. Unlike the external dataset, this database was manually curated to reflect real-world variations and inconsistencies typically encountered by estimators. The dataset contained 125 items from the same Level 3 component categories of B-Shell, B1010 Floors, B1020 Roofs, B2010 Exterior Walls, B2020 Exterior Windows, and B2030 Exterior Doors. To replicate real-world estimation challenges, this dataset included additional items beyond those in the external dataset and introduced discrepancies in material costs, installation costs, and total unit costs for identical items. Similar to the external dataset, variations were introduced in the form of (i) items with exact descriptions, (ii) missing items, (iii) entirely different items absent in the quantities dataset, and (iv) items with closely related but non-identical descriptions. These two cost datasets were designed to expose the LLM to multiple data sources, their variations in content, formatting, and descriptions, requiring it to interpret and integrate cost information while navigating inconsistencies, missing data, and variations in terminology. The structured, yet intentionally inconsistent nature of the datasets aimed to mimic the real-world complexities of construction cost estimation scenario, where estimators generally refer to different data formats, account for variations in cost parameters, and identify the most relevant pricing information. By simulating these challenges, the study aimed to evaluate the LLM's ability to process and synthesize cost data effectively, improving its reliability in practical estimation workflows. In addition, this study used construction specification data that was synthetically generated to reflect standard practices for commercial building projects, with a focus on the concrete floor system. The dataset included material properties, installation methods, quality assurance measures, and testing requirements, following industry standards such as ACI and ASTM. The major parameters covered concrete mix design, reinforcement details, curing methods, and compliance criteria. The specification data was created to feed into LLM whenever needed for cross reference.

 doi:10.20944/preprints202510.1060.v1

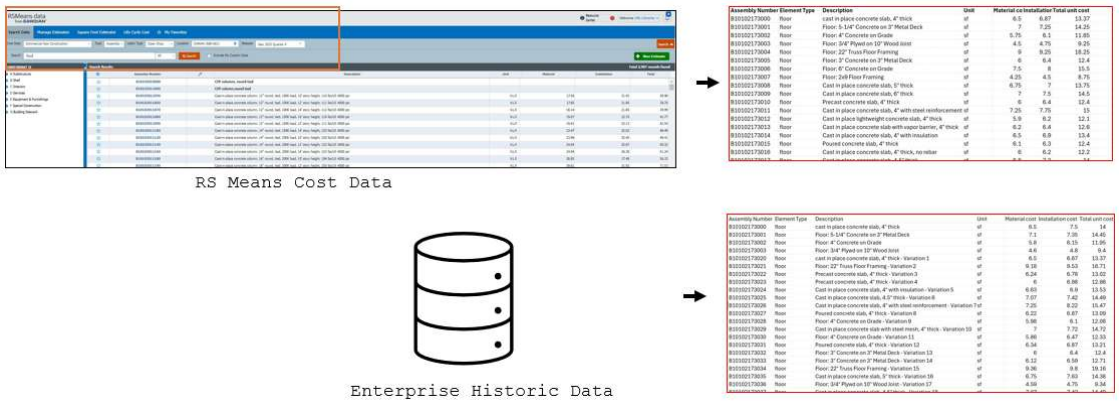


Figure 8. Unit cost from cost databases.

4.2. Modular Framework for the Scenario

The modular framework, as shown in Figure 9 for conceptual cost estimation, was designed to systematically address the complexity of estimation tasks while ensuring adaptability and scalability for ai-assisted workflows. Traditional cost estimation often suffered from fragmented data structures, inconsistencies in cost references, and challenges in retrieving quantity and cost information. These challenges emerged with pre-trained large language models (LLMs) when zero-shot prompting was used for multi-step tasks. These tasks needed to be presented in a clear order; otherwise, LLMs lost track of relationships and context, leading to incomplete reasoning and unreliable estimates. To overcome these limitations, a structured modular approach was developed for a scenario of creating the conceptual estimate, breaking down cost estimation burdens, tasks, and sub-tasks that were identified earlier into four core modules: (1) aggregating quantities, (2) Referencing enterprise historical cost, (3) referencing external cost database, and (4) cross-verification with project specifications. Each module was also decomposed into tasks and sub-tasks so that all the interdependencies were established well to enable a stepwise execution process that was logistically coherent. A modular structure has one of the benefits: it is flexible- each module is ready to be executed independently so that users can work on a specific module without needing to execute all others. This means that if the user simply wants to look at historical cost data, user can enter keywords or Module 2 without having to perform quantity aggregation or external referencing of costs. This modular running capability ensures that different project requirements can be provided for without users having to be restricted to a rigid workflow. This means that if the user simply wants to look at historical cost data, they can enter keywords or Module 2 without having to perform quantity aggregation or external referencing of costs. This modular running capability ensures that different project requirements can be provided for without users having to be restricted to a rigid workflow. For example, in my case, estimating a commercial building project in Omaha, Nebraska, I may only need Module 2 (Enterprise Historic Cost) and Module 4 (Cross-Verification with Project Specifications) while excluding external cost referencing altogether. This modularity allows individuals to use the framework to meet their specific estimation needs without affecting the integrity of the workflow. This design allows users to tailor cost estimation to their specific needs, for example, adding, removing, or adjusting modules, tasks, and sub-tasks, that supports various project types, data availability, and workflow variations without major reconfigurations. For example, in the event a new cost database is introduced, it can be integrated as another module without affecting the existing workflow. Besides straightforward applications, the framework is adaptable to various LLMs and can evolve according to future innovations in LLMs. Since current LLMs struggle with multi-step reasoning in unstructured workflows, this structured approach ensures that activities are logically dependent and sequenced, reducing reliance on open-ended inference. With the evolution of LLMs, particularly long-context reasoning, the system can leverage these advances by enhancing

 doi:10.20944/preprints202510.1060.v1

inter-task dependency monitoring. This modular architecture is an AI-enhanced, scalable solution that enables construction professionals to streamline estimation processes, automate processes, and customize execution according to project-specific needs. As LLM technology evolves, the structured framework ensures AI-aided cost estimation remains future-proof and adaptable while supporting various construction situations without disrupting the core estimation logic.

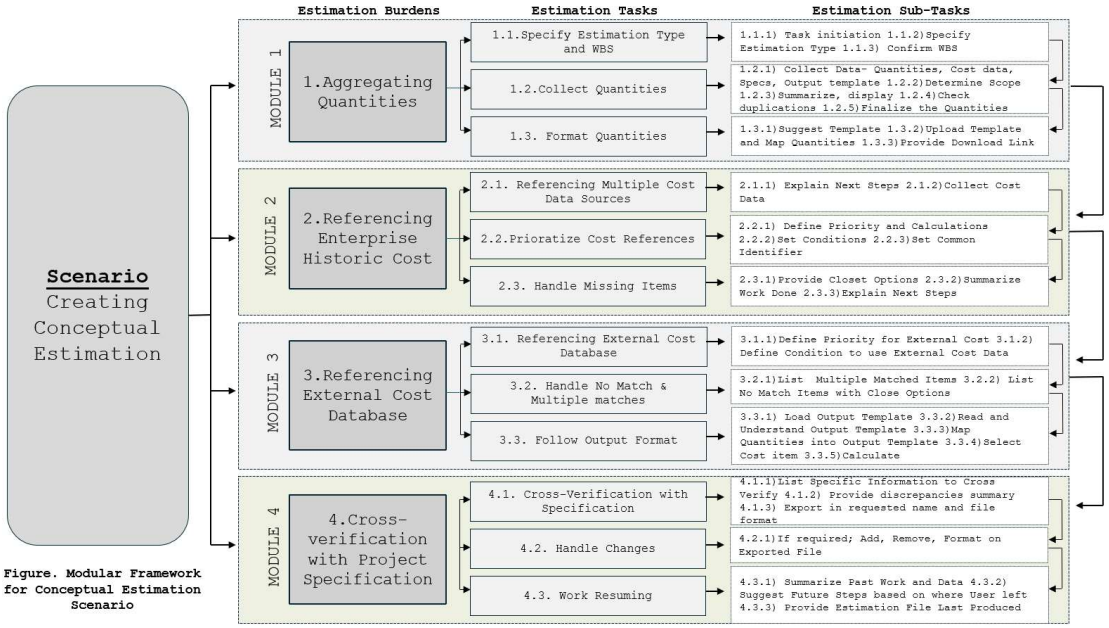


Figure 9. Modular framework for conceptual estimation scenario.

4.3. CoT Instructions

The instruction set for the modular cost estimation framework was developed to provide a structured and flexible approach to estimation. It follows a modular Chain-of-Thought (CoT) system, where tasks and sub-tasks are arranged hierarchically to maintain logical consistency and task dependencies. The structured modular design aimed to improve a key limitation that was identified of zero-shot prompting in large language models (LLMs), which struggle with multi-step reasoning when instructions are not explicitly sequenced (Wei et al., 2022). Each step was designed using user queries and system outputs to give this structure a logical sequence rather than relying on free-form AI responses. This modular structure, as shown in Figure 10, also makes it possible for users to interact with individual modules independently while offering flexibility in the overall estimation process.

[START: USER PROMPT]

↓

[Module 1: Aggregating Quantities] → *[If complete, move to Module 2]*

↓

[Module 2: Referencing Enterprise Historic Cost] → *[If missing, move to Module 3]*

↓

[Module 3: Referencing External Cost Database] → *[If complete, move to Module 4]*

↓

[Module 4: Cross-verification with Specification] → *[If complete, Ask User for Next Step]*

↓

[CONCEPTUAL ESTIMATE] → *[USER REVIEW]* → *[CONTINUE TO NEXT MODULE]* OR *[END]*

 doi:10.20944/preprints202510.1060.v1

Figure 10. Modular flow structure.

The instructions, as shown in Figure 11 example, were framed as a dialogic interaction between the user and the system. They comprise 40 user prompt examples, 40 system response examples, and 35 sub-tasks linked by 35 task connectors. The user submits project details, quantities, cost data references, and estimation preferences, and the system responds with guided prompts, explanations, and formatted outputs. Each module consists of a chain of interrelated tasks carried out conditionally based on the previous inputs. For example, in Module 1 (Aggregating Quantities), the system first collects estimation details, confirms the work breakdown structure (WBS), and compiles quantity data before moving to Module 2 (historic cost referencing). If historical cost data is unavailable, the system automatically redirects execution to Module 3 (external Cost Database) to retrieve the missing information. The modular design ensures workflow follows a sequential pattern, reducing errors and inconsistencies in mapping and process transparency. The flexibility of one of the advantages of modular CoT instruction system is that users can add, remove, or modify modules, tasks, and sub-tasks based on project needs [47]. This makes the framework scalable for different construction projects, estimation approaches, and data sources. For instance, if a new database of costs is released, it can be added as another module without any changes to the current workflow. Similarly, as LLMs improve in long-context reasoning and task dependency tracking, the framework can incorporate more advanced AI-assisted decision-making. The structured modular CoT approach aligns with AI advancements, demonstrating that explicitly defined reasoning pathways improve AI performance in complex multi-step tasks [53].

```
### Sequential Chain of Thought for Scenario 1: Conceptual Estimation

#### Module 1: Aggregating Quantities

##### Task ID 1.1: Specify Estimation Type and WBS
|__ Sub-Task ID 1.1.1: Task Initiation
 |__ User: "I have a new project for which I need to create an estimate. Can you help me?"
 |__ System: "Yes, I can. Could you provide some details about the project, such as the type of construction, location of the project,
       and any specific type of estimation- Assembly or Item based- that you want to follow?"
 |__ Connection to Task ID 1.1.2: Asking for project details and estimation type to initiate further discussion.

|__ Sub-Task ID 1.1.2: Specify Estimation Type
 |__ User: "This is a new commercial building construction in Omaha, Nebraska. I want to do an Assembly cost estimation."
 |__ System: "Ok. For Assembly estimation, we'll need to break down the project into Uniformat II categories before starting.
       Do you know how Assembly estimation works with Uniformat II?"
 |__ Connection to Task ID 1.1.3: Asking for confirmation on the user's understanding of the Assembly breakdown.

|__ Sub-Task ID 1.1.3: Confirm Work Breakdown Structure (WBS)
 |__ User: "No, can you explain the Assembly breakdown in Uniformat II?"
 |__ System: "Of course! Uniformat II divides a building into several functional assemblies, such as Substructure, Shell, Interiors, Services, and more.
       For example, 'B10 Superstructure' covers the main load-bearing elements, and 'B20 Exterior Enclosure' refers to exterior walls and windows.
       This breakdown helps categorize costs based on building elements. Would you like more details on Uniformat II and each category?"
 |__ Connection to Task ID 1.1.4: Proceeding with the breakdown explanation and ensuring the user's understanding. Next, to Collect data.

##### Task ID 1.2: Collect Quantities
|__ Sub-Task ID 1.2.1: Collect Data- Quantities, Cost data, Specs, Output template
 |__ User: "Ok, understood. Now how can I start?"
 |__ System: "Now, you need to provide quantities, cost data, project specifications, and a particular format or template for the final output of the estimate."
 |__ User: "OK. Let's go one by one with the data."
 |__ System: "Great. Provide me with quantities first."
 |__ Connection to Task ID 1.2.2: Focusing on collecting data for quantities and Confirming scope for Quantities.

|__ Sub-Task ID 1.2.2: Determine Scope
 |__ User: "OK. I only have quantities for the shell category. And want to estimate only that portion."
 |__ System: "I got it! I will focus on the Shell (B) category for now. Please provide the quantities."
 |__ Connection to Task ID 1.2.2: Displaying Items and Quantities to user.

|__ Sub-Task ID 1.2.3: Summarize and Display Collected Quantities
 |__ User: "Here's the quantity Excel file I have uploaded. Please show me a list of items and quantities in tabular format."
 |__ System: Ok. Here is the list of your items and respective quantities. You have total 111 items .
       "Do you want to know more about or do you want to move ahead to next steps?"
 |__ Connection to Task ID 2.1.1: Analyzing as per user request- Duplication check.

|__ Sub-Task ID 1.2.4: Check Duplications
 |__ User: "Can you check if there are any duplicate items based on the description and assembly code?"
 |__ System: "These are items that have exact same description and assembly code. Do you want to add them to a single item?"
 |__ User: "Yes, please add. Also, can you find potential duplication that doesn't match exactly but potentially?"
 |__ System: "Here are these group items that look potentially similar based on size, and descriptions."
 |__ Connection to Task ID 1.2.5: Finalizing Quantities.
```

Figure 11. Chain-of-thoughts instruction example.

4.4. Modular CoT Architecture

The modular CoT architecture illustrated in Figure 12 is a systematic and flexible framework for cost estimation. It ensures the tasks are logically sequential and hierarchically run and permits modular flexibility. This is particularly vital in AI-integrated workflow when considering specific task ordering, as it is important for maintaining logical progression and awareness of context. This architecture is specified through a multi-layered module structure in which the estimation process is divided into modules, tasks, and sub-tasks with clear execution steps and dependencies. Each

 doi:10.20944/preprints202510.1060.v1

module (e.g., Module 1: Aggregating Quantities) is subdivided into tasks (e.g., 1.1: Specify Estimation Type, 1.2: Collect Quantities) and further into sub-tasks (e.g., 1.1.1: Confirm WBS, 1.1.2: Validate Scope), ensuring sequential and conditional execution. This numbering structure prevents logical drift by maintaining strict task dependencies, requiring lower-level sub-tasks (e.g., 1.2.3: Remove Duplicates) to be completed before higher-level transitions occur. The organized segmenting allows the system to behave sequentially or conditionally so that each module can be treated as a standalone process or integrated workflow. At the highest level, this architecture consists of modules representing major estimation components of conceptual estimation scenarios such as aggregating quantities, cost referencing, and specification cross-verification. Each module contains tasks that define specific cost estimation objectives, which are then broken down into sub-tasks for data collection, validation, and decision-making. This structure enables LLMs to carry out tasks step by step, based on instructions, reducing errors and enabling transparency in cost estimation. The input for the user is an NL query, and the LLM converts it into the relevant module through the module calling function. After completing a module, the system proceeds to the next module according to the provided instructions beforehand. This approach maintains each step of the estimation process connected to each other.

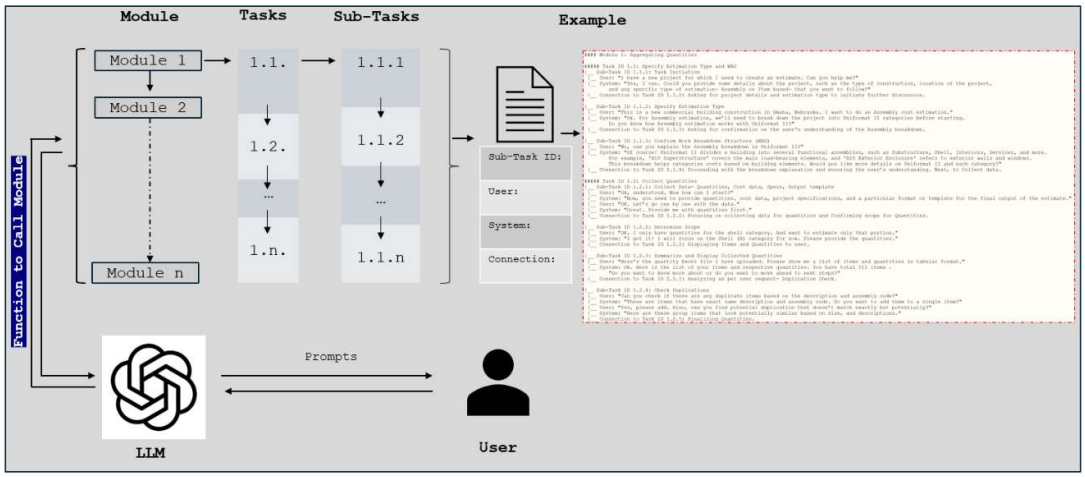


Figure 12. Modular CoT architecture for conceptual estimation.

4.5. Module Calling Function

The module calling function is designed to dynamically select and execute modules based on user input. By enforcing a consistent decision-making approach, this function prevents arbitrary execution of multi-step processes and maintains task dependencies. The module calling function is illustrated in Figure 13. It processes NL prompts, parses it, maps queries to the appropriate modules, confirms selections, and executes the chosen module while verifying that all prerequisite tasks are completed. The execution process begins with user input analysis, where the system extracts key terms and matches them to a predefined module dictionary. When a user submits a prompt, the system first conducts a task analysis and module selection process, evaluating the request against predefined categories. If an exact match is not identified, the system asks for further clarification, either by prompting the user for additional details or by suggesting the most suitable modules based on the detected keywords. This step is designed to prevent misclassification by providing the user a list of current modules and subsequent steps with detail descriptions for confirmation. This confirmation also supports active user involvement in the process. After the user's confirmation, the function executes the module by doing its tasks and sub-tasks in logical order. To further support task automation, this function utilizes a dictionary-based keyword mapping system that links specific terminology to predefined modules. For example, commonly used terms such as "Estimation Type" and "WBS" are automatically assigned to Module 1: Aggregating Quantities. This dynamic module

 doi:10.20944/preprints202510.1060.v1

calling function works to eliminate ambiguity and allows the system to automatically map user queries to the correct estimation module with consistency under different estimation scenarios.

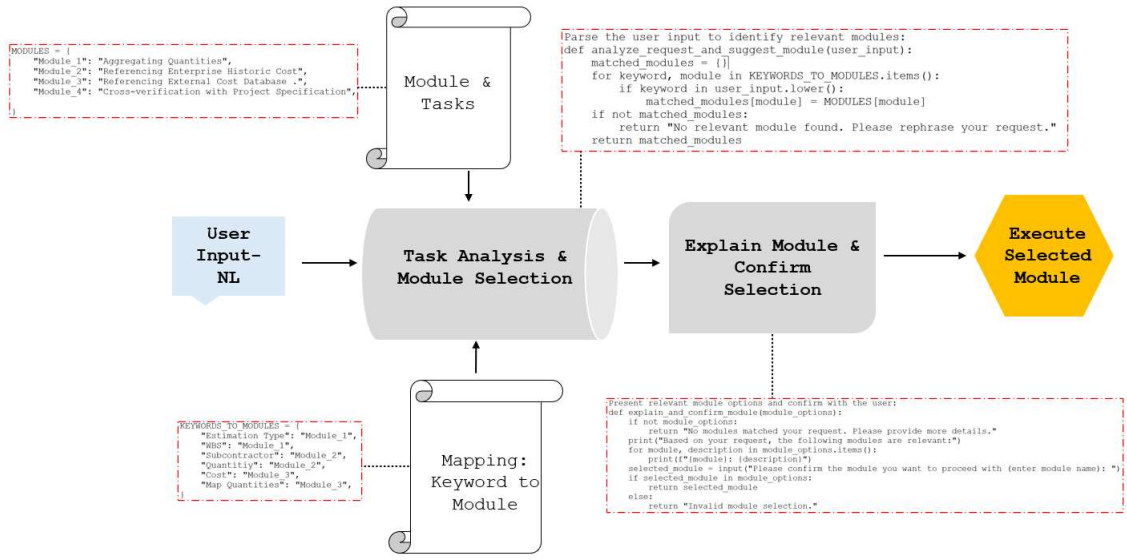


Figure 13. Module calling function.

The main workflow, which includes analyzing the request, confirming module selection, and executing the corresponding module, is shown in Figure 14.

```
5. MAIN WORKFLOW
Integrate all steps into the main execution logic:
if __name__ == "__main__":
    print("Welcome to the Construction Cost Estimation Assistant!")
    user_input = input("This is a new commercial building construction in Omaha, Nebraska.
         I want to do an Assembly cost estimation.: ")

    # Step 1: Analyze request and suggest modules
    module_options = analyze_request_and_suggest_module(user_input)
    if isinstance(module_options, str):  # If no match is found
        print(module_options)
    else:
        # Step 2: Explain and confirm module selection
        selected_module = explain_and_confirm_module(module_options)
        if "Invalid" in selected_module:
            print(selected_module)
        else:
            # Step 3: Execute selected module
            result = execute_selected_module(selected_module, user_input)
            print(result)

Error Handling:
- Unmatched Input: "No relevant module found. Please rephrase your request OR
 I can provide you available module options to select from. Please let me know how you want to proceed"
- Invalid Selection: "Invalid module selection. Please confirm a valid module."
"""
```

Figure 14. Module calling function workflow.

5. Results & Discussion

This study employed two evaluation approaches to assess GPT-4o's overall performance in conceptual cost estimation, combining qualitative and quantitative methods. The qualitative evaluation involved human ratings based on confidence on LLMs output and module following, where confidence measures the model's certainty in its responses, and module following assessed its ability to follow predefined workflows. Meanwhile, the quantitative evaluation was conducted using five widely used NLP metrics- BLEU, ROUGE-L, METEOR, Content Overlap, and Semantic Similarity- to measure response accuracy, fluency, and contextual alignment.

 doi:10.20944/preprints202510.1060.v1

5.1. Qualitative Evaluation

Human evaluation is important in assessing the effectiveness of large language models (LLMs), particularly in complex and connected tasks such as cost estimation. Human evaluation accounts for reasoning, contextual accuracy, and real-world applicability[50,72–74]. This study rated confidence scores on a scale of 1 to 3, based on how complete, accurate, and aligned each response was with the correct module. A score of 1 represented low confidence, i.e., the response was incomplete, incorrect, or did not correspond to the expected module. A score of 2 represented medium confidence, in which the response was partially correct and corresponded to the correct module but was not completely accurate or complete. Finally, a score of 3 represented high confidence, i.e., the response was completely accurate, complete, and corresponding to the correct module. The average confidence score of 2.52 (84%) in modular CoT prompting marked an 20% improvement over zero-shot evaluation (1.91, 64%) for the provided cost estimation scenario, confirming that structured reasoning improves response reliability. Figure 15 illustrates comparative examples of zero-shot vs. CoT responses, highlighting improvements in response quality and module adherence.

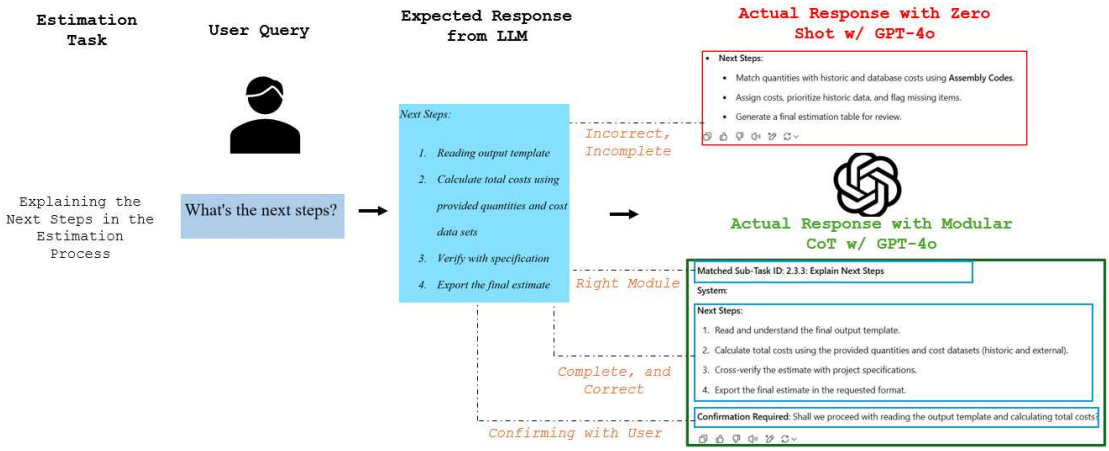


Figure 15. Example of expected vs. actual response in modular CoT.

5.2. Quantitative Evaluation

There are five common evaluation metrics- BLEU, ROUGE-L, METEOR, Content Overlap, and Semantic Similarity- that measure the LLMs outputs are not only numerically correct but also contextually accurate, linguistically coherent, and semantically aligned with expected responses [75–79]. A single metric could not fully evaluate the LLM performance on cost estimation text generation, where technical accuracy, structured reasoning, and factual completeness are equally important. In evaluating the performance of GPT-4o on cost estimation tasks, this study compared Zero-Shot and Chain-of-Thought (CoT) prompting across five key metrics: BLEU, ROUGE-L, METEOR, Content Overlap, and Semantic Similarity. Table 7 presents a comparison on the average CoT performance scores across all metrics, evaluated on 24 sub-tasks from Modules 1 and 2 in Scenario 1 - Conceptual Estimation. These results are compared with the previously conducted zero-shot evaluation, as shown in Table 4. This presents a quantitative assessment of how CoT prompting influences the quality of generated responses. Each metric captures a different aspect of text generation- BLEU measures exact word match precision, ROUGE-L evaluates phrase-level recall, METEOR incorporates synonym and stemming considerations, while content overlap and semantic similarity assess factual consistency and conceptual alignment, respectively.

Table 4. GPT-4o Performance Evaluation- Zero-Shot vs. CoT.

Evaluation Category	Zero-Shot with GPT4o	CoT with GPT4o

 doi:10.20944/preprints202510.1060.v1

BELU	0.023365	0.382353
ROUGE_L	0.185215	0.622845
METEOR	0.196798	0.610922
Content Overlap	0.109057	0.497031
Semantic Similarity	0.245202	0.597011

The Bilingual Evaluation Understudy (BLEU) score [64] is a standard metric for evaluating text generation models by comparing their outputs to reference texts. It operates on n-gram precision and penalizes overly short responses through a brevity penalty factor:

$$BLEU = \mathrm{BP}.\exp\left(\sum_{n=1}^{N} \mathrm{w_n}\ \mathrm{logp_n}\right)$$

where $\mathrm{p_n}$ represents the precision of n-grams, $\mathrm{w_n}$ is a weight distribution, and BP accounts for length differences. In the conceptual estimation scenario, the zero-shot BLEU score was 0.0234, highlighting a lack of direct word alignment between the generated and expected response. With CoT prompting, BLEU rose significantly to 0.3824, marking a 1536.43% increase. This drastic improvement suggests that CoT facilitates a structured approach to LLM to generate response, thereby reducing errors in word choice and improving syntactic accuracy.

The Recall-Oriented Understudy for Gisting Evaluation (ROUGE-L) [68] extends beyond n-gram precision by evaluating the longest common subsequence (LCS) between generated and reference texts:

$$\mathrm{ROUGE_L} = \frac{LCS(X,Y)}{|Y|}$$

where X is the generated response, and Y is the reference text. The zero-shot ROUGE-L score of 0.1852, indicating relatively weak phrase alignment and coherence. CoT prompting, however, boosted ROUGE-L to 0.6228, a 236.28% improvement. This result shows that CoT-driven responses have more relevant output when compared with the expected response from LLM, suggesting that logical connection in prompt help to improve in producing more contextually aligned responses.

Unlike BLEU and ROUGE, the Metric for Evaluation of Translation with Explicit Ordering (METEOR) [69] incorporates stemming, synonym matching, and function word weighting, offering a more nuanced reflection of semantic alignment. It is computed as:

$$METEOR = \mathrm{F_{mean}}\ X\ (1 - \mathrm{penalty})$$

where $\mathrm{F_{mean}}$ is the harmonic mean of precision and recall, and the penalty term discourages fragmented matches. The zero-shot METEOR score of 0.1968 indicated poor conceptual resemblance between expected and generated texts. However, with CoT prompting, METEOR rose to 0.6109, reflecting a 210.43% improvement. This suggests that CoT enhances the model's ability to use more appropriate word choices, improving fluency, synonym integration, and contextual awareness.

Content overlap measures the Jaccard similarity [80] between words in generated and reference texts:

$$\mathrm{J(A,B)} = \frac{|\mathrm{A}\ \cap \mathrm{B}|}{|\mathrm{A}\ \cup \mathrm{B}|}$$

where A and B are the sets of words in the generated and expected responses, respectively. The zero-shot content overlap was just 0.1091, indicating that most words in the generated text were not directly present in the reference. However, CoT increased this value to 0.4970, representing a 355.75% improvement. This contrast underscores how CoT prompting improves the inclusion of key factual elements, helping the model generate responses that align more precisely with expected content.

 doi:10.20944/preprints202510.1060.v1

Semantic similarity, unlike content overlap, focuses on the meaning preservation between texts rather than exact word matches [78,81]. A common approach is cosine similarity over embedding vectors:

$$\cos(\theta) = \frac{X \cdot Y}{||X||\,||Y||}$$

where X and Y are the embedding representations of the generated and output texts. The zero-shot semantic similarity score was 0.2452, reflecting a substantial gap in conceptual alignment between model output and reference responses. CoT prompting increased this score to 0.5970, a 143.48% improvement. This result suggests that CoT enables GPT-4o to generate responses that are more semantically meaningful, ensuring greater alignment with the intended concepts rather than just syntactic resemblance.

A comparison, as shown in Figure 16, shows that CoT prompting substantially improves model outputs across all metrics, suggesting that structured reasoning enhances coherence, factual retention, and fluency in GPT-4o's responses on cost estimation scenario and tasks. The bar chart visualizes the comparison between Zero-Shot GPT-4o and CoT GPT-4o across five evaluation categories.

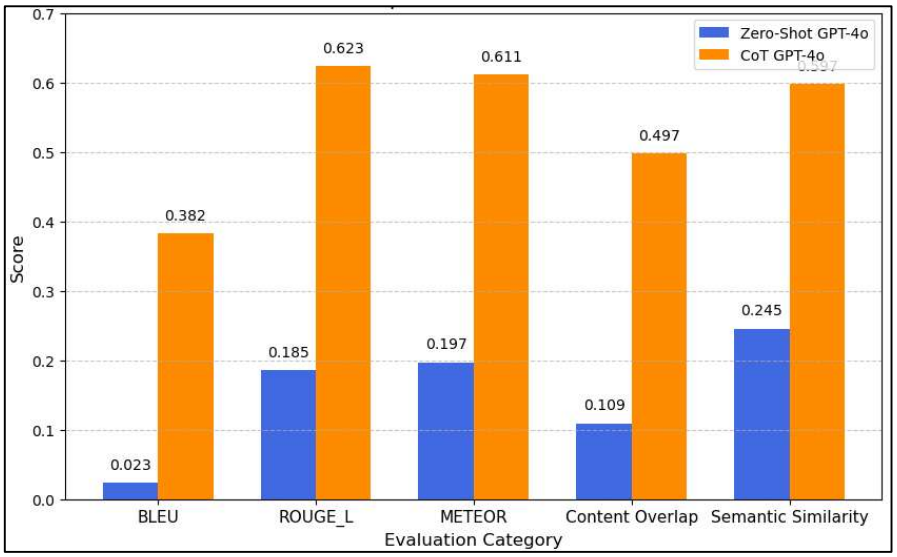


Figure 16. Performance comparison: zero-shot vs. CoT.

To further investigate whether the observed improvements were statistically significant, a paired t-test was conducted:

$$t = \frac{\bar{d}}{s_d/\sqrt{n}}$$

where $\bar{d}$ is the mean difference, s_d the standard deviation of differences, and $\sqrt{n}$ is the number of evaluation categories. The test returned to a t-statistic of 24.02 and a p-value of 0.000018, well below the 0.05 significance threshold. This confirms that the improvements achieved through CoT prompting is a significant performance improvement. These findings strongly support the adoption of CoT prompting strategies in improving LLM outputs on estimation tasks, confirming greater accuracy, contextual understanding, and meaningfulness in responses.

6. Limitations and Future Work

While this study evaluated the performance of existing LLMs in construction cost estimation tasks and demonstrated the effectiveness of the CoT prompting approach, several limitations must be acknowledged. First, although different components are involved in construction cost estimation,

 doi:10.20944/preprints202510.1060.v1

including conceptual estimation, evaluating subcontractor estimates, and change management, the scope of this study was a conceptual estimation scenario using pre-trained LLMs. Future research can extend the evaluation to other scenarios to assess how LLMs handle subcontractor bid analysis, estimation completeness checks, and version control in cost estimation workflows. Second, this study focused specifically on the Uniformat II, specifically within the Shell category, limiting the scope of cost estimation components analyzed. Future research can also apply this analysis to all Uniformat II categories, for a more comprehensive evaluation of LLMs' performance in handling diverse cost elements of various building systems. In addition, future research can consider CoT prompting approach to other estimation techniques such as unit cost estimate. Third, while CoT prompting significantly improved LLM performance compared to zero-shot prompting, its practical integration into real-world estimation workflows remains unexplored. Future research should focus on developing and deploying AI-driven cost estimation assistants that integrate CoT-enhanced LLMs into existing industry workflows.

7. Conclusion

Construction cost estimation has long been a manual and time-consuming process, heavily dependent on expert judgment, intuition, and past experience. And, estimators generally work with fragmented data sources and data formats, making the process time-intensive. With tight project deadlines, even small miscalculations can lead to costly errors that affect both budgets and timelines. While LLMs are being explored for various construction applications, their zero-shot performance in cost estimation and the potential for structured prompting improvements remain underexplored. This study addressed these gaps by developing a modular Chain-of-Thought framework specific to cost estimation workflows and systematically evaluating LLMs using a scenario-based approach.This study developed a conceptual estimation scenario and tested using pre-trained LLMs under zero-shot prompting, indicating gaps in completeness and accuracy, with human evaluation yielding an average confidence score of 1.906 (64%). Among the evaluated models- GPT-4o, LLaMA 3.2, Gemini 2.0, and Claude 3.5 Sonnet- GPT-4o performed best across BLEU, ROUGE-L, METEOR, Content Overlap, and Semantic Similarity metrics. The modular CoT prompting approach, incorporating structured reasoning and modular task awareness, significantly improved estimation performance, with human evaluation reflecting an improved confidence score of 2.52 (84%)- a 20% increase over zero-shot performance. Based on the results, this study concludes that pre-trained LLMs without additional knowledge are not reliable for performing detailed cost estimation tasks. However, their performance can be largely improved through the use of a Chain-of-Thought approach. This study makes theoretical, methodological, and practical contributions by demonstrating how pre-trained LLMs can be adapted for cost estimation, introducing a modular CoT framework to standardize estimation workflows, and providing structured estimation scenarios and prompts for industry adoption.

Appendix A. Burdens to Taks to Sub- Tasks Mapping [59]

Estimation Burdens	Tasks	Sub-Tasks
Aggregating Quantities	Specify Estimation Type and WBS	▪ *Task initiation* ▪ *Specify Estimation Type* ▪ *Confirm WBS*
	Collect Quantities	▪ *Collect Data- Quantities, Cost data, Specs, Output template* ▪ *Determine Scope*

 doi:10.20944/preprints202510.1060.v1

		▪ *Summarize, display* ▪ *Check duplications* ▪ *Finalize the Quantities*
	Format Quantities	▪ *Suggest Template* ▪ *Upload Template and Map Quantities* ▪ *Provide Download Link*
Referencing Enterprise Historic Cost	Referencing Multiple Cost Data Sources	▪ *Explain Next Steps* ▪ *Collect Cost Data*
	Prioritize Cost References	▪ *Define Priority and Calculations* ▪ *Set Conditions* ▪ *Set Common Identifier*
	Handle Missing Items	▪ *Provide Closet Options* ▪ *Summarize Work Done* ▪ *Explain Next Steps*
Referencing External Cost Database	Referencing External Cost Database	▪ *Set Priority for External Cost* ▪ *Define Condition to use External Cost Data*
	Handle No Match & Multiple matches	▪ *List of Multiple Matched Items* ▪ *List No Match Items with Close Options*
	Follow Output Format	▪ *Load Output Template* ▪ *Read and Understand Output Template* ▪ *Map Quantities into Output Template* ▪ *Select Cost item* ▪ *Calculating*

 doi:10.20944/preprints202510.1060.v1

Cross-verification with Project Specification	Cross-Verification with Specification	▪ *List Specific Information to Cross Verify* ▪ *Provide discrepancies summary* ▪ *Export in requested name and file format*
	Handle Changes	▪ *If required; Add, Remove, Format on Exported File*
	Work Resuming	▪ *Summarize Past Work and Data* ▪ *Suggest Future Steps based on where user left* ▪ *Provide Estimation File Last Produced*
Planning Evaluation	Initialize Subcontractor Evaluation	▪ *Context Understanding* ▪ *Plan context based Evaluation*
	Handle Inconsistencies & Mapping	▪ *Review Submissions* ▪ *Plan Mapping into Standard Template* ▪ *Handle Multiple Submissions in Trade* ▪ *Compile Estimates*
Evaluating Completeness	Check Completeness	▪ *Understand Completeness Criteria* ▪ *Cross Check with Conceptual Estimate* ▪ *Flag Incomplete Submissions*
	Evaluating Discrepancies	▪ *List Missing Items* ▪ *List Additional Items* ▪ *Remove Incomplete Submissions*
	Identify Options	▪ *Identify Trades with Multiple Proposals*

 doi:10.20944/preprints202510.1060.v1

Metrics Based Evaluation	Metrics Based Evaluation	▪ *Understand Evaluation Metrics* ▪ *Evaluation* ▪ *Explain Calculations and Priority List* ▪
	Create Negotiation List	▪ *Find Outstanding Items* ▪ *List of Outstanding Items* ▪ *Create Negotiation Table*
Visualizing and compiling	Calculate and visualize	▪ *Calculate Differences* ▪ *Visualize Differences*
	Compile Estimates	▪ *List Selected Subs* ▪ *Compile Selected Subs Estimates* ▪ *Add Self-Performing Trades*
	Generate Final Estimate	▪ *Element Breakdown Visualization* ▪ *Compare Final vs Conceptual Estimate* ▪ *Generate Final Output File*
Managing Changes	Navigate Recent Estimation	▪ *Open Recent Estimation*
	Update Changes	▪ *Integrate Meeting Updates* ▪ *List Changes*
Version Control	Naming Standard	▪ *Automatic Naming*
	Changes between Versions	▪ *Summary of changes*
Data Re-Cycling	Recycle to Historic database	▪ *Add Metadata to Historic Data* ▪ *Ensure Formatting Consistency* ▪ *Appending*

References

[1] Z. H. Ali and A. M. Burhan, "Hybrid machine learning approach for construction cost estimation: an evaluation of extreme gradient boosting model," *Asian J Civ Eng*, vol. 24, no. 7, pp. 2427–2442, Nov. 2023, doi: 10.1007/s42107-023-00651-z.

 doi:10.20944/preprints202510.1060.v1

[2] O. Swei, J. Gregory, and R. Kirchain, "Construction cost estimation: A parametric approach for better estimates of expected cost and variation," *Transportation Research Part B: Methodological*, vol. 101, pp. 295–305, Jul. 2017, doi: 10.1016/j.trb.2017.04.013.

[3] S. Tayefeh Hashemi, O. M. Ebadati, and H. Kaur, "Cost estimation and prediction in construction projects: a systematic review on machine learning techniques," *SN Appl. Sci.*, vol. 2, no. 10, p. 1703, Sep. 2020, doi: 10.1007/s42452-020-03497-1.

[4] L. Holm and J. E. Schaufelberger, *Construction Cost Estimating*. London: Routledge, 2021. doi: 10.1201/9781003023494.

[5] D. D. Ahiaga-Dagbui and S. D. Smith, "Rethinking construction cost overruns: cognition, learning and estimation," *Journal of Financial Management of Property and Construction*, vol. 19, no. 1, pp. 38–54, Apr. 2014, doi: 10.1108/JFMPC-06-2013-0027.

[6] P. Ghimire, S. Pokharel, K. Kim, and P. Barutha, "Machine learning-based prediction models for budget forecast in capital construction," in *Proceedings of the 2nd International Conference on Construction, Energy, Environment & Sustainability, Funchal, Portugal*, 2023, pp. 27–30.

[7] J. Messner, "Introduction to Construction Cost Estimating," Aug. 2022, Accessed: Mar. 08, 2025. [Online]. Available: https://psu.pb.unizin.org/buildingconstructionmanagement/chapter/introduction-to-construction-cost-estimating/

[8] F. H. Abanda, B. Kamsu-Foguem, and J. H. M. Tah, "BIM – New rules of measurement ontology for construction cost estimation," *Engineering Science and Technology, an International Journal*, vol. 20, no. 2, pp. 443–459, Apr. 2017, doi: 10.1016/j.jestch.2017.01.007.

[9] P. Ghimire, K. Kim, and M. Acharya, "Opportunities and Challenges of Generative AI in Construction Industry: Focusing on Adoption of Text-Based Models," *Buildings*, vol. 14, no. 1, Art. no. 1, Jan. 2024, doi: 10.3390/buildings14010220.

[10] N. Rane, "Role of ChatGPT and Similar Generative Artificial Intelligence (AI) in Construction Industry," Oct. 10, 2023, *Social Science Research Network, Rochester, NY*: 4598258. doi: 10.2139/ssrn.4598258.

[11] M.-Y. Cheng, H.-C. Tsai, and W.-S. Hsieh, "Web-based conceptual cost estimates for construction projects using Evolutionary Fuzzy Neural Inference Model," *Automation in Construction*, vol. 18, no. 2, pp. 164–172, Mar. 2009, doi: 10.1016/j.autcon.2008.07.001.

[12] H. H. Elmousalami, "Artificial Intelligence and Parametric Construction Cost Estimate Modeling: State-of-the-Art Review," *Journal of Construction Engineering and Management*, vol. 146, no. 1, p. 03119008, Jan. 2020, doi: 10.1061/(ASCE)CO.1943-7862.0001678.

[13] J. R. Walton and J. D. Stevens, "Improving Conceptual Estimating Methods Using Historical Cost Data," *Transportation Research Record*, vol. 1575, no. 1, pp. 127–131, Jan. 1997, doi: 10.3141/1575-18.

[14] S.-H. Ji, M. Park, and H.-S. Lee, "Cost estimation model for building projects using case-based reasoning," *Can. J. Civ. Eng.*, vol. 38, no. 5, pp. 570–581, May 2011, doi: 10.1139/l11-016.

 doi:10.20944/preprints202510.1060.v1

[15] R. P. Charette and H. E. Marshall, "UNIFORMAT II elemental classification for building specifications, cost estimating, and cost analysis," National Institute of Standards and Technology, Gaithersburg, MD, NIST IR 6389, 1999. doi: 10.6028/NIST.IR.6389.

[16] M. Sayed, M. Abdel-Hamid, and K. El-Dash, "Improving cost estimation in construction projects," *International Journal of Construction Management*, vol. 23, no. 1, pp. 135–143, Jan. 2023, doi: 10.1080/15623599.2020.1853657.

[17] M. Juszczyk, "The Challenges of Nonparametric Cost Estimation of Construction Works with the use of Artificial Intelligence Tools," *Procedia Engineering*, vol. 196, pp. 415–422, Jan. 2017, doi: 10.1016/j.proeng.2017.07.218.

[18] C. Lim, W.-K. Hong, D. Lee, and S. Kim, "Automatic Rebar Estimation Algorithms for Integrated Project Delivery," *Journal of Asian Architecture and Building Engineering*, vol. 15, no. 3, pp. 411–418, 2016, doi: 10.3130/jaabe.15.411.

[19] A. O. Elfaki, S. Alatawi, and E. Abushandi, "Using Intelligent Techniques in Construction Project Cost Estimation: 10-Year Survey," *Advances in Civil Engineering*, vol. 2014, no. 1, p. 107926, 2014, doi: 10.1155/2014/107926.

[20] RSMeans and S. A. Mubarak, *How to Estimate with RSMeans Data: Basic Skills for Building Construction*. John Wiley & Sons, 2020.

[21] S. O. Babatunde, S. Perera, D. Ekundayo, and T. E. Adeleye, "An investigation into BIM-based detailed cost estimating and drivers to the adoption of BIM in quantity surveying practices," *Journal of Financial Management of Property and Construction*, vol. 25, no. 1, pp. 61–81, Nov. 2019, doi: 10.1108/JFMPC-05-2019-0042.

[22] A. Wahab and J. Wang, "Factors-driven comparison between BIM-based and traditional 2D quantity takeoff in construction cost estimation," *Engineering, Construction and Architectural Management*, vol. 29, no. 2, pp. 702–715, Mar. 2021, doi: 10.1108/ECAM-10-2020-0823.

[23] I. Mahamid, "Factors affecting cost estimate accuracy: Evidence from Palestinian construction projects," *International Journal of Management Science and Engineering Management*, vol. 10, no. 2, pp. 117–125, Apr. 2015, doi: 10.1080/17509653.2014.925843.

[24] T. Akanbi and J. Zhang, "Design information extraction from construction specifications to support cost estimation," *Automation in Construction*, vol. 131, p. 103835, Nov. 2021, doi: 10.1016/j.autcon.2021.103835.

[25] S. Feuerriegel, J. Hartmann, C. Janiesch, and P. Zschech, "Generative AI," *Bus Inf Syst Eng*, vol. 66, no. 1, pp. 111–126, Feb. 2024, doi: 10.1007/s12599-023-00834-7.

[26] D. Baidoo-anu and L. O. Ansah, "Education in the Era of Generative Artificial Intelligence (AI): Understanding the Potential Benefits of ChatGPT in Promoting Teaching and Learning," *Journal of AI*, vol. 7, no. 1, Art. no. 1, Dec. 2023, doi: 10.61969/jai.1337500.

[27] J. Zheng and M. Fischer, "Dynamic prompt-based virtual assistant framework for BIM information search," *Automation in Construction*, vol. 155, p. 105067, Nov. 2023, doi: 10.1016/j.autcon.2023.105067.

[28] C. Li, Y. Su, and W. Liu, "Text-To-Text Generative Adversarial Networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2018, pp. 1–7. doi: 10.1109/IJCNN.2018.8489624.

 doi:10.20944/preprints202510.1060.v1

[29] C. Zhang, C. Zhang, M. Zhang, and I. S. Kweon, "Text-to-image Diffusion Models in Generative AI: A Survey," Apr. 02, 2023, *arXiv*: arXiv:2303.07909. doi: 10.48550/arXiv.2303.07909.

[30] V. Liu, T. Long, N. Raw, and L. Chilton, "Generative Disco: Text-to-Video Generation for Music Visualization," Apr. 17, 2023, *arXiv*: arXiv:2304.08551. Accessed: Aug. 27, 2023. [Online]. Available: http://arxiv.org/abs/2304.08551

[31] T. Lei, R. Barzilay, and T. Jaakkola, "Rationalizing Neural Predictions," Nov. 02, 2016, *arXiv*: arXiv:1606.04155. Accessed: Aug. 27, 2023. [Online]. Available: http://arxiv.org/abs/1606.04155

[32] A. N. Wu, R. Stouffs, and F. Biljecki, "Generative Adversarial Networks in the built environment: A comprehensive review of the application of GANs across data types and scales," *Building and Environment*, vol. 223, p. 109477, Sep. 2022, doi: 10.1016/j.buildenv.2022.109477.

[33] I. Goodfellow et al., "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, Oct. 2020, doi: 10.1145/3422622.

[34] D. P. Kingma and M. Welling, "An Introduction to Variational Autoencoders," *MAL*, vol. 12, no. 4, pp. 307–392, Nov. 2019, doi: 10.1561/2200000056.

[35] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners".

[36] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2020, pp. 6840–6851. Accessed: Sep. 16, 2023. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html

[37] M. Kumar et al., *VideoFlow: A Flow-Based Generative Model for Video*. 2019.

[38] J. Lee, H. Kim, J. Shim, and E. Hwang, "Cartoon-Flow: A Flow-Based Generative Adversarial Network for Arbitrary-Style Photo Cartoonization," in *Proceedings of the 30th ACM International Conference on Multimedia*, in MM '22. New York, NY, USA: Association for Computing Machinery, Oct. 2022, pp. 1241–1251. doi: 10.1145/3503161.3548094.

[39] H. Wan, J. Zhang, Y. Chen, W. Xu, and F. Feng, "Generative AI Application for Building Industry," Oct. 01, 2024, *arXiv*: arXiv:2410.01098. doi: 10.48550/arXiv.2410.01098.

[40] A. Saka et al., "GPT Models in Construction Industry: Opportunities, Limitations, and a Use Case Validation," May 30, 2023, *arXiv*: arXiv:2305.18997. doi: 10.48550/arXiv.2305.18997.

[41] S. A. Prieto, E. T. Mengiste, and B. García de Soto, "Investigating the Use of ChatGPT for the Scheduling of Construction Projects," *Buildings*, vol. 13, no. 4, Art. no. 4, Apr. 2023, doi: 10.3390/buildings13040857.

[42] H. A. Mohamed Hassan, E. Marengo, and W. Nutt, "A BERT-Based Model for Question Answering on Construction Incident Reports," in *Natural Language Processing and Information Systems*, P. Rosso, V. Basile, R. Martínez, E. Métais, and F. Meziane, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, pp. 215–223. doi: 10.1007/978-3-031-08473-7_20.

 doi:10.20944/preprints202510.1060.v1

[43] K. Kim, M. Ivashchenko, P. Ghimire, and P.-C. Huang, "Context-Aware and Adaptive Task Planning for Autonomous Construction Robots Through Llm-Robot Communication," May 14, 2024, *Social Science Research Network, Rochester, NY*: 4827728. doi: 10.2139/ssrn.4827728.

[44] K. Kim, P. Ghimire, and P.-C. Huang, "Framework for LLM-Enabled Construction Robot Task Planning: Knowledge Base Preparation and Robot–LLM Dialogue for Interior Wall Painting," *Robotics*, vol. 14, no. 9, p. 117, Sep. 2025, doi: 10.3390/robotics14090117.

[45] P. Parsafard, O. Elezaj, D. Ekundayo, E. Vakaj, M. Parmar, and M. Ahmad Wani, "Automation in Construction Cost Budgeting using Generative Artificial Intelligence," in *Proceedings of the International Conference on Industrial Engineering and Operations Management*, Dubai, UAE: IEOM Society International, Feb. 2024. doi: 10.46254/AN14.20240466.

[46] C. Gatto, J. Cassandro, C. Mirarchi, and A. Pavan, "LLM based automatic relation between cost domain descriptions and IFC objects," MAR, 2024. Accessed: Feb. 23, 2025. [Online]. Available: https://re.public.polimi.it/handle/11311/1280791

[47] T. Kojima, S. (Shane) Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large Language Models are Zero-Shot Reasoners," *Advances in Neural Information Processing Systems*, vol. 35, pp. 22199–22213, Dec. 2022.

[48] A. Lazaridou, E. Gribovskaya, W. Stokowiec, and N. Grigorev, "Internet-augmented language models through few-shot prompting for open-domain question answering," May 23, 2022, *arXiv*: arXiv:2203.05115. doi: 10.48550/arXiv.2203.05115.

[49] S. Yao et al., "Tree of Thoughts: Deliberate Problem Solving with Large Language Models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 11809–11822, Dec. 2023.

[50] J. Wei et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24824–24837, Dec. 2022.

[51] Y. Nong, M. Aldeen, L. Cheng, H. Hu, F. Chen, and H. Cai, "Chain-of-Thought Prompting of Large Language Models for Discovering and Fixing Software Vulnerabilities," Feb. 27, 2024, *arXiv*: arXiv:2402.17230. doi: 10.48550/arXiv.2402.17230.

[52] T. Brown et al., "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2020, pp. 1877–1901. Accessed: Feb. 23, 2025. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[53] D. Zhou et al., "Least-to-Most Prompting Enables Complex Reasoning in Large Language Models," Apr. 16, 2023, *arXiv*: arXiv:2205.10625. doi: 10.48550/arXiv.2205.10625.

[54] A. Madaan et al., "Self-Refine: Iterative Refinement with Self-Feedback," *Advances in Neural Information Processing Systems*, vol. 36, pp. 46534–46594, Dec. 2023.

[55] H. Conklin, B. Wang, K. Smith, and I. Titov, "Meta-Learning to Compositionally Generalize," Jun. 29, 2021, *arXiv*: arXiv:2106.04252. doi: 10.48550/arXiv.2106.04252.

 doi:10.20944/preprints202510.1060.v1

[56] A. Creswell, M. Shanahan, and I. Higgins, "Selection-Inference: Exploiting Large Language Models for Interpretable Logical Reasoning," May 19, 2022, *arXiv*: arXiv:2205.09712. doi: 10.48550/arXiv.2205.09712.

[57] E. Zelikman, Y. Wu, J. Mu, and N. Goodman, "STaR: Bootstrapping Reasoning With Reasoning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 15476–15488, Dec. 2022.

[58] A. Havrilla et al., "Teaching Large Language Models to Reason with Reinforcement Learning," Mar. 07, 2024, *arXiv*: arXiv:2403.04642. doi: 10.48550/arXiv.2403.04642.

[59] P. Ghimire, "Framework for Integrating Industry Knowledge into a Large Language Model to Assist Construction Cost Estimation," Ph.D., The University of Nebraska - Lincoln, United States -- Nebraska, 2025. Accessed: Oct. 12, 2025. [Online]. Available: https://www.proquest.com/docview/3198872319/abstract/D556793967F749FCPQ/1

[60] R. Islam and O. M. Moushi, *GPT-4o: The Cutting-Edge Advancement in Multimodal LLM*. 2024. doi: 10.36227/techrxiv.171986596.65533294/v1.

[61] H. Touvron et al., "LLaMA: Open and Efficient Foundation Language Models," Feb. 27, 2023, *arXiv*: arXiv:2302.13971. Accessed: Aug. 26, 2023. [Online]. Available: http://arxiv.org/abs/2302.13971

[62] R. Islam and I. Ahmed, "Gemini-the most powerful LLM: Myth or Truth," in *2024 5th Information Communication Technologies Conference (ICTC)*, May 2024, pp. 303–308. doi: 10.1109/ICTC61510.2024.10602253.

[63] R. Kurokawa et al., "Diagnostic performances of Claude 3 Opus and Claude 3.5 Sonnet from patient history and key images in Radiology's 'Diagnosis Please' cases," *Jpn J Radiol*, vol. 42, no. 12, pp. 1399–1402, Dec. 2024, doi: 10.1007/s11604-024-01634-z.

[64] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, Philadelphia, Pennsylvania: Association for Computational Linguistics, 2001, p. 311. doi: 10.3115/1073083.1073135.

[65] N. Diab, "Out of the BLEU: An Error Analysis of Statistical and Neural Machine Translation of WikiHow Articles from English into Arabic," *CDELT Occasional Papers in the Development of English Education*, vol. 75, no. 1, pp. 181–211, Jul. 2021, doi: 10.21608/opde.2021.208437.

[66] S. Lee et al., "A Survey on Evaluation Metrics for Machine Translation," *Mathematics*, vol. 11, no. 4, Art. no. 4, Jan. 2023, doi: 10.3390/math11041006.

[67] K. Ganesan, "ROUGE 2.0: Updated and Improved Measures for Evaluation of Summarization Tasks," Mar. 05, 2018, *arXiv*: arXiv:1803.01937. doi: 10.48550/arXiv.1803.01937.

[68] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," in *Text Summarization Branches Out*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. Accessed: Mar. 02, 2025. [Online]. Available: https://aclanthology.org/W04-1013/

[69] S. Banerjee and A. 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*, J. Goldstein, A. Lavie, C.-Y. Lin, and C. Voss, Eds., Ann Arbor, Michigan: Association

 doi:10.20944/preprints202510.1060.v1

for Computational Linguistics, Jun. 2005, pp. 65–72. Accessed: Mar. 02, 2025. [Online]. Available: https://aclanthology.org/W05-0909/

[70] A. Holzinger, K. Zatloukal, and H. Müller, "Is human oversight to AI systems still possible?," *New Biotechnology*, vol. 85, pp. 59–62, Mar. 2025, doi: 10.1016/j.nbt.2024.12.003.

[71] I. O. Gallegos et al., "Bias and Fairness in Large Language Models: A Survey," *Computational Linguistics*, vol. 50, no. 3, pp. 1097–1179, Sep. 2024, doi: 10.1162/coli_a_00524.

[72] J. Jung, F. Brahman, and Y. Choi, "Trust or Escalate: LLM Judges with Provable Guarantees for Human Agreement," Jul. 25, 2024, *arXiv*: arXiv:2407.18370. doi: 10.48550/arXiv.2407.18370.

[73] *prolego-team/pdd*. (Feb. 19, 2025). Python. prolego-team. Accessed: Mar. 03, 2025. [Online]. Available: https://github.com/prolego-team/pdd

[74] Y. Virk, P. Devanbu, and T. Ahmed, "Enhancing Trust in LLM-Generated Code Summaries with Calibrated Confidence Scores," Dec. 03, 2024, *arXiv*: arXiv:2404.19318. doi: 10.48550/arXiv.2404.19318.

[75] R. Liu, M. Li, S. Zhao, L. Chen, X. Chang, and L. Yao, "In-Context Learning for Zero-shot Medical Report Generation," in *Proceedings of the 32nd ACM International Conference on Multimedia*, in MM '24. New York, NY, USA: Association for Computing Machinery, Oct. 2024, pp. 8721–8730. doi: 10.1145/3664647.3680760.

[76] B. Merkus, "An assessment of Zero-Shot Open Book Question Answering using Large Language Models," Master Thesis, 2023. Accessed: Mar. 02, 2025. [Online]. Available: https://studenttheses.uu.nl/handle/20.500.12932/44625

[77] J. Salvador, N. Bansal, M. Akter, S. Sarkar, A. Das, and S. K. Karmaker, "Benchmarking LLMs on the Semantic Overlap Summarization Task," Feb. 26, 2024, *arXiv*: arXiv:2402.17008. doi: 10.48550/arXiv.2402.17008.

[78] S. Xu et al., "Reasoning before Comparison: LLM-Enhanced Semantic Similarity Metrics for Domain Specialized Text Analysis," Feb. 20, 2024, *arXiv*: arXiv:2402.11398. doi: 10.48550/arXiv.2402.11398.

[79] G. Yang, Y. Zhou, X. Chen, X. Zhang, T. Y. Zhuo, and T. Chen, "Chain-of-Thought in Neural Code Generation: From and for Lightweight Language Models," *IEEE Transactions on Software Engineering*, vol. 50, no. 9, pp. 2437–2457, Sep. 2024, doi: 10.1109/TSE.2024.3440503.

[80] S. Niwattanakul, J. Singthongchai, E. Naenudorn, and S. Wanapu, "Using of Jaccard Coefficient for Keywords Similarity," *Hong Kong*, 2013.

[81] P. Sitikhu, K. Pahi, P. Thapa, and S. Shakya, "A Comparison of Semantic Similarity Methods for Maximum Human Interpretability," in *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, Nov. 2019, pp. 1–4. doi: 10.1109/AITB48515.2019.8947433.