A Hybrid Framework for COSMIC Measurement: Combining Large Language Models with a Rule-Based System

Safae Laqrichi

July 29, 2024

A Hybrid Framework for COSMIC Measurement: Combining Large Language Models with a Rule-Based System

Safae Laqrichi

July 25, 2024

A Hybrid Framework for COSMIC Measurement: Combining Large Language Models with a Rule-Based System

Safae Laqrichi

Arts et Métiers Campus of Rabat, Technopolis Rabat-Shore, Rabat 11103, Maroc

Abstract
Accurate Functional Size Measurement (FSM) is crucial for effective project management and resource allocation in software development. The COSMIC FSM provides a consistent and standardized method, allowing for objective comparisons and accurate estimation of effort. However, manual FSM is often time-consuming, error-prone, and subject to assumptions and human bias. Automating this process can greatly benefit organizations but introduces challenges associated with handling requirements expressed in Natural Language (NL). In this paper, we explore the integration of Large Language Models (LLM) with rule-based systems to enhance the automation of the COSMIC Function Point Measurement. Our hybrid framework combines the contextual understanding and adaptability of LLMs, such as GPT-4, with the precision and consistency of rule-based engines crucial for tasks requiring strict adherence to rules and regulations. This approach enables a more robust analysis, processing of NL requirements, and precise application of the COSMIC measurement method. The effectiveness and feasibility of this approach are demonstrated through a series of experiments conducted on COSMIC public case studies. Our results show a promising F1-score of 0.95 in identifying COSMIC key components using GPT-4, an F1-score of 0.97 in deducing data movements using the rule-based system, and a global error rate of only 6.6% in measuring COSMIC function points using the integrated framework. These results underscore the potential for more reliable and efficient system analysis through the synergistic capabilities of LLMs and rule-based systems.

Keywords
COSMIC FSM, Automation, LLM, GPT-4, ChatGPT, Rule-based-system, Prompting

1. Introduction

Software measurement plays a crucial role in software development by providing quantitative insights into various aspects of software products, such as size, complexity, and effort estimation. Accurate measurement of software requirements is particularly important as it forms the foundation for successful project planning and management. The COSMIC function point (CFP) method is a widely accepted approach for measuring software size and estimating effort based on functional requirements analysis [1]. However, the process of manually measuring software requirements can be time-consuming, error-prone, and subject to interpretation biases [2].

To address these challenges, there is growing interest in leveraging Large Language Models

IWSM-Mensura, September 30–october 04, 2024, Montréal, Canada
safae.laqrichi@estimancy.com (S. Laqrichi)
0000-0002-0877-7063 (S. Laqrichi)

(LLMs) like GPT-3 [3] and GPT-4 [4], which use powerful NLP techniques and deep learning algorithms. They have demonstrated remarkable capabilities in natural language processing (NLP) and generation, raising the possibility of automating the measurement of natural language (NL) software requirements using the COSMIC function point method.

In this paper, we propose an approach that utilizes LLM for automating the measurement of NL software requirements in the COSMIC function point method. Our objective is to leverage the language understanding and generation capabilities of these models to analyze and measure the functional aspects of software requirements, thereby streamlining the measurement process and reducing the manual effort involved.

Prior research has explored various techniques for automating software requirement measurement, including rule-based approaches, machine learning-based models, and ontology [5]. However, the recent advancements in LLMs offer an exciting opportunity to overcome the limitations of previous approaches and improve the accuracy and efficiency of software requirement measurement.

The remainder of this paper is organized as follows. Section 2 provides background information on the COSMIC function point method and reviews the related work on automating COSMIC FSM for NL requirements. Section 3 describes our proposed framework using LLMs and outlines the methodology involved. Section 4 presents the results of our experiments and evaluates the performance of the applied LLM in measuring NL software requirements. Finally, section 5 concludes the paper and outlines potential avenues for future research.

2. Related work

2.1. COSMIC Function point method

Numerous studies have demonstrated the effectiveness of the COSMIC method as a functional size measurement (FSM) approach. The COSMIC method's has gained global adoption due to its flexibility allowing to measure any type of software, including web applications, agile development contexts, and large-scale software projects.

The fundamental concept underlying the COSMIC method is that, for many types of software, a significant portion of development efforts is dedicated to handling data movements between persistent storage and users. Therefore, the total number of these data movements can provide valuable insight into the size of the system [6, 7].

In COSMIC FSM, data movement is the Base Functional Component which moves a single data group. To measure these data movements, three elements need to be identified. A Data Group is a distinct, non-empty, non-ordered and non redundant set of data attributes where each included data attribute describes a complementary aspect of the same one object of interest. An object of interest is any 'thing' that is identified from the point of view of the FUR about which the software is required to process and/or store data. A Data Attribute is smallest parcel of information, within an identified data group, carrying a meaning from the perspective of the software's FUR.

Data movements are categorized based on their direction and purpose within a system: An Entry (E) transfers data from a user to the necessary functional process. An Exit (X) sends data from a process back to the user. A Read (R) retrieves data from persistent storage for use within a process, and a Write (W) stores data from a process into persistent storage.

2.2. Software requirements in Natural Language

The majority of software requirement specifications are written in NL, including initial conceptions and requests for proposals (RFPs). However, NL is inherently ambiguous, leading to messy software requirements specification (SRS). There is a tradeoff between using NL or a mathematics-based formal language (MBFL). NL allows for easy understanding by stakeholders, although interpretations may vary. MBFL is unambiguous but may lack writers and understanding among stakeholders. Research in requirements engineering (RE) aims to address the ambiguity of RSs by promoting MBFLs and improving accessibility. NL SRSs are inevitable as they bridge the informal ideas of software development with formal coding. The transition from informal to formal occurs during the requirements engineering process. NL remains essential, even if only during the initial conception [8]. However, requirements that are expressed in NL and in a free-form manner are susceptible to being imprecise, long and incomplete [9]. These characteristics pose significant challenges for efficient and standard measurement, particularly in contexts like COSMIC function point analysis, which requires a high degree of clarity, completeness, unambiguity and a high level of granularity.

2.3. Literature review of COSMIC FSM automation studies

The automation of the COSMIC method has sparked a lot of interest over the past decades. Several studies have investigated both structured FURs, such as UML diagrams, and unstructured FURs, like free-form text or NL.

To the best of our knowledge, there is limited academic research that has been conducted on automating functional measurement using NL specifications. Only a few studies, such as those by Les [10, 11, 12, 13, 14, 9], have addressed this topic. In [10], Hussain, Ishrar, and Ormandjieva proposed an innovative concept based on a supervised approach to word processing. Their method involved identifying parts of speech and associating them with specific features to automate functional measurement from informal textual specifications. However, their method had limitations such as not considering context and assuming consistent language styles in specifications.

In his thesis [11], Hussain proposed a method combining machine learning and NLP. The method involved preprocessing the data by tokenizing, separating sentences, tagging part of speech, and extracting names. The machine learning algorithm was used for classifying input data into types of data movement. While the results were satisfactory, the use of tokenization without considering context was acknowledged as a limitation.

In a different study [12], authors presented an NLP method for functional measurement based on the extraction of part of speech. They focused on specifications written in SBVR (Semantic Business Vocabulary and Rules) syntax and used a parsing method to extract object classes. However, limitations were observed due to the restrictions of the parsing method and the fact that SBVR specifications were not common.

Another study [13] proposed a contextual NLP model (CNLP) and a flexible NL representation called "two level grammar (TLG)." However, the method has not been validated yet.

Tripathy et al. [14] presented a technique for transforming NL specifications into object-oriented structures using the part of speech (POS) NLP methodology. They aimed to generate a tree structure and a conceptual model for function point counting. The authors acknowledged the need to optimize the parsing step and expressed the intention to test the model on multiple use cases.

Bagrayanik et al. [5] designed a newly requirements engineering ontology and develop a method to automatically measure the software size in COSMIC FP using the ontology. The method has been validated using real projects conducted within the ICT department of a leading telecommunications provider in Turkey. The results demonstrated a consistent agreement between manual and automated measurement outcomes. One limitations of this study is the need to specify requirements in terms of the proposed requirements ontology. Another limitation arises from the fact that the proposed method exclusively supports the COSMIC Function Point method as the functional size measurement methodology. This limitation is particularly unfavorable for companies that adopt other FSM methods, such as IFPUG.

A more recent research study performed by Erdin, Colin, and Alain in [9] introduced Scope-Master®, a commercial tool automating COSMIC FSM on requirements in user story format. The tool's methodology leverages NLP and pattern matching techniques. It focuses on identifying the "Subject verb object" structure within requirements to extract the necessary measurement elements. For instance, in a user story like "As a user, I want to display orders," the subject "user" is considered a candidate for the Functional User, the object "orders" is a candidate for an Object of Interest, and the verb "display" corresponds to one of the functions from the CRUDL (Create, Read, Update, Delete, List) set. The specific details of how ScopeMaster® performs these techniques are proprietary and subject to a pending patent application. Currently, examples handled by the tool are only in a US format; there is no experimentation conducted on free-form text FUR. Another mentioned limitation pertains to the non-customizability of the used ontology. For instance, the list of recognized verbs is fixed and identical across all projects. However, there may arise a need for nuances or combinations of verbs to accommodate diverse development contexts.

3. Capabilities of LLMs in applying COSMIC Function Point principles

LLM [3, 15] such as GPT-4 [16], PaLM [17] and LLaMA [18], have shown impressive performance in various NLP tasks and gained significant attention from academia and industry. These models leverage extensive pre-training on massive datasets and reinforcement learning from human feedback (RLHF) [19] to excel in language understanding, generation, interaction, and reasoning.

LLMs typically refer to Transformer-based models with hundreds of billions of parameters. They use multiple layers of attention mechanisms, known as multi-head attention, in a deep neural network architecture. LLMs implement similar Transformer architectures and pre-training objectives as smaller models like BERT [20]. Scaling involves increasing the number of parameters, training data, and computational resources to enhance performance [21].

LLMs exhibit emergent abilities, which are not present in smaller models but arise in larger ones [21]. Typical emergent abilities include in-context learning [3], instruction following [22], and chain-of-thought prompting [23]. These abilities open up possibilities for building advanced AI systems [24].

For our research, we explored instruction-following LLMs. These models understand a wide range of user instructions, also called prompts, and produce contextually and grammatically correct responses. Due to their generalization ability, they perform strongly on unseen tasks presented as instructions without needing explicit examples [25, 26].

Applying the COSMIC method requires high-level reasoning and strict adherence to specific rules. Precision in applying these rules is imperative; failure can result in approximate rather than accurate measurements.

Using rule-based prompts, chain-of-thought, or least-to-most prompting [27] can guide the LLM to generate responses within a specific framework of rules. This is effective for systems with easily stated rules, but less so for complex tasks with hierarchical rule structures, like COSMIC measurement.

An alternative approach integrates a rule-based system with LLM prompting. The rule-based component refines the LLM's output to ensure adherence to predefined rules. This hybrid model leverages the LLM's advanced NL understanding and generation capabilities while ensuring compliance with intricate rules, advantageous in scenarios demanding high precision and strict rule compliance.

4. Integrating LLMs and Rule-Based System for Automated COSMIC Measurement

Our framework represents an initial attempt to harness the capabilities of LLMs combined with rule-based systems for COSMIC measurement. This preliminary research is based on several assumptions: (i) the software is categorized as a new development project, (ii) the software architecture follows a single-layer structure, eliminating the need to identify multiple layers, and (iii) the scope of measurement is clearly defined, with the requirements document exclusively describing the functionalities within this scope.

Consequently, our framework for automating COSMIC measurement will bypass the steps of determining the purpose and scope of measurement, as well as the identification of architectural layers.

Our framework is structured into two major stages as illustrated in Figure 1: a) identification of COSMIC key components using LLM, and b) deduction of data movements using rule-based system.

LLMs are utilized for NL understanding and processing, and information extraction tasks, required for step a) of the framework. The highly rule-based task of deducing data movements in step c) is managed using a rule-based system. This approach ensures that the analytical and interpretative tasks are effectively addressed by leveraging both advanced machine learning techniques and structured logical processes.

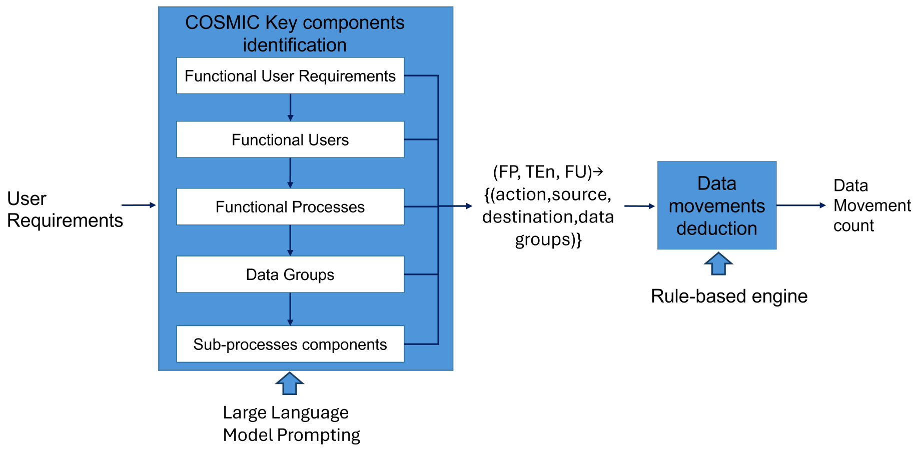
COSMIC Key components identification
Functional User Requirements
Functional Users
Functional Processes
Data Groups
Sub-processes components
User Requirements
Large Language Model Prompting
(FP, TEn, FU)→ {(action,source, destination,data groups)}
Data movements deduction
Rule-based engine
Data Movement count

Figure 1: Hybrid Framework for COSMIC Measurement integrating LLMs and Rule-Based System

4.1. COSMIC key components identification using LLM

To identify these COSMIC key components, our methodology, depicted in Figure 1, requires executing various steps of the COSMIC process. This methodology comprises five distinct steps: identifying Functional User Requirements (FUR), Functional Users, Functional Processes, Data Groups, and finally, the identification of Sub-process Components. The latter step, not explicitly outlined in the official manuals, aims to properly partition the functional process into sub-processes while adhering to the granularity required for COSMIC measurement.

Furthermore, we have identified additional critical components essential for the final phase of identification and classification of data movements, such as action verbs, destinations, and sources of data. Each component plays a vital role in both understanding and quantifying data movement. Definitions of these additional key components are :

- Action verb: This specifies the operations performed on the data, such as entering, exiting, reading, writing, sending, or receiving, detailing the manner in which the data is transferred or altered.
- Destination: This indicates where the data is directed or the endpoint of the action. The destination may be internal (within the system) or external (to another system or a user).
- Source: This identifies the origin of the data, which may also be internal (originating within the system) or external (coming from another system or a user).

The integration of both the source and the destination is critical as they delineate the trajectory of the data movement, essential for gauging the type of each interaction within the system. The source is particularly significant in establishing the starting point of the data flow, which is vital for analyzing data transfer across system boundaries, especially in complex systems with numerous interacting external applications or subsystems.

4.1.1. Definitions and rules adaptations for improved linearity

The process described in the official manual appears linear at first glance. However, some rules establish a component's definition based on subsequent components in the procedure, introducing an inherent non-linearity. For instance, the rule 7 of the official COSMIC manual for functional users identification states that "all functional users that trigger, provide information to, or receive information from functional processes in the FUR of the software within the scope of the FSM shall be identified". Yet, these functional processes themselves are only determined at a later stage in the procedure. Another example is Rule 10 and its Guidance, which specify that to identify a functional process, one must identify the "Triggering Event" that causes a functional user to generate a data group, thereby initiating a sequence of data movements. At this stage, the definitions of 'data group' and 'data movement' remain unspecified, with their detailed identification addressed later in the COSMIC process. This observation highlights the need to adapt definitions to ensure better linearity of the process and facilitate the automation of the COSMIC methodology using LLM prompting.

These insights have guided our approach to ensure that each step logically follows the previous one, enhancing the clarity and effectiveness of the COSMIC methodology application. Below, we provide the definitions and rules used at each step.

- Step 1: Identification of Functional User Requirements (FUR)

  The LLM is used to examine user requirements documents to identify FUR, suggest improvements, and ensure they are measurable in FSM methods. If the requirements are in free-form text, the LLM breaks them down into FUR. In agile developments, these FUR can be transformed by the LLM into User Stories (US) efficiently.

- Step 2: Identification of Functional Users

  Functional users, as defined in the COSMIC methodology, are identified based on their role as senders or recipients of data. In this definition, we do not reference functional processes at this step, as they have not yet been defined.

- Step 3: Identification of Functional Processes

  Functional processes are identified by recognizing triggering events, the functional user responding to these events, and the data attributes sets initiated representing the Triggering Entry (TEn). In this step, data groups are not referenced, as they have not yet been defined, "data attributes sets" term are utilized rather.

- Step 4: Identification of Data Groups

  Data Groups is a critical concept in many functional sizing methods, including COSMIC. In COSMIC method, data groups are identified by pinpointing objects of interest and their data attributes sets, which include frequency occurrence and identification keys. Data representing control commands are not data groups. Data attributes sets that describes different objects of interest represent different data groups. For data attributes sets that describe the same objects of interest, to identify if they represent distinct data groups, a straightforward rule is applied : if two data attribute sets differ in either their key attribute or their frequency of occurrence, they are considered separate data groups.

- Step 5: Identification of Sub-process Components

  This step involves partitioning functional processes into steps then identifying sub-processes to meet the granularity required for COSMIC measurement and extracting complementary key components for identifying and classifying data movements.

4.1.2. Prompt Design and Optimization

The prompts used at each stage of the methodology have been carefully designed, tested, and optimized to fully leverage the capabilities of the LLM through prompt engineering techniques. Table 1 synthesizes the components of these prompts. They may range from simply defining the key component to be identified and specifying the required task, to outlining detailed steps necessary for identifying the target component. The prompts were crafted based on COSMIC method rules from the official manual, and the 'Rule Reference' column in the table indicates the specific rules (R) and guidelines (GR) from the manual applied at each step.

Here is the structured approach for crafting effective prompts used in our approach :

- Introduction and Context: Introduce the task's background and contextual details, including specific roles or expertise required (e.g., a software measurement expert for COSMIC tasks).
- Task Description: Clearly describe the objective or task to be achieved.
- Step-by-Step Instructions: Break down the task into sequential stages, providing clear definitions and instructions to guide the language model or tool.
- Expected Output Format: Define the desired structure or format of the output, specifying essential attributes or information.
- Example or Scenario: Utilize few-shot learning techniques by presenting an example or scenario that illustrates the task or problem, thereby enhancing the quality of responses.

Example of a prompt for identifying functional processes:

Role: Your are a software measurement expert specializing in the COSMIC method for functional size measurement.
Task Description: your task involves identifying functional processes from the provided Functional User Requirements (FUR).
Step-by-Step Instructions: The process involves the following steps:

1. Identify the Triggering Events: These are distinct events in the world of the functional users that the software being measured must respond to.

2. Identify the implied Functional User types: these are functional users of the software that are likely to respond to each triggering event.

3. Identify the Triggering Entry: these are the data that each functional user may generate in response to the event.

4. Identify the Functional Process: which is initiated by each triggering data. This process comprises all operations necessary to fulfill its FUR, addressing all potential responses to the triggering event.

Expected Output Format: Present the result in this python dictionary format :
functional-process-format = { "Add-Professor":
{ "Triggering Events": "New employement"
"Functional User" : "Administator",
"Triggering Entry" : "Professor details"}
Example: "As a student, I want be able to check my courses schedule."
Identified functional process :
{ "Check-schedule":
{ "Triggering Events": "Student request"
"Functional User" : Student",
"Triggering Entry" : "Student ID"}
Here is the FUR :

Table 1
Overview of Functional Components

Steps	Prompt Elements	Rules Reference
Functional User Requirements	Use: FUR Definition, examples of FUR and NFR	R3
Functional Users	Use FU definition : Sender or recipient of data	R7, GR7
Functional Processes	Identify: Triggering Event, Functional User responding to, Data attributes set(s) initiated	R10
Data Groups	Identify: Objects Of Interest, Data attributes sets(freq, Id Key)	R11, GR13-14, GR16-19
Sub-Processes	Identify: functional steps, use sub-processes definition	

4.1.3. Adjusting Gpt-4 sampling temperature

Several inference hyperparameters can be adjusted to modify the Gpt-4 outputs at runtime. These include sampling temperature, Top-p (Nucleus Sampling), and frequency penalty [28].

The most critical parameter in controlling the behavior of LLMs during our process is the sampling temperature, which affects the randomness and the variability of the model's output. Lower temperatures exploit more probable solutions, while higher temperatures explore the solution space more broadly [29]. For our measurement task, preliminary tests indicated that lower temperatures (e.g., 0.2 to 0.3) make the output more deterministic and focused, leading to consistent and accurate identification of COSMIC key components. Conversely, higher temperatures (e.g., 0.7 to 1.0) introduce more variability and creativity, which can sometimes reduce precision. Therefore, we fixed the sampling temperature at 0.2 to ensure that GPT-4's responses adhered closely to the COSMIC methodology's rules and guidelines. Regarding the Top-p value and frequency penalty hyperparameters, they are maintained at default settings, typically at 1 for Top-p to allow for a full spectrum of probable outcomes, and 0 for the frequency penalty to avoid artificially inhibiting the model's NL generation capabilities. These settings help balance the model's creativity with the need for accuracy and relevance in its outputs.

4.2. Rule-based system for data movements deduction

Rule-based systems are an important class of expert systems. They are fundamentally composed of a series of IF-THEN rules, which can serve various domains, including decision support and predictive decision-making across real-world applications. These rules are generally constructed through the use of expert knowledge or through learning from real data. These rules can represent the specialised knowledge of a task area.

The table below provides a detailed overview of the COSMIC measurement rules implemented within our rule-based engine. These rules are derived and synthesized from the official COSMIC measurement manual, further enriched by the extensive practical experience I have gained through applying COSMIC Function Points across a variety of projects during my tenure at Estimancy. Each rule is precisely defined to ensure the accurate enumeration and classification of data movements, tailored to specific actions and data interactions. Organized for clarity, the table delineates the conditions under which each rule applies, the actions involved, the sources or destinations impacted, the resulting data movement type, and the counting methodology employed. This integration of established guidelines with empirical insights not only solidifies the theoretical basis of our rules but also confirms their practical efficacy, providing a robust framework for accurately quantifying software size and complexity in diverse environments.

RU1: A triggering entry (TEn) must trigger only one functional process, for the set of identified functional process inspect the triplets (FP, TEn, FU), IF there is many triplets with the same TEn THEN merge them into one FP.

RU2: For each functional process, inspect all quadruplets (action verb, data groups, destination, source). IF there are repetitions, THEN remove them from the dataset to ensure uniqueness of data movements.

RU3: For each quadruplet (action verb, data groups, destination, source), deduce the data movements and their types using the table 2.

Rule ref	Action type	Source	Destination	Data Movement type
RU3-1	Input	FU	Internal storage	Entry
RU3-2	Retrieve	Internal storage	Software	Read
RU3-3	Store	Software	Internal storage	Write
RU3-4	Output	Software	FU	Exit
RU3-5	Retrieve	External Application	Software	Exit+Entry

Table 2
Implemented COSMIC Measurement Rules

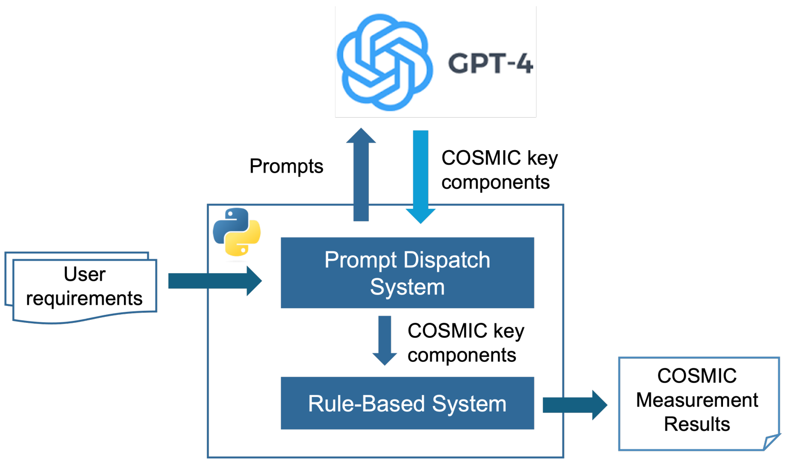

Figure 2: Architecture of the Hybrid COSMIC Measurement Framework

RU4: For each functional process, verify the presence of an Entry data movement. IF no Entry data movement is identified, THEN include an additional Entry to the total count of identified data movements.

4.3. Implementation details

The architecture of our hybrid framework, depicted in Figure 2, is implemented in Python. The "Prompt Dispatch System" handles the transformation of user requirements into structured prompts that are processed by GPT-4. The outputs from GPT-4, which include identified COSMIC key components are then relayed to the Rule-Based System. This architecture offers flexibility by allowing easy replacement of the LLM, should the need arise, ensuring adaptability to new developments and improvements in language model technology.

5. Evaluating the approach through public case studies

5.1. Case studies description

These Case Studies can be accessed in the Case Study section of the COSMIC website [30]. Case studies selected for this experiment are examples from the Real-Time and Business Applications.

Table 3
COSMIC case studies description

Domain	Software name	Req format	#FP	CFP
Business Application	Course Registration ('C-REG') System	FFT	19	102
	RestoSys	FFT/ US/ UML Use Cases	29	119
Real Time	Rice Cooker	FFT/ diagrams	3	12

Functional requirements of software presented in case studies are documented in various format : free-form text, user stories and UML Use Cases. Descriptions of case studies are presented in the table 3.

As our approach focused in NL processing leveraging LLMs, our experiments will be held on the free-form text (FFT) and user stories (US). In order to be able to compare the performance of the proposed approach on generating US from FFT, a manual rewriting task has been undertaken on functional requirements originally documented in FFT to convert them to US format. For the purpose of comparison and testing, the datasets used in this study can be accessed my Git repository [31].

5.2. Evaluation Method

The outputs of our automated system are of two distinct types: textual components such as FUR, FP, FU, DG and TE and categorical components such as action verb type, source, destination, data movement type and data movement number. Each type requires a different evaluation approach to accurately measure the performance of our COSMIC Function Point method automation.

5.2.1. Evaluation of categorical components

For components classified into predefined categories by our automated system, we employed standard classification metrics such as precision, recall, and F1 score to evaluate performance.

5.2.2. Evaluation of textual components

For the textual component, we used a tailored evaluation strategy that integrates precision, recall, and semantic similarity metrics. This section details the methodology used to calculate True Positives (TP), False Positives (FP), and False Negatives (FN) by leveraging semantic similarity provided by GPT-4 embedding. The following steps outline the evaluation process in detail:

- Data Preparation: We defined two sets of components: the reference set (ground truth) with manually identified COSMIC key components and the output set generated by our approach.
- Embedding Generation: Using the OpenAI GPT-4 API, we generated high-dimensional vector embeddings for each component in both sets, enabling cosine similarity calculations.

- Similarity Calculation: Cosine similarity measured the semantic equivalence between reference and output components. We created a similarity matrix where each element represents the cosine similarity score between a reference component and an LLM output component.
- Threshold Determination: A similarity threshold of 0.8 was set based on preliminary experiments. Components with a cosine similarity score equal to or higher than this threshold were considered equivalent.
- Matching and Evaluation: For each reference component, we identified the output component with the highest similarity score above the threshold. Components were classified as follows:
  - True Positives (TP): Reference components with a corresponding output component above the threshold.
  - False Positives (FP): Output components not matching any reference component.
  - False Negatives (FN): Reference components without a corresponding output component above the threshold.

This evaluation method ensures a robust assessment of our approach performance, capturing both exact matches and semantically equivalent variations in the identified COSMIC key components.

By using this dual evaluation methodology, we comprehensively assess our approach's performance, capturing both exact matches and semantically equivalent variations in the identified COSMIC key components for textual components and categorical components.

5.3. Results of framework experiments on case studies

To evaluate the effectiveness of our hybrid framework for COSMIC measurement, we conducted two separate experiments. The first experiment assessed the performance of the LLM in identifying COSMIC key components which are textual. The second experiment evaluated the rule-based system's accuracy in deducing data movements.

5.3.1. COSMIC key component identification Using LLM

When applying the designed prompt-based strategy on case studies to identify the COSMIC key components, the results presented in 4 indicates favorable F1-scores for the identification of key components using LLM in each case study.

The 'C-REG' System and RestoSys demonstrate high F1 scores (close to or at 1.00) across almost all categories, indicating the LLM-based approach is highly effective in accurately identifying key components for business application type.

The Rice Cooker case study shows slightly lower F1 scores, particularly for Functional Processes (0.86), Triggering Events (0.86), Data Groups (0.88), and sub-process components (0.92). This suggests some challenges in identifying these components in more complex or nuanced scenarios related to real time software.

Table 4
COSMIC key component identification using LLM on case studies : F1-Score

Software	FP	TEn	FU	DG	SP	Action	Source	Destination	Average
‘C-REG’ System	1	0.92	1	0.91	1	1	1	1	0.98
RestoSys	1	0.92	1	1	1	0.94	1	1	0.98
Rice cooker	0.86	0.86	1	0.88	0.92	0.92	0.92	0.92	0.91
F1-Scores Averages	0.95	0.90	1.00	0.93	0.96	0.97	0.97	0.97	0.95

Table 5
COSMIC data movements deduction using rule-based system : F1-Score

Software name	Entry	Read	Write	Exit	Average
C-REG	1.00	1.00	1.00	1.00	1
RestoSys	0.96	0.96	1.00	0.96	0.97
Rice cooker	0.92	0.94	0.94	1.00	0.95
Metric averages	0.96	0.98	0.98	0.98	0.97

The overall average F1-scores for all components remain high, with values ranging from 0.90 to 1.00 and a global F1-score of 0.95 highlighting the effectiveness of using LLM for COSMIC key component identification across different case studies.

5.3.2. COSMIC Data Movements Deduction Using Rule-Based System

The table 5 presents the F1 scores for the classification of sub processes using rule-based system in each case study. This table includes the scores for the four types of data movements: Entry, Read, Write, and Exit, along with an average score for each software system and overall metric averages across all systems.

The results demonstrate high accuracy for the C-REG system and RestoSys, with slightly lower performance for the Rice cooker case study. This discrepancy may be attributed to the vocabulary and context specific to the real-time domain, which differ from the business application vocabulary on the basis of which the majority of the rules were built.

The overall metric average across all systems and data movement types is 0.97, indicating that the rule-based system is highly effective and reliable in classifying data movements across different case studies. However, further improvements may be required for real time software.

The resulting total COSMIC Function Point (CFP) sizes calculated using our global framework for the case studies are presented in Table 6. They exhibit an average error of 6.6%, indicating that our automated approach yields reasonably accurate estimations of the functional size of software.

5.4. Conclusion

Validated across three case studies, our framework demonstrated significant accuracy and efficiency, substantially reducing the time, effort, human error, and subjective bias associated

Table 6
Total COSMIC Function Point (CFP) sizes counting: Error

Software name	Actual CFP	Measured CFP	Error
C-REG	102	102	0
RestoSys	119	114	0.04
Rice cooker	12	10	0.16
Metric average			0.066

with manual measurements. This leads to more reliable and objective estimates of software size.

Our framework is language-agnostic, leveraging LLMs like GPT-4, which are trained on extensive multilingual datasets from the internet. This capability allows it to process texts in multiple languages, such as English, Spanish, French, and more, enhancing its adaptability for various software development projects regardless of the language used in requirements specifications.

Finally, our LLM and rule -based framework offers high customizability. In fact, in addition COSMIC Function Point method, it could be adapted to support various FSM methods, including IFPUG and SiFP (Simple Function Point), as well as approximation methods like E&QFP (Early and Quick Fuction Point). Indeed, initial experiments have been carried out on some examples for SiFP measurement and have shown a promising level of accuracy.

5.5. Threats to validity

In our study, the initial validation of the framework involved experiments across three distinct case studies. While these provided valuable initial insights, the limited number of case studies is generally insufficient for robust scientific validation. This limitation is particularly significant given the broad spectrum of software types and complexities, such as Artificial Intelligence (AI) software, which exhibit unique behaviors and requirements not covered by a small, homogeneous set of case studies. Such limitations in scale and complexity raise concerns about the external validity of our findings, questioning the generalizability of the results across varied software types and more complex system architectures.

Our research has primarily focused on measuring the functional size of new software development projects, inadvertently overlooking the distinct challenges posed by software evolution or maintenance stages. This limitation represents a significant gap, as these project types involve unique complexities and lifecycle considerations that could crucially influence both the applicability and the effectiveness of our proposed framework. Addressing these oversight will be crucial for enhancing our framework's relevance and utility across different software project types.

Additionally, our research has predominantly focused on measuring the functional size of new development projects, not tackling the distinct challenges posed by software evolution or maintenance stages. This limitation represents a significant gap, as these project types involve unique complexities and lifecycle considerations that could crucially influence both the applicability and the effectiveness of our proposed framework.

Furthermore, the internal validity of our study faces challenges due to potential misconfigurations or biases in setting hyperparameters like sampling temperature, which could significantly distort outcomes. Construct validity is also critical, as the reliance solely on F1-scores and error rates may not fully capture the effectiveness of our framework. Consistency and clarity in the operational definitions of measured constructs are essential to ensure accurate reflection of the framework's capabilities.

6. Conclusion and future work

In summary, our exploration into the integration of Large Language Models (LLMs) like GPT-4 with rule-based systems for automating COSMIC Function Point Measurement has shown promising results. Our hybrid framework effectively leverages the contextual understanding of LLMs alongside the precision of rule-based engines, enhancing the robustness and accuracy of software measurement in diverse settings. Experiments conducted across three case studies yielded an F1-score of 0.9 in identifying COSMIC key components and 0.97 in deducing data movements, with a global error rate of 6.6% in COSMIC function point measurements.

However, the limited number of case studies presents a significant threat to the validity of our findings, highlighting the need for broader testing across various software types and complexities to ensure the generalizability and applicability of our framework.

Future development should focus on enhancing the framework's performance in real-time software contexts. This can be achieved by creating more specialized prompts and rules tailored to the specific vocabulary and operational characteristics of real-time systems, enabling the framework to deliver more precise and contextually relevant measurements. Further improvement involves enriching prompts with examples from real-time applications, which would help LLMs better understand and process the unique requirements and scenarios of such environments. Insights for these enhancements can be drawn from studies like those by [32, 33], which offer valuable perspectives on automating COSMIC measurement for real-time embedded software.

Experiments with our framework reveal a slightly lower performance in identifying data groups, suggesting that the Gpt-4 lacks the necessary knowledge to efficiently extract data models from requirements. To enhance the LLM's specialization in this task, fine-tuning may be necessary. This process aims to improve the model's performance and accuracy for specific tasks by training it on a domain-specific dataset [22]. Future work will involve developing a dataset specifically for data model or UML extraction from NL requirements. This dataset will be used to fine-tune the LLM, enhancing its ability to accurately analyze and generate data models from requirements.

For this study, we utilized a non-open source GPT-4 model through the OpenAI API, which is a cloud-based service. Hence, it is crucial to consider the privacy implications associated with using this service. To address these concerns, our ongoing research includes exploring and implementing alternative open-source LLMs that can be deployed on private servers, such as Llama2 [34] and Llama3 [35].

References

[1] A. J. Albrecht, Measuring application development productivity, volume 10, 1979, pp. 83–92. URL: http://www.bfpug.com.br/Artigos/Albrecht/MeasuringApplicationDevelopmentProductivity.pdf.

[2] H. Soubra, Y. Abufrikha, A. Abran, Towards universal COSMIC size measurement automation, CEUR-WS, Mexico City, Mexico, 2020. URL: http://ceur-ws.org/Vol-2725/paper2.pdf, issue: 2725 Num Pages: 15 Number: 2725.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language Models are Few-Shot Learners, 2020. URL: http://arxiv.org/abs/2005.14165. doi:`10.48550/arXiv.2005.14165`, arXiv:2005.14165 [cs].

[4] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, J. H. Kirchner, J. Kiros, M. Knight, D. Kokotajlo, Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O'Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. d. A. B. Peres, M. Petrov, H. P. d. O. Pinto, Michael, Pokorny, M. Pokrass, V. H. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. B. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley,

J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. J. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, B. Zoph, GPT-4 Technical Report, 2024. URL: http://arxiv.org/abs/2303.08774. doi:10.48550/arXiv.2303.08774, arXiv:2303.08774 [cs].

[5] S. Bagriyanik, A. Karahoca, Automated COSMIC Function Point Measurement Using a Requirements Engineering Ontology, Inf. Softw. Technol. 72 (2016) 189–203. URL: http://dx.doi.org/10.1016/j.infsof.2015.12.011. doi:10.1016/j.infsof.2015.12.011.

[6] A. Abran, P. Fagg, A. Lesterhuis, The COSMIC Functional Size Measurement Method Manual v5.0, ???? URL: https://cosmic-sizing.org/publications/combination-of-parts-v5-0/.

[7] S. Di Martino, F. Ferrucci, C. Gravino, F. Sarro, Web Effort Estimation: Function Point Analysis vs. COSMIC, Information and Software Technology 72 (2016) 90–109. URL: https://www.sciencedirect.com/science/article/pii/S0950584915002037. doi:10.1016/j.infsof.2015.12.001.

[8] D. M. Berry, Ambiguity in Natural Language Requirements Documents, in: B. Paech, C. Martell (Eds.), Innovations for Requirement Analysis. From Stakeholders' Needs to Formal Designs, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2008, pp. 1–7. doi:10.1007/978-3-540-89778-1_1.

[9] E. Ungan, C. Hammond, A. Abran, Automated COSMIC measurement and requirement quality improvement through ScopeMaster® tool, CEUR-WS, Beijing, China, 2018. URL: http://ceur-ws.org/Vol-2207/IWSM_Mensura_2018_paper_11.pdf, issue: 2207 Num Pages: 13 Number: 2207.

[10] I. Hussain, L. Kosseim, O. Ormandjieva, Approximation of COSMIC functional size to support early effort estimation in Agile, Data & Knowledge Engineering 85 (2013) 2–14. URL: https://linkinghub.elsevier.com/retrieve/pii/S0169023X1200064X. doi:10.1016/j.datak.2012.06.005.

[11] H. Hussain, Linguistic Approaches for Early Measurement of Functional Size from Software Requirements, 2014. URL: https://www.semanticscholar.org/paper/Linguistic-Approaches-for-Early-Measurement-of-Size-Hussain/b524b694d142f411fa324cd0b7d4bec5291a34a3.

[12] M. Mohanan, P. Samuel, Software Requirement Elicitation Using Natural Language Processing, 2016, pp. 197–208. doi:10.1007/978-3-319-28031-8_17.

[13] B.-S. Lee, B. Bryant, Automated conversion from requirements documentation to an object-oriented formal specification language, 2002. doi:10.1145/508791.508972, journal Abbreviation: Proceedings of the ACM Symposium on Applied Computing Pages: 936 Publication Title: Proceedings of the ACM Symposium on Applied Computing.

[14] A. Tripathy, A. Agrawal, S. Rath, Requirement Analysis using Natural Language Processing, 2014.

[15] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, L. Zettlemoyer, OPT: Open Pre-trained Transformer Language Models, 2022. URL: http://arxiv.org/abs/2205.01068. doi:10.48550/arXiv.2205.01068, arXiv:2205.01068

[cs].

[16] OpenAI, GPT-4 Technical Report, 2023. URL: http://arxiv.org/abs/2303.08774. doi:10.48550/arXiv.2303.08774, arXiv:2303.08774 [cs].

[17] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, N. Fiedel, PaLM: Scaling Language Modeling with Pathways, 2022. URL: http://arxiv.org/abs/2204.02311. doi:10.48550/arXiv.2204.02311, arXiv:2204.02311 [cs].

[18] L. Zhao, W. Alhoshan, A. Ferrari, K. Letsholo, M. Ajagbe, R. Batista-Navarro, E.-V. Chioasca, Natural Language Processing (NLP) for Requirements Engineering: A Systematic Mapping Study https://arxiv.org/abs/2004.01099 (2020). doi:10.1145/3444689.

[19] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, R. Lowe, Training language models to follow instructions with human feedback, 2022. URL: http://arxiv.org/abs/2203.02155. doi:10.48550/arXiv.2203.02155, arXiv:2203.02155 [cs].

[20] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019. URL: http://arxiv.org/abs/1810.04805. doi:10.48550/arXiv.1810.04805, arXiv:1810.04805 [cs].

[21] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, W. Fedus, Emergent Abilities of Large Language Models, 2022. URL: http://arxiv.org/abs/2206.07682. doi:10.48550/arXiv.2206.07682, arXiv:2206.07682 [cs].

[22] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, Q. V. Le, Finetuned Language Models Are Zero-Shot Learners, 2022. URL: http://arxiv.org/abs/2109.01652. doi:10.48550/arXiv.2109.01652, arXiv:2109.01652 [cs].

[23] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, D. Zhou, Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, 2023. URL: http://arxiv.org/abs/2201.11903. doi:10.48550/arXiv.2201.11903, arXiv:2201.11903 [cs].

[24] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, Y. Zhuang, HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in HuggingFace, 2023. URL: http://arxiv.org/abs/2303.17580. doi:10.48550/arXiv.2303.17580, arXiv:2303.17580 [cs].

[25] V. Sanh, A. Webson, C. Raffel, S. H. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, T. L. Scao, A. Raja, M. Dey, M. S. Bari, C. Xu, U. Thakker, S. S. Sharma, E. Szczechla, T. Kim, G. Chhablani, N. Nayak, D. Datta, J. Chang, M. T.-J. Jiang, H. Wang, M. Manica, S. Shen, Z. X. Yong, H. Pandey, R. Bawden, T. Wang, T. Neeraj, J. Rozen, A. Sharma, A. Santilli, T. Fevry, J. A. Fries, R. Teehan, T. Bers, S. Biderman, L. Gao, T. Wolf, A. M.

Rush, Multitask Prompted Training Enables Zero-Shot Task Generalization, 2022. URL: http://arxiv.org/abs/2110.08207. doi:10.48550/arXiv.2110.08207, arXiv:2110.08207 [cs].

[26] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, J.-R. Wen, A Survey of Large Language Models, 2023. URL: http://arxiv.org/abs/2303.18223. doi:10.48550/arXiv.2303.18223, arXiv:2303.18223 [cs].

[27] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le, E. Chi, Least-to-Most Prompting Enables Complex Reasoning in Large Language Models, 2023. URL: http://arxiv.org/abs/2205.10625. doi:10.48550/arXiv.2205.10625, arXiv:2205.10625 [cs].

[28] C. Wang, S. X. Liu, A. H. Awadallah, Cost-Effective Hyperparameter Optimization for Large Language Model Generation Inference, 2023. URL: http://arxiv.org/abs/2303.04673. doi:10.48550/arXiv.2303.04673, arXiv:2303.04673 [cs].

[29] M. Renze, E. Guven, The Effect of Sampling Temperature on Problem Solving in Large Language Models, 2024. URL: http://arxiv.org/abs/2402.05201. doi:10.48550/arXiv.2402.05201, arXiv:2402.05201 [cs].

[30] C. community, Cosmic website, https://cosmic-sizing.org/cosmic-publications/overview/, 2024. Accessed: 2024-07-11.

[31] S. Laqrichi, Cosmic_Case_studies_measured_requirements, 2024. URL: https://github.com/slaqrichi/COSMIC-case-studies-dataset.

[32] S. Salem, H. Soubra, Functional Size Measurement Automation for IoT Edge Devices, Rome, Italy, 2023.

[33] H. Soubra, A. Abran, Functional Size Measurement for the Internet of Things (IoT): An example using COSMIC and the Arduino open-source platform, 2017. doi:10.1145/3143434.3143452.

[34] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, T. Scialom, Llama 2: Open Foundation and Fine-Tuned Chat Models, 2023. URL: http://arxiv.org/abs/2307.09288. doi:10.48550/arXiv.2307.09288, arXiv:2307.09288 [cs].

[35] W. Huang, X. Ma, H. Qin, X. Zheng, C. Lv, H. Chen, J. Luo, X. Qi, X. Liu, M. Magno, How Good Are Low-bit Quantized LLaMA3 Models? An Empirical Study, 2024. URL: http://arxiv.org/abs/2404.14047. doi:10.48550/arXiv.2404.14047, arXiv:2404.14047 [cs].