Evaluating the Role of Heuristics in LLM-Based Extraction of Entity-Relationship Models

Tatjana Stojanović
Faculty of Organisational Sciences
University of Belgrade
Belgrade, Serbia
tatjana.stojanovic@fon.bg.ac.rs
[0000-0001-7191-6444]

Kristina Jovanović
Faculty of Organisational Sciences
University of Belgrade
Belgrade, Serbia
kristina.jovanovic@fon.bg.ac.rs
[0009-0005-2443-4094]

Saša D. Lazarević
Faculty of Organisational Sciences
University of Belgrade
Belgrade, Serbia
sasa.lazarevic@fon.bg.ac.rs
[0000-0002-5588-4195]

***Abstract*— Translating natural language requirements into precise conceptual models is a challenging and very important step in software development. This paper investigates the effectiveness of combining modelling heuristics with large language models for extracting Extended Entity-Relationship (EER) models from text. Two prompting strategies were compared: zero-shot LLM prompting and heuristic-guided prompting. The evaluation, based on 14 manually annotated models, compares precision, recall, and F1-score for modeling constructs: entities, attributes, and relationships. Results suggest that integrating heuristics can improve the reliability and consistency of LLM-generated models, particularly in entity and attribute extraction. Although the heuristic-guided method requires more processing time, it shows stronger alignment with established modeling practices and offers better resilience to textual ambiguity. These findings indicate that heuristics can provide valuable structure and guidance to LLMs, and highlight the potential for further refinement to address recurring errors and improve performance across diverse domains.**

Keywords*— *extended entity-relationship model, large language models, heuristics

I. INTRODUCTION

Software requirements analysis is a critical phase in the software development lifecycle. Translating informal natural language requirements into precise conceptual models such as Extended Entity-Relationship models, UML class diagrams, and relational schemas is important for bridging the gap between stakeholders and programmers. Mistakes made during this phase can be very difficult and expensive to correct during development. This translation process is challenging due to the ambiguity and complexity of natural language. Many existing approaches rely on linguistic heuristics and NLP techniques to convert informal text into formal models, but they often face difficulties with ambiguity and domain-specific details. Meanwhile, large language models (LLMs) offer strong contextual understanding but can lack the precise constraints necessary for accurate modeling. This paper investigates how combining heuristics with LLMs can improve extraction of Extended Entity-Relationship model quality.

Extended Entity-Relationship (EER) is a high-level data model that incorporates the extensions to the original Entity-Relationship (ER) model. EER is high-level conceptual data model that sets the stage for more advanced database design and analysis [1]. In addition to ER model concepts, EER includes [2]:

- Subclasses and Super classes.
- Specialization and Generalization.
- Category or union type.
- Aggregation.

By guiding the LLM through step-by-step heuristics, the accuracy and consistency of generated models improve compared to zero-shot LLM prompting alone. Experimental results demonstrate that this combined approach better captures entities, attributes, and relationships from textual descriptions.

II. RELATED WORK

The extraction of conceptual models such as EERM, UML class diagrams and relational schemas from natural language specifications has been extensively studied, with a range of approaches leveraging linguistic heuristics and natural language processing (NLP) techniques to bridge the gap between informal text and formal models.

Large language models are a category of foundation models trained on immense amounts of data making them capable of understanding and generating natural language and other types of content to perform a wide range of tasks [3]. Generative pretrained transformers (GPTs) are a family of large language models based on a transformer deep learning architecture. Developed by OpenAI, these foundation models power ChatGPT and other generative AI applications capable of simulating human-created output [4].

Tjoa and Berger [5] introduce the Data Model Generator (DMG), a rule-based tool that maps natural language to EER concepts using defined syntactic and semantic heuristics. Their key contribution lies in categorizing heuristics into five domains: entity types, attributes, relationships, generalization hierarchies, and cardinalities. While rules denote deterministic mappings, heuristics handle ambiguity and rely on user feedback to resolve uncertainty in natural language constructs. Their empirical findings show that even complex sentence structures can be parsed effectively, although name and structural conflicts may arise, emphasizing the need for user-system interaction during model generation.

Omer and Wilson [6] extend the focus to relational database schema extraction and emphasize the importance of using sentence subjects and their environments to accurately

identify tables. They achieve up to 96% accuracy in mapping subject environments to table names, demonstrating a quantifiable improvement, over traditional methods. They also introduce heuristics based on linguistic markers like prepositions, genitives and adjectives to infer relationships and attributes, but they do not provide evaluation metrics for all aspects of their approach, which remains a noted limitation.

Btoush and Hammad [7] apply NLP pipelines like tokenization, POS tagging, chunking and parsing to identify ER elements. Their work uniquely incorporates tools such as MBSP and WordNet to semantically filter nouns and enhance attribute recognition. They emphasize syntactic heuristics e.g., using verbs to identify relationships and genitive phrases to detect attributes and highlight the challenges of linguistic ambiguity, calling for deeper semantic analysis in future work. Similarly, Herchi and Abdessalem [8] introduce "DC Builder" that automates UML class diagram generation using GATE and Chen's heuristics, filtering out domain-neutral nouns and using gerunds and possessive structures to distinguish attributes and relationships. Both studies underscore the importance of preprocessing (e.g., removing pluralization) and contextual filtering to improve accuracy.

Studies like those by Hartmann and Link [9], Javed and Lin [10], and Robeer et al. [11] introduce more advanced, flexible models by treating complex sentence structures as higher-order relationships and applying type dependency rules (via CoreNLP) to infer entities, attributes, and cardinalities. Their iterative processes and evaluation against case studies reveal the practical utility of these heuristics, showing performance metrics (80–92% accuracy in the Visual Narrator tool [11]) that validate the heuristic-based approaches.

Arulmohan et al. [12] demonstrated that using ChatGPT cannot surpass specialized tools for creating domain models. The presented tool automatically generates domain models from task lists in agile development, but it still depends on a structured user story format.

K. Chen et al. [13] attempted to develop a system that would enable complete automation of domain model creation. In their work, they analyzed different models – GPT-3.5 and GPT-4 – and various prompting strategies, such as zero-shot, n-shot, and chain-of-thought, with evaluation carried out manually. Their results showed that the solutions are reliable, but certain elements are often omitted, indicating that full automation of modeling is still not possible, with the biggest issue being the definition of relationships. GPT-4 performed better than GPT-3.5. As for prompting strategies, adding examples had a positive impact, but the chain-of-thought approach actually reduced performance.

Cámara et al. [14] demonstrated that a large language model can generate syntactically correct UML models and OCL constraints based on text expressing user intent. The models were mostly syntactically correct, but semantic correctness depended both on the information the model has about the domain (from its training data) and on the exchanged messages. They showed that smaller problems, with 10–12 classes, can be successfully modeled, but the model has issues with larger problems. Regarding concepts, it can successfully use associations, aggregations, compositions, simple inheritance, and role names.

Overall, the reviewed literature shows that heuristics, when systematically applied via NLP tools, significantly improve the automation and reliability of generating conceptual models from natural language specifications, although there is a continuing need for support for ambiguity and domain-specific lexicons. Augmenting these heuristic pipelines with large language models can further elevate performance, because LLMs provide broad semantic context and reasoning while heuristics impose domain-specific rules that keep the output precise and consistent. Combining heuristics with large language models can further elevate performance, as LLMs supply broad semantic context and reasoning, whereas heuristics impose domain-specific constraints that maintain precision and consistency.

III. Methodology

To evaluate the impact of integrating heuristics with a large language model (LLM) for the extraction of extended entity-relationship (EER) models, a series of experiments were conducted using two distinct prompting strategies on a set of textual model descriptions. It was implemented as a C# application, which automated the process of dataset preparation, LLM communication and evaluation.

In the first strategy, the LLM was prompted in a "zero-shot" fashion, providing it only with a system prompt and a natural language description of the system to be modeled.

The system prompt was:

"*You are an AI assistant designed to extract entity-relationship models from natural language descriptions. You will receive a verbal system description.*"

After generating the initial entity-relationship model, the system was further instructed to revise and improve the model by issuing the prompt:

"*Revise additionally model and return the final result.*"

The function calling feature of the OpenAI API was used, supplying the model with a predefined JSON schema that formally described the metamodel of the extended entity-relationship language (EERL). The schema included the following key constructs:

- **Entity**: Each entity is identified by a unique name and a collection of attributes.
- **Attribute**: Each attribute is characterized by its name and may be further annotated with type or identifier status.
- **Relationship**: Every relationship possesses a unique name and consists of two binary mappings, each defining the participating domain and codomain entities along with multiplicity (cardinality) bounds.
- **Generalization/Specialization**: This construct models inheritance hierarchies, specifying a super-entity and one or more sub-entities.

The second strategy extended the zero-shot approach by introducing domain-specific modelling heuristics. After receiving the initial verbal description, the LLM was used in combination with heuristics adapted from the literature. [15]. The heuristics were categorized as follows:

- Entity-related heuristics
- Attribute-related heuristics
- Relationship-related heuristics
- Cardinality-related heuristics
- Additional modelling decision heuristics.

The heuristics used are shown in Table I.

TABLE I MODELLING HEURISTICS

Entities
• Common nouns in a sentence (e.g., person, employee, airplane) are candidates for entity types in an Entity-Relationship diagram. • Proper nouns in a sentence indicate entities. • In the case of multiple consecutive nouns, pay attention to the last one. If the noun does not belong to the set S, where S = {Number, Code, Name, ID, Date, Type, Quantity, Address, Phone, FirstName, LastName}, it is likely referring to an entity type. Otherwise, the noun indicates an attribute. • Every entity type must be relevant to the problem domain being modeled by the entity-relationship diagram. • Nouns such as database, company, enterprise, bank, system, information, organization, and record are usually not suitable candidates for entity types. In general, nouns that establish the general context of the problem domain or refer to the business environment should not be considered as entity types. • If an entity type has only a single instance and there is no need to track changes to its value over time, it is typically not modeled as a separate entity type but rather as an attribute.
Attributes
• Adjectives in a sentence may indicate attributes of an entity type. • Attributes of an entity type can also be nouns mentioned in the text in connection with that entity type. Such nouns often appear alongside possessive or relational verbs such as has, possesses, contains, is stored, is recorded, is tracked, etc., suggesting that the entity type possesses or contains certain properties. • If a noun is used alongside another noun from the set S = {Number, Code, Name, ID, Date, Type, Quantity, Address, Phone, FirstName, LastName}, then both nouns are likely to indicate attributes of the entity type. Otherwise, the first-mentioned noun likely refers to an entity type. • Possessives indicate ownership and can point to nouns that represent attributes of an entity type. Possessives are most commonly expressed through possessive pronouns, possessive adjectives, or possessive forms of nouns (e.g., employee's salary). • The results of mathematical operations, mathematical objects, and numerically expressed properties may indicate attribute value types of the corresponding entity types.
Relations
• Transitive verbs are verbs that require a direct object and typically express a clear relationship between the subject (one entity) and the object (another entity). • A verbal noun in a sentence is a candidate for aggregation in an Entity-Relationship diagram (ERD), as it usually describes a process or activity that involves multiple entities and relationships. Adverbs and adverbial phrases in a sentence are candidates for relationship attributes in an Entity-Relationship diagram. • A verb followed by a preposition such as "from," "to," "in," or "on" may indicate a type of relationship. • Sentences of the form "A is/can be B and/or C and/or D…"—in which two or more nouns are connected in this way—indicate a generalization-specialization (IS-A) hierarchy among entities A, B, C, and D. • If a subset of entities within an entity set has specific attributes or participates in specific relationships with entities from other sets, this indicates a need to introduce an IS-A hierarchy. • Relationships between entities of the same set suggest the need for a recursive relationship type. The roles of the entities in the relationship must be explicitly stated.
Cardinalities
• Phrases such as "at least one," "exactly one," "at most one," "more," "can," and "must" indicate cardinalities of relationships. • Temporal or spatial precedence of entities from one set over entities from another set may suggest an existential dependency of the second set's entities on the first, which is expressed through a mandatory cardinality: min cardinality = 1. • The adjective "more" followed by the preposition "than" and a cardinal number may indicate the degree of cardinality between two entities—specifically the lower bound. • If entities from the first set cannot be uniquely identified without entities from the second set, this indicates both identification and existential dependency of the second set on the first, which is expressed by a Strong-Weak relationship
Other
• If a real-world object has, or can have, multiple distinct and meaningful properties, it should be modeled as an entity; if it is simple and lacks interesting properties, it should be modeled as an attribute. • If a real-world object has a complex structure, it can be modeled as: a single composite attribute – the simplest but least precise approach, multiple simple attributes – more precise and appropriate when there is no dependency between them and values do not repeat, a separate entity – the most precise but also the most resource-intensive; used when there is dependency among attributes or when the entity can have multiple values. • If the specific characteristics are important to the model's semantics, a generalization-specialization (IS-A) hierarchy should be used; otherwise, the concept should be modeled as an attribute. • If the semantics of a real-world object correspond to aggregation, it should be modeled as an aggregation rather than as a set of binary relationships. • If different terms or word forms appear in the text that may refer to the same concept, all such terms should be considered potential synonyms and unified into a single concept in the model (entity, attribute, or relationship). • If an entity type has only one attribute and participates in only one relationship with another entity type, it will most likely be transformed into an attribute of that other entity type. • A good modeling practice is to treat every binary relationship where both mappings have maximum cardinality = M as an aggregation • If heuristics are in conflict, domain analysis takes precedence. If a concept plays an essential role in the domain, it should be modeled as an entity, regardless of formal grammatical indications.

System prompts were designed to explicitly instruct the model to follow the heuristics step by step:

"*You are an AI assistant designed to extract entity-relationship models from natural language descriptions. You will first receive a verbal system description, followed by a series of heuristics to apply step by step. Follow each heuristic carefully to construct the model.*"

"*You will receive a description first, then heuristics one by one. Follow each heuristic step by step to construct the model.*"

During execution, the LLM was sequentially supplied with the verbal description, followed by heuristics of each group as a separate user message, ensuring the stepwise reasoning process. After all heuristics had been provided, a final revision prompt was issued to encourage the model to refine and consolidate the output. Fig. 1 illustrates the differences between two prompting strategies. The evaluation dataset comprised 14 verbal system descriptions, manually selected from a collection of 120 descriptions [16]. Each description was annotated with a reference EER model, constructed manually in accordance with the ERL schema and intended as the ground truth for evaluation. All experiments were conducted using the OpenAI "o3-mini-2025-01-31" model. The temperature parameter was set to 0 to minimize randomness and maximize response consistency. The o3 model is described as a reasoning-oriented language model with advanced capabilities in code generation, mathematical reasoning, scientific analysis, visual perception, and chain-of-thought prompting [17].

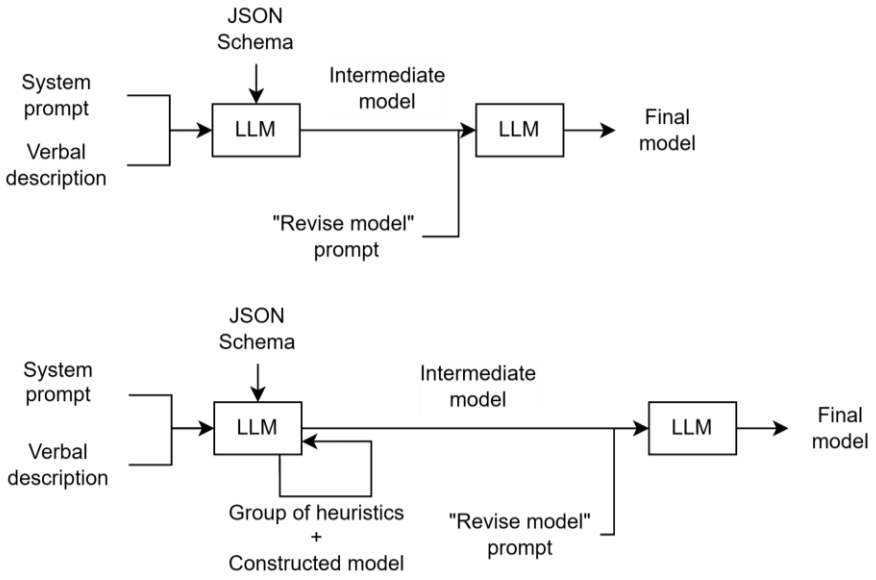

Fig. 1 Prompting strategies: zero-shot (top) vs. heuristic-guided (bottom) model extraction using LLMs.

For each system description and prompting strategy, the output model generated by the LLM was parsed and evaluated against the corresponding manually constructed reference model. The evaluation focused on the following core concepts:

- Entity detection and attribute extraction,
- Relationship identification and mapping accuracy,
- Generalization/specialization structure.

To quantify the quality of extraction, precision, recall, and F1-score were computed for each model and strategy. Each automatically generated model was compared against a manually constructed reference model, created by domain experts. These reference models serve as the gold standard for evaluating the correctness of extracted entities, attributes, relationships, and generalizations.

Precision measured the proportion of correctly predicted elements among all elements generated by the LLM. Recall measured the proportion of correctly predicted elements relative to all relevant elements in the reference model. F1-score represented the harmonic mean of precision and recall.

False positives, false negatives, and true positives were counted according to exact matches. Limitations of this approach is that the absence of a single, universally accepted correct solution means that certain modeling decisions may be unjustly penalized. Differences in naming or representation can lead to elements being incorrectly counted as mismatches. To reduce such penalties, all concept names were normalized by removing whitespace and applying uppercase formatting. For relationships, lower bounds of 0 and 1 were treated equivalently when the upper bound was M. Common attribute names, such as 'phone number' and 'email address', were standardized to 'phone' and 'email'. However, some inconsistencies in naming still remained and may have affected the final comparison.

IV. Results

The solution with heuristics involved multiple interactions with the model, leading to a notably longer construction time, as detailed in Table II and Fig. 2. On average, LLM prompting with heuristics required 2 minutes and 8 seconds, whereas the zero-shot LLM approach produced solutions in 51 seconds on average. The longest runtime for the heuristics-based method was 6 minutes and 58 seconds (R1), while the maximum duration for the zero-shot approach was 1 minute and 18 seconds (R9). Across all runs, the heuristics-based approach consistently resulted in longer response times compared to the direct zero-shot LLM prompting.

TABLE II EXECUTION TIMES PER RUN FOR LLM PROMPTING WITH AND WITHOUT HEURISTICS

Time in min	LLM + heuristics	LLM
R1	6 min 58 sec	1 min 4 sec
R2	1 min 0 sec	0 min 40 sec
R3	1 min 49 sec	0 min 49 sec
R4	1 min 48 sec	1 min 2 sec
R5	1 min 32 sec	0 min 40 sec
R6	1 min 39 sec	0 min 41 sec
R7	2 min 3 sec	0 min 20 sec
R8	1 min 39 sec	0 min 59 sec
R9	2 min 41 sec	1 min 18 sec
R10	2 min 10 sec	0 min 54 sec
R11	1 min 0 sec	0 min 25 sec
R12	2 min 1 sec	1 min 13 sec
R13	1 min 34 sec	0 min 45 sec
R14	2 min 0 sec	0 min 57 sec
Average	2 min 8 sec	0 min 51 sec

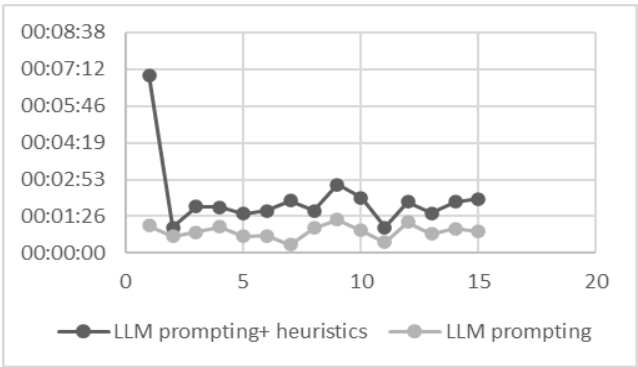

Fig. 2 Comparison of execution times for LLM prompting with and without heuristics.

In Table III and Fig. 3 average values for precision, recall and F1-score are shown. For entities, the combined approach shows consistent improvement across all metrics: precision increases from 0.88 to 0.92, recall from 0.80 to 0.87, and F1-score from 0.84 to 0.89. This indicates that incorporating heuristics helps the model detect entity concepts more reliably and completely. In the case of attributes, precision improves from 0.60 to 0.79, recall remains high (slightly decreasing from 0.88 to 0.86), and the F1-score increases from 0.71 to 0.83. This suggests that heuristics enhance attribute identification, particularly by reducing false positives. Relationships also benefit from the combined method. Precision rises from 0.64 to 0.68, recall from 0.61 to 0.63, and F1-score from 0.63 to 0.66. Although the improvement is not as strong, it still indicates a consistent gain in model performance.

When using the LLM without heuristic support, the generated models frequently include additional attributes for entities but inconsistently so. For example, some entities are assigned an Id, while others receive a name, with no clear rationale for the variation. In contrast, models generated with heuristics exhibit more consistent naming conventions for relationships. For instance, the LLM enhanced with heuristics generated semantically meaningful relationship names such as *managedBy* or *locatedIn*, instead of less precise labels like *Department_Office* or *Department_Manager*. However, in this evaluation, the naming of relationships was not assessed.

In cases where the textual description is ambiguous, it becomes difficult to assess the correctness of either model. However, the zero-shot strategy tends to omit certain concepts entirely, whereas the heuristic-augmented model generally preserves more of the relevant information. When the source text is clear and well-structured, both models perform similarly.

Differences become evident in edge cases. In one example, a feature that should have been modeled as an attribute was incorrectly treated as an operation by the LLM-only version, resulting in a poorly defined entity. Conversely, the heuristic-based model failed to extract any meaningful structure in that case. In another scenario, the LLM without heuristics completely misrepresented the domain by omitting key entities and generating recursive relationships, along with several cardinality errors. These problems were not present in the LLM with heuristics strategy.

The LLM-only model defined inheritance relationships but failed to associate the corresponding sub entities with concrete parent entities. Both models failed to capture the intended semantics of the statement “Some people are children and others are adults (it is forbidden to employ children)”, mistakenly creating a relationship Employs between Hotel and Person instead of Hotel and Adult.

In certain cases, ambiguity in the source text was substantial enough to make even human interpretation uncertain. In such instances, the LLM-only model occasionally produced more plausible structures, though it remains unclear how well these aligned with the intended meaning. In a few scenarios, the model without heuristics incorrectly treated the system modeling as a separate entity, even though such modeling was not appropriate.

The use of an expert-annotated reference model as ground truth introduces certain limitations that should be acknowledged. Although the experts were instructed to follow strictly the textual description, modeling decisions were inevitably influenced by domain familiarity and personal judgment. In well-known domains, such as library systems, the expert introduced elements like Loan entities to capture historical data, even though the requirement was not explicitly stated, based on conventional design practices.

Similarly, modelers often apply common design patterns by default, such as introducing Order and OrderItem entities to represent compositions or transactions, despite the absence of such structures in the verbal description. This reflects a general tendency in conceptual modeling to solve problems “the usual way,” using familiar solutions rather than relying solely on the text. In less familiar domains, the expert adhered more literally to the description, resulting in models that closely followed the surface structure of the input. This inconsistency mirrors the behavior of the LLM, which performs better in domains it implicitly recognizes and struggles with more obscure or underspecified contexts.

While the reference models offer a practical evaluation baseline, their subjectivity highlights the need for multiple annotators or inter-annotator agreement analysis in future work to increase objectivity and reproducibility.

Overall, combining heuristics with the LLM improves the extraction of all concept types, supporting the idea that domain-specific rules can complement the LLM’s general understanding of language. Still, the current set of heuristics might benefit from further refinement, especially to better target the kinds of errors the LLM tends to make on its own.

TABLE III COMPARISON OF PRECISION, RECALL, AND F1-SCORE FOR BOTH PROMPTING STRATEGIES ACROSS ENTITIES, ATTRIBUTES, AND RELATIONSHIPS.

	LLM			**LLM + Heuristics**		
Concepts	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Entities	0.88	0.80	0.84	0.92	0.87	0.89
Attr.	0.60	0.88	0.71	0.79	0.86	0.83
Rel.	0.64	0.61	0.63	0.68	0.63	0.66

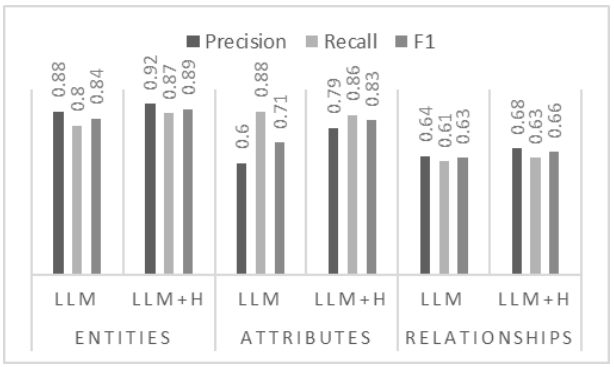

Fig. 3 Comparison of precision, recall, and F1-score for LLM prompting with and without heuristics across entities, attributes, and relationships.

V. Conclusion

This study explored the impact of integrating domain-specific heuristics with a large language model (LLM) for the extraction of extended entity-relationship models from natural language descriptions. The results show that the combined approach consistently outperforms zero-shot LLM prompting across all evaluated concept types based on precision, recall, and F1-score. In particular, the use of heuristics led to more consistent model structures, better handling of ambiguous descriptions, and improved naming coherence, especially for attributes and relationships.

Despite the increased execution time, the heuristic-supported approach proves beneficial in enhancing both accuracy and stability of the extracted models. However, the analysis also revealed that the current heuristic set may not address all failure cases, especially in more complex or ambiguous scenarios. This highlights the need for further refinement of the heuristics, potentially guided by an error analysis of LLM-only outputs. Future work should also consider dynamic heuristic selection based on text complexity. The findings support the conclusion that heuristics are a valuable complement to LLMs in domain modeling tasks, contributing not only to improved extraction quality but also to greater alignment with domain-specific modeling conventions.

References

[1] "What is an Entity Relationship Diagram? | IBM." Accessed: Jun. 19, 2025. [Online]. Available: https://www.ibm.com/think/topics/entity-relationship-diagram

[2] "Extended Entity-Relationship (EER) Model." Accessed: Jun. 19, 2025. [Online]. Available: https://www.tutorialspoint.com/Extended-Entity-Relationship-EE-R-Model

[3] "What Are Large Language Models (LLMs)? | IBM." Accessed: Jun. 19, 2025. [Online]. Available: https://www.ibm.com/think/topics/large-language-models

[4] "What is GPT (generative pre-trained transformer)? | IBM." Accessed: Jun. 19, 2025. [Online]. Available: https://www.ibm.com/think/topics/gpt

[5] A. M. Tjoa and L. Berger, "Transformation of requirement specifications expressed in natural language into an EER model," in *Entity-Relationship Approach — ER '93*, vol. 823, R. A. Elmasri, V. Kouramajian, and B. Thalheim, Eds., in Lecture Notes in Computer Science, vol. 823. , Berlin/Heidelberg: Springer-Verlag, 1994, pp. 206–217. doi: 10.1007/BFb0024368.

[6] M. A. M. Omer and D. Wilson, "New rules for deriving formal models from text," in *2016 International Conference for Students on Applied Engineering (ISCAE)*, Newcastle upon Tyne, United Kingdom: IEEE, Oct. 2016, pp. 328–333. doi: 10.1109/ICSAE.2016.7810212.

[7] E. S. Btoush and M. M. Hammad, "Generating ER Diagrams from Requirement Specifications Based On Natural Language Processing," *Int. J. Database Theory Appl.*, vol. 8, no. 2, pp. 61–70, Apr. 2015, doi: 10.14257/ijdta.2015.8.2.07.

[8] H. Herchi and W. Ben Abdessalem, "From user requirements to UML class diagram," Nov. 2012.

[9] S. Hartmann and S. Link, *English Sentence Structures and EER Modeling.*, vol. 67. 2007, p. 35.

[10] M. Javed and Y. Lin, "Iterative Process for Generating ER Diagram from Unrestricted Requirements:," in *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering*, Funchal, Madeira, Portugal: SCITEPRESS - Science and Technology Publications, 2018, pp. 192–204. doi: 10.5220/0006778701920204.

[11] M. Robeer, G. Lucassen, J. M. Van der Werf, F. Dalpiaz, and S. Brinkkemper, *Automated Extraction of Conceptual Models from User Stories via NLP*. 2016, p. 205. doi: 10.1109/RE.2016.40.

[12] S. Arulmohan, M.-J. Meurs, and S. Mosser, "Extracting Domain Models from Textual Requirements in the Era of Large Language Models," in *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, Västerås, Sweden: IEEE, Oct. 2023, pp. 580–587. doi: 10.1109/MODELS-C59198.2023.00096.

[13] K. Chen, Y. Yang, B. Chen, J. A. Hernández López, G. Mussbacher, and D. Varró, "Automated Domain Modeling with Large Language Models: A Comparative Study," in *2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, Västerås, Sweden: IEEE, Oct. 2023, pp. 162–172. doi: 10.1109/MODELS58315.2023.00037.

[14] J. Cámara, J. Troya, L. Burgueño, and A. Vallecillo, "On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML," *Softw. Syst. Model.*, vol. 22, no. 3, pp. 781–793, Jun. 2023, doi: 10.1007/s10270-023-01105-5.

[15] Lazarević Saša, Programiranje i podaci: Specifikacija softvera, III sveska, U štampi. II LSI, 2025..

[16] A. K. Khalilipour, "TextRequirements2Models." IEEE DataPort. doi: 10.21227/R9J6-ND62.

[17] "Introducing OpenAI o3 and o4-mini." Accessed: Jun. 07, 2025. [Online]. Available: https://openai.com/index/introducing-o3-and-o4-mini/