Assessing LLMs Suitability for Knowledge Graph Construction

Vasile Ionut Remus Iga [a] and Gheorghe Cosmin Silaghi [a,*]
[a] *Business Informatics Research Center, Babeş-Bolyai University, Cluj-Napoca, Romania*
E-mails: vasile.iga@ubbcluj.ro, gheorghe.silaghi@ubbcluj.ro

Abstract. Recent work has shown the capability of Large Language Models (LLMs) to solve tasks related to Knowledge Graphs including Knowledge Graph Construction, even in Zero- or Few-Shot paradigms. However, they are known to hallucinate answers or output results in a non-deterministic manner, thus leading to wrongly reasoned responses, even if they satisfy the user's demands. This hinders the inclusion of a LLM in the automatic processing pipeline of software products based on natural language processing, like chatbots or Task-Oriented Dialogue systems. To highlight opportunities and challenges in knowledge graphs-related tasks, we experiment with three distinguished LLMs, namely Mixtral-8x7b-Instruct-v0.1, GPT-3.5-Turbo-0125 and GPT-4o, on Knowledge Graph Construction for static knowledge graphs, using prompts constructed following the TELeR taxonomy in Zero- and One-Shot contexts, on a Task-Oriented Dialogue system usecase. We introduce a flexible measurement paradigm for the computation of the performance evaluation metrics in order to better assess all usable pieces of information produced by the LLM. When evaluated using both strict and flexible metrics measurement manners, our results show that LLMs could be fit for such a task if prompts encapsulate sufficient information and relevant examples.

Keywords: Large language models, Knowledge graph, Knowledge graph construction, Prompt engineering, Task-oriented dialogue system

1. Introduction

Knowledge Graphs (KGs) are defined as graphs of data intended to accumulate and convey knowledge of the real world [7]. Their nodes represent entities of interest and edges represent potentially different relations between these entities. KGs are integrated into various systems to enhance their abilities of storing and processing information.

Task-Oriented Dialogue (TOD) systems, alongside chatbots, are conversational agents possessing capabilities of engaging in natural language dialogues with human users. Different from chatbots, TOD systems aim to solve the user's specific tasks within certain domains [2]. In our previous work [9] we focused on developing an ontology-enhanced TOD system equipped with a static KG capable of mapping the context of the discussion and storing relevant information. Numerous benefits stem from adopting this approach, including enabling concurrent threads of conversation within a single discourse and utilizing the KG to validate data as a proxy. Additionally, the system gains the capability to execute Create-Retrieve-Update-Delete (CRUD) operations on domain-specific KGs. The acronym CRUD refers to the four basic operations that can be executed against persistent storages, such as relational or object databases, or other types of knowledge bases like KGs, to create, maintain or update them. As TOD systems aim to solve a variety of specific tasks, we decided that enabling such basic, but important operations is a suitable use-case to start with. Specifically, in our TOD system we employed the Knowledge Graph Construction (KGC) task to

*Corresponding author. E-mail: gheorghe.silaghi@ubbcluj.ro.

create two KGs and the Knowledge Graph Reasoning (KGR) task to handle CRUD operations. KGC's objective is to extract structured information from natural text and further use the extracted information to create or extend KGs.

Nonetheless, our TOD system [9] relies on input text template-matching rules, constraining the authenticity of dialogues and hindering adaptability to novel concepts beyond the predefined ontology. Hence, in a subsequent study [8], we experimented with training neural networks - specifically fine-tuning BERT, a pre-trained model, to discern user intent and pertinent associated entities from input text. While embedding deep learning models into the TOD system architecture demonstrated encouraging outcomes, we still failed to completely address the previously mentioned limitations.

Therefore, in our current work, we study the use of LLMs to solve the KGC task, in the context of a TOD system. Literature [6, 15, 16, 23, 25] identified a potential for synergy between KGs and LLMs, as KGs can enrich LLMs by supplying external knowledge for inference and explainability, while LLMs, in turn, can address KG-related tasks through natural language prompts. The aim is that, with the help of a LLM, to extract facts from the natural text and furthermore, to be able to automatically use the extracted facts in the processing pipeline of the TOD system.

Our experiments explore LLMs for static KGs contexts. Three well-established LLMs are used: Mixtral-8x7b-instruct-v0.1[1] [12], GPT-3.5-Turbo-0125[2], alongside the most advanced GPT model, the GPT-4o[3] version, each possessing different properties. Communicating with such models involves the use of prompts, which are natural language instructions formatted in such way that the model understands the user's intent. We test their capabilities of solving the aforementioned KGC task using multiple prompting styles, including human-created and model-specific rephrased ones. Each prompt belongs to a level defined according to the TELeR taxonomy [18], that includes techniques such as Direct Prompting (DP), In-Context Learning (ICL), or Chain of Thought (COT), under Zero- and One-Shot contexts. To illustrate an appropriate application scenario for LLMs and KG tasks, we extract sample phrases from the training phase of our TOD system. Two datasets are obtained, including one with an increased difficulty, with test cases requiring reasoning steps that are not explicitly mentioned in the prompts. This approach allows us to not only evaluate the capability of LLMs in addressing the KG-specific tasks, but also to investigate their synergy with TOD systems. Finally, we report the recall and triple F1 scores [4, 6] of each LLM on both dataset, under two measurement paradigms: strict and flexible.

Our research makes the following contributions. (i) We assess the performance of three prominent LLMs: one open-source and the other two proprietary, for the KGC task. This evaluation involves employing various prompts, either defined by humans or rephrased by the LLMs themselves, across different levels of complexity. We utilize three distinct prompting techniques (DP, ICL, COT) within two data contexts (Zero-Shot and One-Shot), yielding valuable insights into the capabilities of a robust LLM in performing such task. Performance is measured within two paradigms: strict and flexible, shedding light on the challenges encountered during post-processing. (ii) We construct and propose a novel flexible metric, aiming to positively evaluate every piece of information produced by the LLM that could be automatically incorporated in the TOD system pipeline with some additional post-processing computation steps. (iii) We introduce two personalized datasets tailored to gauge the performance of LLMs in the KGC task, featuring varying levels of difficulty. (iv) We investigate the feasibility of integrating such models into a domain-specific ontology-enhanced TOD system, by extracting and using test phrases specific to its context and by assessing its performance under the flexible metric measurement paradigm.

This paper extends our previous work [10] by better defining the preliminary conditions, by revising the related work section with the inclusion of the latest relevant research, by formally introducing the flexible measurement paradigm and by presenting more detailed results that allow extracting further conclusions, including some related with the capability of the tested LLMs to solve tasks of various levels of difficulty.

The paper evolves as follows. Section 2 describes the related work about solving the KGC task with LLMs. Section 3 presents our methodology, describing the ingredients of our experiments. Section 4 presents and discusses the results, while section 5 wraps up the paper with concluding remarks.

[1] https://huggingface.co/mistralai/Mixtral-8x7b-Instruct-v0.1
[2] https://platform.openai.com/docs/models/gpt-3-5-turbo
[3] https://platform.openai.com/docs/models/gpt-4o

2. Related Work

Knowledge Graph Construction (KGC) aims to build a structured representation of knowledge within a defined domain from a free text by identifying entities and their corresponding relationships. The process generally involves several stages in a standard pipeline approach: 1) entity discovery, 2) coreference resolution, and 3) relation extraction. Recent methods also include 4) end-to-end KGC, which constructs a complete knowledge graph in a single step, and 5) extracting knowledge graphs directly from from text with the help of LLMs [16].

Many non-LLM techniques address the task by solving the first three stages rather separately. However, they could also be combined into a single process, known as Knowledge Graph Completion. This task aims to deduce absent information within a specified KG [16], drawing from either input text or preexisting knowledge. Ji et al. [11] present multiple solutions for the KG Completion utilizing embedding-based models like TransE [1], relation path reasoning exemplified by the Path-Ranking Algorithm [14], reinforcement-learning path finding [21], rule-based reasoning such as KALE [5], and meta relational learning [22] utilizing R-GCN or LSTM. Similar insights are shared by Zhang et al. [23], categorizing them into neural, symbolic, and neural-symbolic approaches.

The aforementioned studies emphasize the usage of neural networks, logic networks, logic rules, or mathematical operations to address KGC. Interestingly, none of these endeavors particularly delve into the utilization of LLMs. Wei et al. [20] advocate for a multi-stage dialogue with ChatGPT to extract pertinent information from input texts, based on a predefined schema. They do solve the KGC task by dividing it into Named Entity Recognition, Relation Extraction and Event Extraction. Zhu et al. [25] experiment with ChatGPT and GPT4 for KGC in the pipeline manner, determining that while they lag behind state-of-the-art fine-tuned Pre-Trained Language models (PLMs) in a zero/one shot paradigm for construction, their reasoning capabilities often match or surpass those of SOTA models. Nevertheless, the comparative efficiency of an LLM versus a specialized PLM remains ambiguous. They also tackle the end-to-end KGC task, by designing an interface where an AI assistant and AI user collaborate in a multi-party setting to complete the specified task. Their findings show that LLMs can solve the KGC task on their own, when a multi-turn interaction takes place. Maintaining the end-to-end KGC paradigm, Han et al. [6] introduce PiVE, a prompting technique where a ChatGPT-based LLM extracts facts from input texts, while a smaller fine-tuned PLM iteratively verifies and supplements its responses. They demonstrate that the verifier module is key to preserve the correctness of LLMs. Khorashadizadeh et al. [13] explore the capabilities of foundation models such as ChatGPT to generate KGs from the knowledge it captured during pre-training as well as the new text provided to it in the prompt, grounded by several research questions. Their results show promising use cases for such models. Trajanoska et al. [19] experiment with a specialized pre-trained model (REBEL) and ChatGPT for automating the extraction of KGs from news articles. They conclude that ChatGPT, when prompted adequately using enough information and guidelines, can solve the task with promising results. Ghanem et al. [4] evaluate various LLMs using direct prompting techniques like Zero- and Few-Shot, or precedes them by model fine-tuning. They report metrics including TF1, GF1, and Graph Edit Distance (GED) introduced in [6], while also defining new metrics for hallucination and information omission.

As opposed to the above mentioned literature, we emphasize the use of a well-defined ontology to guide the extraction of facts and, subsequently, construction of the KG. This approach stands in contrast to methods that either lack background information or rely solely on small, predefined lists of specific types and relationships. Moreover, we increase the number of textual inputs, expanding the generality of our conclusions. We share similarities with [15, 17]. Mihindukulasooriya et al. [15] distill two datasets specifically for KGC from other well-established sources and create additional metrics to test two LLMs, Vicuna-13B and Alpaca-LoRA-13 on the aforementioned task, resulting in a benchmark for KGC. However, unlike their approach, our datasets are manually curated, and we utilize an in-house designed flexible paradigm to evaluate LLM performance from a different perspective, while testing models of various types and sizes. Polat et al. [17] experiment with different prompting techniques and paradigms, from Zero to Few-Shot and DP to COT for the extraction of KGs from free input text. Different from us, prompts are enhanced with extra information obtained via various RAG approaches, while the evaluation of the output is done using SPARQL queries to Wikidata. Similar to us, they test also with Mixtral. However, in our paper, we intend to assess the performance of the LLMs on the KGC task solely based on the user free text input, without helping the LLM with additional contextual information, like the one that could be supplied with a RAG.

Consequently, we test the capacity of a proprietary LLM – namely GPT, with two versions: GPT-3.5-Turbo-0125 and GPT-4o on the KGC task. Additionally, we include an open-source LLM - Mixtral-8x7b-Instruct-v0.1 [12], to facilitate research on open-source models, given their greater adaptability and cost-effectiveness compared to proprietary alternatives. Another difference from them is that our prompts are more diverse and easier to track, as they are leveled according to the TELeR taxonomy [18]. We introduce flexible metrics for gauging additional post-processing efforts. Finally, we also test the possibility of integrating an LLM with an ontology-enhanced TOD system, to sharpen its natural language processing and KG-related capabilities, by utilizing sample phrases from its training routine, resulting in two datasets, differentiated by their level of difficulty.

3. Methodology

This section introduces our methodology used thorough this paper. We describe preliminary definitions of key concepts, the ontology used to anchor the LLMs knowledge, the datasets format and distribution, the prompt engineering steps, and the metrics measurement paradigms.

3.1. Preliminaries

Definition 1. A Knowledge Graph typically represents information as triples (or facts). Let KG denote the graph, where each triple (h, r, t) consists of head (h) and tail (t) entities, and a relationship (r) between them. The set of all entities is denoted by E, and the set of all relationships as R. The definition of a KG can be formalized as:

$$KG = \{(h, r, t)|h, t \in E, r \in R\} \tag{1}$$

An example of a fact can be $(Bill\_Clinton, presidentOf, USA)$, where *Bill_Clinton* is the head entity, $presidentOf$ is the relationship, and USA is the tail.

Definition 2. A Large Language Model (LLM) is a neural network with billions of parameters, trained on vast datasets to understand the semantics of words in texts, making it highly effective for Natural Language Processing (NLP) tasks. LLMs are classified into three types: 1) encoder-only, 2) encoder-decoder, and 3) decoder-only [16]. The decoder-only models, like those in the GPT series, are the most widely used. These models predict the next word in a sequence solely based on the preceding words. In essence, a decoder-only LLM can be described as:

$$LLM(w_0, w_1, ..., w_n) = p(w_{n+1}|w_0, w_1, ..., w_n) \tag{2}$$

where the sequence $(w_0, w_1, ..., w_n)$ contains the words in the input text, n is its length, while w_{n+1} is the next predicted word in the sentence. Hence, based on a given input text, a decoder-only LLM generates a probabilistic distribution p over all the possible words in the vocabulary.

Definition 3. The Knowledge Graph Construction task, as outlined in section 2, involves extracting entities and relationships as triples needed to build or update a KG. Typically, a dedicated system or model performs this task. In our case, an LLM serves as the extractor of the target triples - deemed as golden labels, based on a predefined ontology of entity types and relationships. The model's input is a prompt containing the task description (TD), ontology (O), and input text (IT), with an optional set of examples ($[EX]$) illustrating the process on different text inputs. Hence, KGC is formulated as follows:

$$LLM(Prompt(TD, [EX], O, IT)) = [(h, r, t)_0, (h, r, t)_1, ..., (h, r, t)_i] \tag{3}$$

where $(h, r, t)_i$ is an extractable triple from the input text, while i is the total number of predicted triples.

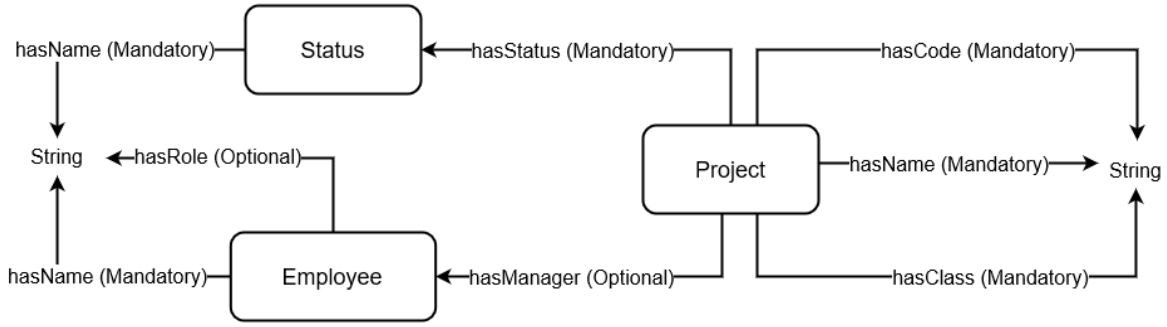

Fig. 1. The ontology used throughout the experiments with three classes and six relationships

Table 1
Input phrase examples and their types

Input phrase example	Input phrase type
I want you to insert a project instance with code being A9I, its class is C, named BestApp and put Robert as the manager.	explicit information
Please put a project called UBBDemo identified by ZK5 managed by someone with something Mara and put it with other Python projects.	implicit information
Add a porject with code DS2, nme as Taskmate, class is Python and someone with role assistant as maager.	misleading information (MS1)
I want you to insert an program instance with code being something like 0-Q7 its class is BASIC named UBBDemo and put Oscar as the manager.	misleading information (MS2)

3.2. Datasets format and distribution

Fig. 1 depicts the ontology introduced in our prior research [9] and used here. It comprises three classes: *Project*, *Employee*, and *Status*, along with six relationships connecting them - such as *hasManager* and *hasStatus* or associating classes with literal values - like *hasName*, *hasRole*, *hasClass*, and *hasCode*. The ontology is described in RDF, using the Turtle syntax.

Input phrases are sourced from the training schedule of the TOD system developed in [9], aimed to solve business operations around the concepts described in the above-presented ontology. Each phrase corresponds to the *Create* (*Insert*) intent within CRUD operations, being focused either to one of the three available classes in the ontology or to other out-of-distribution (OOD) classes. Only few phrases resemble basic tasks without the *Insert* intent (labeled as *w/o Insert*). These, along with the OOD phrases, do not contain extractable triples and are labeled as having *None* class type. The texts convey a various range of information, including both (i) explicit information where intent, class type, associated relationships, and values are clearly articulated, and/or (ii) implicit information where additional reasoning steps are required to identify the necessary details. For instance, texts may already provide an ID for the related instance, or the value might imply a name, role, or unspecified property. We also constructed phrases with misleading alternatives, where the first category contains grammatical errors, while the second one focuses on the *None* class type and ad-hoc values for some relationships of the target instance. For example, as the relationship *hasClass* requires a programming language, we include words that do not resemble existing ones, such as Dandy, Erlang etc. (labeled as WV, which stands for Wrong Values).

Table 1 presents examples from each category. Details about the explicit types are available in the project's repository[4]. Each text is accompanied by its related relationships, values, intent, and text type. Using regex templates, the information is converted into a dictionary. At the end, we obtained two datasets: *Templates Easy (TE)* and *Templates Hard (TH)*. The first dataset includes easier explicit and misleading text types with a lower number of implicit ones, while the second one benefits of an increased overall difficulty and additional implicit-type texts. Table 2 presents the distribution of texts per class type, on each dataset.

[4]https://github.com/IonutIga/LLMs-for-KGC

Table 2
Datasets distribution of texts per class type - number of phrases

Datasets	Class type				Total
	Project	Employee	Status	None	
Templates Easy (TE)	58	4	3	7	72
Templates Hard (TH)	56	4	3	15	78


```
{ 'text': 'i want you to insert a project instance with code being A9I its class
          is C named BestApp and put Robert as the manager.',
'golden_labels': "[{'subject': 'Project1', 'relationship': 'rdf:type', 'object': 'Project'},
                   {'subject': 'Project1', 'relationship': 'hasManager', 'object': 'Employee1'},
                   {'subject': 'Employee1', 'relationship': 'rdf:type', 'object': 'Employee'},
                   {'subject': 'Employee1', 'relationship': 'hasName', 'object': 'Robert'},
                   {'subject': 'Project1', 'relationship': 'hasCode', 'object': 'A9I'},
                   {'subject': 'Project1', 'relationship': 'hasClass', 'object': 'C'},
                   {'subject': 'Project1', 'relationship': 'hasName', 'object': 'BestApp'}]",
'alternative_labels': "[{'subject': 'Project1', 'relationship': 'hasManager', 'object': 'Robert'}]",
'fp_ok_labels': "[{'subject': 'Employee1', 'relationship': 'hasRole', 'object': 'Manager'},
                   {'subject': 'Employee1', 'relationship': 'hasRole', 'object': 'manager'}]",
'class_type': 'Project',
'text_type': 'MPS'}
```


Fig. 2. Example of a dictionary object with its text-related details.

We associate each text with a set of golden labels, which are the target triples that can be extracted from the input text. Additionally, we consider that under the flexible metrics measurements paradigm introduced in subsection 3.4, we can accept some triples as alternatives for the golden ones, i.e. some facts reported as false positives could be accepted if no other background information is available. It is widely acknowledged that extracting triples from text could yield a variety of results depending on the expertise of the annotator. Therefore, we should allow room for LLMs to exhibit such variability.

To better illustrate the above description of the dataset, Fig. 2 presents an example of the obtained dictionary. The proposed alternative triple substitutes for triples two, three, and four. The accepted false positive triple refers to the user's role, that can be inferred from the *hasManager* relationship.

3.3. Prompt engineering

To better encapsulate information and facilitate the replication of experiments, we utilize regex templates to convert each dictionary object - representing a text from a dataset, into a Prompt object. Such objects hold many details, such as the system prompt, its version and level, the input text, its type and mentioned class type, the golden labels and the ones that could be accepted alternatively, the system message order, and metadata information about each model's prediction of the input prompt. For a more comprehensive analysis, we enable the adjustment of the system's message position within the final prompt. Practice suggests that positioning the system's prompt after the user's message could potentially enhance the performance of LLMs by mitigating long-context memory limitations. Lastly, we provide the option to flatten each Prompt object into dictionaries that are placed in text files for future reuse.

Three important paradigms are usually employed when designing prompts [24]: Zero-, One- and Few-Shot. The former's prompt includes only the objective's description and the input data that should be processed. One-Shot includes exactly one example of how the task should be solved against some different input data. Intuitively, Few-Shot refers to multiple, relevant examples added to the prompt. After selecting the paradigm, one should decide about the prompting technique [24]. The current work employes three main approaches, as follows: Direct Prompting which refers to a prompt that only comprises the task's description and the input to work on; In-Context Learning (ICL) which adds a relevant example of a solution to the given task on a different input data, and Chain of Thought (COT) which expands the prompt with the exemplified solution's intermediary reasoning steps.

To test the model's capacity to solve a task, we follow the guidelines of [18] by assigning a level to each version of a system prompt. Specifically, we utilize levels 1 through 4 as outlined in [18]. Within the fourth level, we further divide it into 4.1 and 4.2 to accommodate both ICL and COT variations of the prompt.

'You are a Knowledge Graph Expert. A domain ontology is provided to you, delimited by double quotes. The syntax used to describe the ontology is Turtle. Your input is a natural language text. The input text may or may not contain references to instances of classes provided in the ontology, together with specific relationships. Given the provided ontology, your task is to extract triples about the mentioned instances from the input text. Each instance should be identified by an ID, using the format "Class" + "1", where "Class" is the name of the detected class and + is concatenation. Put each triple in a JSON object, as follows: {{"subject" : ID, "relationship" : value, "object" : value}}. If any triple refers to another instance, add all triples you assumed of that instance too. Respond only with the JSON object(s) in a list. If no triple is detected, output "None". \n Provided ontology: {ontology} \n'

Fig. 3. The level 1 system prompt.

For better understanding, we exemplify the first level in Fig. 3. It sets the model's role as a KG expert, followed by instructions regarding the provided ontology. Subsequently, we outline the task at hand along with formatting guidelines for each instance's ID and triple. Finally, the required output pattern is presented, attaching the target ontology. Level 2 adds a directive about the addition of the *rdf:type* relationship. It then evolves into level 3, where we transform the text into a detailed bullet list of sub-tasks to be performed. All these levels adhere to the Zero-Shot paradigm, while levels 4.1 and 4.2 emulate ICL and COT, respectively, in a One-Shot manner. Depending on the golden labels of the target input text, we include either an example with no output triples or one with existing golden labels to better suit the specific scenario. Moreover, as suggested in [16], we ask each model to rephrase the existing system prompts to better suit their needs. Therefore, we end up with two types of prompts: hand-written, and model-rephrased.

3.4. Metrics

To gain a deeper understanding of the models' performance, we measure their precision, recall, and F1 score as TF1 (Triple-matching F1) [4, 6] on both datasets. We employ two paradigms, namely strict and flexible metrics measurements. In what follows, we describe the metrics used in our paper.

For each text, the LLM produces a list of m triples. Let's denote it with $PT = \{s_i, i = \overline{1,m}\}$. For that text, the set of golden labels is $GL = \{t_j, j = \overline{1,n}\}$ with n triples. Worth-to-consider triples are those $s_i \in PT$ such that $\exists t_j \in GL | s_i \equiv t_j$.

Under the strict criterion, metrics will be calculated in a standard text extraction manner, specifically counting how many predicted triples from PT are among the golden ones GL, adhering to identical formatting. This approach enables the assessment of a model's ability to exactly follow the given prompt and process the input text, such that its results can be directly used in subsequent pipelines.

With the flexible metric measurement paradigm proposed here we allow the LLM to produce formatting mistakes that can be corrected in post-processing steps, or triples that are partially true to be counted as being accurate. This flexible measurement allows one to positively evaluate models that might not be such precise, but require fewer resources than more elaborate ones and parts of their output could be still used in the other subsequent processing steps.

To accommodate this, for each triple $s_i \in PT$ we compute a *hit score* $0 \leqslant h_i \leqslant 1$ with the GL, where $h_i = 1$ if the triple is found as-it-is in the GL (as under the strict measurement paradigm) and $h_i = 0$ is the triple is not found in GL or is totally unusable in further post-processing steps.

Under the flexible measurement paradigm we include certain penalties in the hit score of a triple that could be considered valid, even if it does not exactly follow the given prompt instructions. Therefore, $h_i = 1 - \sum_k p_k$ and h_i will still remain non-negative but less than 1.

To summarize, we have:

$$h_i = \begin{cases} 0, & if\ s_i \notin GL \\ 1, & if\ \exists\, t_j \in GL \mid s_i \equiv t_j \\ 1 - \sum_k p_k, & if\ \exists\, t_j \in GL \mid s_i \approx t_j,\ under\ flexible\ measurement\ paradigm \end{cases} \tag{4}$$

where p_k are the penalties considered in the design phase of the experiment.

Below, we list the penalties we considered for each sort of LLM output errors.

Format Penalties at the whole output level. The prompt demands a reply comprising solely a list of triples adhering to the specified template. Therefore, we consider a penalty of 2.5% for outputs with multiple lists and a penalty of 7.5% for the LLM producing additional text. We apply these penalties for all triples that result after eliminating the global formatting errors.

Format Penalties at the triple level. If the prompt asks not to include the full IRI of an entity i.e. without the namespace, we penalize each addition with 1%. Finally, if a triple is output but does not contain exactly the three necessary keys, a penalty of 10% is applied.

Content Penalties refer to penalties related to the information content of a triple. For example, let's ask a model to construct a simple ID for each given instance of a class - specifically, the capitalized name of the class concatenated with "1". We have noticed that some models tend to replace the number "1" with another single digit. Thus, if altering the final digit of a predicted identifier to "1" signifies correctness for a triple, the model is subjected to a penalty of 33%. This percentage value adheres to the three-component structure of a fact, such that, if one part is wrong, while the other two are correct, the model should still gain benefit of its prediction. This method of evaluation only applies to the validity of a constructed ID(s). Any other type of mistakes are not allowed, since they would alters the factuality of the implied information.

As previously noted, we permit certain alternative triples to the designated correct ones to be regarded as valid. Specifically, in Fig. 2, concerning the relationship labeled as *hasManager* between a *Project* instance and an *Employee* instance, if a model predicts the value of the object to directly be the employee's name, instead of creating an *Employee* instance and assign its type and name, the substitution will be counted as being correct. Nevertheless, the flexible metrics will attribute only one-third of the replacement as being accurate, implicitly penalizing the model for deviating from the prescribed ontology and guidelines. Additionally, some false positive triples may be deemed true in the absence of background knowledge (*FP_okay* triples), such as inferring the role of an employee as being a manager from the *hasManager* relationship, thus not counting them as being wrong during the calculation of the model's precision. However, we treat them as any other triple under the flexible paradigm, thus penalizing them for content mistakes, if any.

The penalty values described above and considered in our paper fit our specific task. We emphasize that these values are not fixed, and someone who wants to adapt the flexible measurement paradigm for another task is free to change them, according with her specific needs.

Next, the evaluation metrics are calculated according to the following widely-known formulas:

$$Precision_{text} = \frac{\sum_{i=1}^{m} h_i}{m}\ ,\ Recall_{text} = \frac{\sum_{i=1}^{m} h_i}{n} \tag{5}$$

$$F1_{text} = \frac{2 * Precision_{text} * Recall_{text}}{Precision_{text} + Recall_{text}} \tag{6}$$

$$TF1 = \frac{\sum_{text} F1_{text}}{|text|}\ ,\ Recall = \frac{\sum_{text} Recall_{text}}{|text|} \tag{7}$$

If the strict metric measurement paradigm is considered, the hit scores h_i could be only 0 or 1 and the metrics computed according with eq. 5-7 are the standard ones used in the literature (precision, recall and triple F1). If the flexible metric measurement paradigm is considered, the metrics computed according with the above-presented equations are more optimistic, allowing one better assess usefulness of the LLM output for subsequent processing steps.

4. Results and discussion

This section presents the obtained results and discuss them in order to conclude about the paper's research questions.

Experiments were conducted on Google Colab, utilizing a virtual machine equipped with two Intel Xeon CPU 2.20GHz processors. We experimented with Mixtral-8x7b-Instruct-v0.1, GPT-3.5-Turbo-0125 and GPT-4o. Mixtral is open-source, leveraging the Mixture of Experts[12] architecture, consisting of eight sub-networks, each of 7B parameters, accounting for a total of 56B parameters. GPT-3.5-Turbo-0125 is a well-known proprietary model that represents a fine-tuned version of GPT 3, consisting of 175B parameters. GPT-4o boasts over 200B parameters, being the latest OpenAI model, advertised as their best performer. For Mixtral-8x7b-Instruct-v0.1, we used the HuggingFace Serverless API endpoint, whereas for GPT-3.5-Turbo-0125 and GPT-4o queries were directed to OpenAI's official API.

Each experiment was iterated three times, involving aprox. 6750 prompts in total, with each run lasting approximately 120 minutes. Interaction with Mixtral consumed about 50% of the experimentation time. GPT-4o generated a cost around 40USD, while the GPT-3.5-Turbo-0125 only about 5USD. For Mixtral, the HuggingFace endpoint generated no cost. Each set of predictions could be loaded, tested and visualized from the paper's repository, available at https://github.com/IonutIga/LLMs-for-KGC.

We notice that for the GPT models, an extra post-processing step is required, after receiving the produced output. Due to their ability to generate JSON formatted output, it surrounds its response with a specific tag (i.e. "“‘json...“‘"). One solution is to include a guideline in the prompt to avoid this behavior, but very rarely, around 0.5% of times, it still adds it. Thus, to ensure that prompts are identical for all models, and be sure that the tag is not present in the output, we post-process the GPT output in our code. We do this to enable a fair analysis solely of the output text.

Tables 3 to 6 display the results per model and prompt level, considering both strict and flexible metrics measurement paradigms. The first two tables focus on the *Templates Easy (TE)* dataset, while the latter ones on *Templates Hard (TH)* dataset. Tables 3 and 5 display the results for the hand-written system prompts, while in tables 4 and 6, each model had to rephrase the prompts beforehand.

Tables 7 and 8 present class-wise model performance across both datasets, based solely on metrics from hand-written prompts, which outperformed model-rephrased alternatives, as shown in former tables.

Fig. 4 displays the recall and TF1, under the flexible paradigm for each model per phrase type from Table 1, on both datasets, using hand-written prompts.

Table 9 highlights an in-depth analysis of the "*MS 2*" category from Fig. 4, given three phrase sub-types. Table 10 outlines the results of each model when the link between a *Project* and an *Employee* instance is referenced through an ID or role, compared with standard human names.

We highlight the most effective prompt types per model and level in bold. The overall best prompts are underscored, while the overall best prompts per level are printed in italics.

Several interesting conclusions are discusses below.

Elaborate Instructions Without Examples Do Not Necessarily Yield Better Results. Upon analyzing both types of prompts across all levels, it appears that augmenting the prompt with more information without examples does not consistently enhance performance. Level 3 prompts, when evaluated rigorously, exhibit an average decline of around 7% in recall and TF1 scores compared to levels 1 and 2. When evaluated using more flexible metrics, the discrepancy diminishes to almost zero. GPT-4o tends to increase its performance with each level when using hand-written prompts, while it dramatically decreases it for the model-rephrased ones. The other two models consistently lower their metrics at the third level, especially Mixtral-8x7b, which can be attributed to the inclusion of explanatory text, as it strives to replicate the input text.

ICL and COT Prompting Techniques Lead to Best Results. Most of each models best results happened when prompted at levels 4.1 and 4.2, no matter the dataset or prompting template. Only GPT-4o had its best results for strict metrics at the first level when prompts where model-rephrased, which could be attributed to poor paraphrasing for the latter levels. It is no surprise that such models work best when an adequate output example is given, as literature [18] suggests. However, as Mixtral-8x7b sometimes provides explanations for its output, erroneous reasoning steps are noticeable, especially in cases where the input text contains a class type that is not present in the ontology. Thus, despite the GPT models exhibiting this behavior less frequently, LLMs still have significant room for improvement in terms of reasoning capabilities.

Table 3
Results on Template Easy (TE) dataset, using hand-written system prompts

Model		Mixtral	GPT-3.5-Turbo	GPT-4o	Total
Level	Metric	strict | flexible	strict | flexible	strict | flexible	strict | flexible
1	Recall	0.23 | 0.47	0.38 | 0.45	0.61 | 0.63	0.41 | 0.52
	TF1	0.25 | 0.58	0.47 | 0.58	0.73 | 0.75	0.48 | 0.64
2	Recall	0.19 | 0.49	0.45 | 0.51	0.63 | 0.85	0.42 | 0.62
	TF1	0.18 | 0.53	0.52 | 0.61	0.64 | 0.89	0.45 | 0.68
3	Recall	0.19 | 0.44	0.37 | 0.44	0.71 | 0.86	0.42 | 0.58
	TF1	0.20 | 0.55	0.48 | 0.58	0.72 | 0.89	0.47 | 0.67
4.1	Recall	**0.25** | 0.63	**0.88** | **0.88**	**0.89** | **0.91**	***0.67*** | 0.81
	TF1	**0.25** | 0.63	**0.88** | **0.88**	**0.89** | **0.91**	***0.67*** | 0.81
4.2	Recall	0.19 | **0.69**	0.85 | 0.87	**0.89** | **0.91**	0.64 | ***0.82***
	TF1	0.19 | **0.75**	0.85 | 0.87	**0.89** | **0.91**	0.64 | ***0.84***
Total Recall		0.21 | 0.54	0.59 | 0.63	0.75 | 0.83	0.52 | 0.68
Total TF1		0.22 | 0.61	0.65 | 0.72	0.78 | 0.87	0.55 | 0.73

Table 4
Results on Template Easy (TE) dataset, using model rephrased prompts

Model		Mixtral	GPT-3.5-Turbo	GPT-4o	Total
Level	Metric	strict | flexible	strict | flexible	strict | flexible	strict | flexible
1	Recall	0.38 | 0.50	0.40 | 0.46	**0.62** | 0.64	0.47 | 0.53
	TF1	0.42 | 0.59	0.48 | 0.57	**0.73** | 0.75	0.54 | 0.64
2	Recall	0.15 | 0.37	0.45 | 0.47	0.36 | 0.85	0.32 | 0.56
	TF1	0.17 | 0.49	0.54 | 0.58	0.36 | 0.90	0.36 | 0.66
3	Recall	0.20 | 0.42	0.40 | 0.46	0.07 | 0.77	0.22 | 0.55
	TF1	0.22 | 0.51	0.50 | 0.59	0.07 | 0.73	0.26 | 0.61
4.1	Recall	0.19 | 0.58	**0.85** | **0.89**	0.66 | **0.90**	***0.57*** | 0.79
	TF1	0.19 | 0.59	**0.85** | **0.89**	0.66 | **0.88**	***0.59*** | 0.79
4.2	Recall	**0.42** | **0.74**	0.84 | 0.89	0.31 | 0.87	0.52 | ***0.83***
	TF1	**0.42** | **0.78**	0.84 | 0.89	0.31 | 0.87	0.52 | ***0.85***
Total Recall		0.27 | 0.52	0.59 | 0.63	0.40 | 0.81	0.42 | 0.65
Total TF1		0.28 | 0.59	0.65 | 0.71	0.43 | 0.83	0.45 | 0.71

Mixtral-8x7b Rarely Follows the Required Output Format. The two metric measurement paradigms offer valuable insights into the models capacity to follow the given prompts. While GPT 3.5-turbo and GPT-4o exhibit minimal disparity between the two perspectives, Mixtral 8x7b rarely produces texts that align with the specified template. Common errors include the addition of explanatory text, as evidenced by the 0 scores at the 4.2 level in table 5, or the full IRI of an entity. When strictly evaluated, the open-source model only tops 42% recall on the Template Easy (TE), while on flexible paradigm it reaches 74% recall on the same dataset. GPT-3.5 outputs 88% recall under both metrics measurements, while the GPT-4o variant yields 89%.

Asking Models to Rephrase the System Prompt Might Generally Be a Good Idea for Mixtral-8x7b. Some experiments in the literature [16] ask the LLMs to formulate prompts for a given task. Inspired by it, we ask the LLMs to rephrase our manually written prompts to better align with their capabilities. As a comparison, Mixtral-8x7b benefits the most under rigorous evaluation, with an average increase of 7% for each recall and TF1 score. GPT-3.5-Turbo seems to conserve its behavior, signaling an increase of only 2%. Surprisingly, GPT-4o exhibit a significant decrease in performance when it paraphrased the input prompts. On average, it lowered its performance by 33% for both metrics, with third level prompts being the worst affected. Nonetheless, it's promising to see the

Table 5
Results on Template Hard (TH) dataset, using hand-written prompts

Model		Mixtral	GPT-3.5-Turbo	GPT-4o	Total
Level	Metric	strict | flexible	strict | flexible	strict | flexible	strict | flexible
1	Recall	**0.25** | 0.41	0.37 | 0.42	0.53 | 0.54	0.38 | 0.46
	TF1	**0.28** | 0.48	0.45 | 0.53	0.62 | 0.64	0.45 | 0.55
2	Recall	0.08 | 0.30	0.46 | 0.51	0.54 | 0.72	0.36 | 0.51
	TF1	0.08 | 0.35	0.53 | 0.59	0.55 | 0.75	0.39 | 0.56
3	Recall	0.09 | 0.35	0.36 | 0.41	0.59 | 0.74	0.35 | 0.50
	TF1	0.10 | 0.46	0.45 | 0.52	0.60 | 0.75	0.38 | 0.58
4.1	Recall	0.15 | 0.47	**0.77** | **0.77**	0.71 | **0.76**	***0.54*** | ***0.67***
	TF1	0.15 | 0.49	**0.77** | **0.78**	0.71 | **0.75**	***0.54*** | ***0.69***
4.2	Recall	0.00 | **0.47**	0.75 | 0.76	**0.73** | 0.74	0.49 | 0.66
	TF1	0.00 | **0.56**	0.75 | 0.76	**0.73** | 0.75	0.49 | 0.69
Total Recall		0.11 | 0.39	0.54 | 0.57	0.62 | 0.70	0.42 | 0.56
Total TF1		0.12 | 0.47	0.59 | 0.65	0.64 | 0.73	0.45 | 0.61

Table 6
Results on Template Hard (TH) dataset, using model rephrased prompts

Model		Mixtral	GPT-3.5-Turbo	GPT-4o	Total
Level	Metric	strict | flexible	strict | flexible	strict | flexible	strict | flexible
1	Recall	**0.33** | 0.43	0.37 | 0.42	**0.51** | 0.54	0.40| 0.46
	TF1	**0.37** | 0.50	0.45 | 0.52	**0.60** | 0.64	0.47 | 0.55
2	Recall	0.14 | 0.35	0.43 | 0.47	0.28 | 0.72	0.28 | 0.51
	TF1	0.15 | 0.45	0.50 | 0.56	0.28 | 0.77	0.31 | 0.59
3	Recall	0.12 | 0.39	0.41 | 0.45	0.09 | 0.66	0.20 | 0.50
	TF1	0.13 | 0.46	0.50 | 0.55	0.09 | 0.65	0.23 | 0.55
4.1	Recall	0.07 | 0.47	**0.71** | **0.75**	0.55 | **0.76**	***0.44*** | 0.66
	TF1	0.07 | 0.48	**0.71** | **0.75**	0.55 | **0.75**	***0.44*** | 0.66
4.2	Recall	0.31 | **0.56**	0.70 | 0.74	0.18 | 0.73	0.40 | ***0.68***
	TF1	0.31 | **0.61**	0.70 | 0.74	0.18 | 0.74	0.40 | ***0.70***
Total Recall		0.19 | 0.44	0.52 | 0.57	0.32 | 0.68	0.34 | 0.56
Total TF1		0.20 | 0.50	0.57 | 0.63	0.34 | 0.71	0.37 | 0.61

open-source model enhancing its output by closely adhering to the provided system prompt.

Implicit Reasoning Poses Challenges for LLMs. Template Hard (TH) dataset contains cases where the LLM needs to understand that a given value is already an ID that references an existing instance in a KG or that a statement implies a specific relationship pertaining to a class. As concluded by the results presented in the tables from 3 to 6, under flexible metrics, Mixtral 8x7b achieves an recall of 56% and an TF1 score of 61% on the more difficult dataset, which is 17.5% lower than its performance on the easier one. GPT-3.5-Turbo narrows this margin, reducing from a peak recall and TF1 of 89% to 78% on Template Hard (TH). Same behaviour is observed with GPT-4o, as it falls from 91% recall and TF1 score to around 76%. Interestingly enough, Mixtral-8x7b yields its best scores at level 1 prompts, when strictly measured. Fig. 4 displays the differences in a compact form, based on each phrase's type. Thus, it shows a decrease in performance when phrases require extra reasoning steps, i.e. Implicit Information, compared to simple, direct ones i.e. Explicit Information. For instance, all models reduce their recall, on average, with 12%, and their TF1 score with 13%.

Table 7

Results on Templates Easy (TE) dataset, using hand-written prompts, on class types. Symbol "P" stands for *Project*, "E" for *Employee*, "S" for *Status*, and "N" for *None*

Model		Mixtral	GPT-3.5-Turbo	GPT-4o	Total
Class	Metric	strict | flexible	strict | flexible	strict | flexible	strict | flexible
P	Recall	0.22 | 0.59	0.60 | **0.66**	0.80 | 0.88	0.54 | 0.71
	TF1	0.23 | 0.67	0.67 | **0.76**	0.83 | 0.93	0.57 | 0.79
E	Recall	**0.25 | 0.68**	**0.63** | 0.66	***0.85 | 0.89***	***0.58 | 0.74***
	TF1	**0.23 | 0.70**	**0.68** | 0.72	***0.88 | 0.92***	***0.60 | 0.78***
S	Recall	0.18 | 0.47	0.53 | 0.62	0.60 | 0.68	0.44 | 0.59
	TF1	0.17 | 0.47	0.55 | 0.66	0.64 | 0.72	0.45 | 0.61
N	Recall	0.09 | 0.10	0.43 | 0.43	0.38 | 0.43	0.30 | 0.32
	TF1	0.09 | 0.13	0.43 | 0.43	0.36 | 0.44	0.29 | 0.33
Total Recall		0.21 | 0.54	0.59 | 0.63	0.75 | 0.83	0.52 | 0.68
Total TF1		0.22 | 0.61	0.65 | 0.72	0.78 | 0.87	0.55 | 0.73

Table 8

Results on Templates Hard (TH) dataset, using hand-written prompts, on class types. Symbol "P" stands for *Project*, "E" for *Employee*, "S" for *Status*, and "N" for *None*

Model		Mixtral	GPT-3.5-Turbo	GPT-4o	Total
Class	Metric	strict | flexible	strict | flexible	strict | flexible	strict | flexible
P	Recall	**0.14 | 0.50**	**0.64 | 0.68**	***0.77 | 0.85***	***0.52 | 0.68***
	TF1	**0.15 | 0.59**	**0.71 | 0.76**	***0.81 | 0.90***	***0.56 | 0.75***
E	Recall	0.13 | 0.38	0.43 | 0.51	0.27 | 0.40	0.28 | 0.43
	TF1	0.13 | 0.46	0.49 | 0.60	0.30 | 0.43	0.31 | 0.50
S	Recall	0.07 | 0.29	0.56 | 0.60	0.40 | 0.48	0.34 | 0.46
	TF1	0.07 | 0.33	0.61 | 0.66	0.42 | 0.50	0.37 | 0.50
N	Recall	0.03 | 0.03	0.19 | 0.20	0.18 | 0.22	0.13 | 0.15
	TF1	0.03 | 0.07	0.19 | 0.20	0.19 | 0.24	0.13 | 0.17
Total Recall		0.11 | 0.39	0.54 | 0.58	0.62 | 0.70	0.42 | 0.56
Total TF1		0.12 | 0.47	0.59 | 0.65	0.64 | 0.73	0.45 | 0.61

GPT-4o is More Consistent and Performant, While GPT-3.5-Turbo Achieves the Best Results on the Harder Dataset. Despite showing fluctuations in results when it rephrased the prompts, GPT-4o was the best overall model. Based on Table 3, on the TE dataset, under strict measurements, it had 75% recall and 78% for the TF1 score, almost four times more than Mixtral-8x7b and with 13.5% more than GPT-3.5-Turbo. We can interpret the results as GPT-4o is more reliable than the other two models, regardless of the prompt level. However, GPT-3.5-Turbo came close to it considering their top performances, being only 3% away from GPT-4o on the TE dataset, while surpassing it by 4% on the TH dataset, as we can notice on Table 5. Depending on the user's objectives, while considering the model's costs, the choice of the final model could vary.

Complex Class Types Do Not Imply More Difficult Reasoning. Analyzing both Tables 7 and 8, all models seem to perform better on the *Project* type, compared with the other three classes. It may be attributed to the inclusion of more difficult phrase types, combined with a notable lower amount of examples for the latter three. In spite of this difference, the results for the *Project* type are still significantly higher than for the other ones, although it requires the extraction of five relationships, compared with two for *Employee* and one for *Status*. For example, under the flexible paradigm, the average recall and TF1 score are 70% and 77% for the *Project* class, while for *Employee*, the models only achieve 59% and 64%. This suggests a potential hypothesis regarding LLM behavior when handling complex versus simpler classes. Finally, the recall and TF1 score on the *Status* class are 53% and 56%, respectively

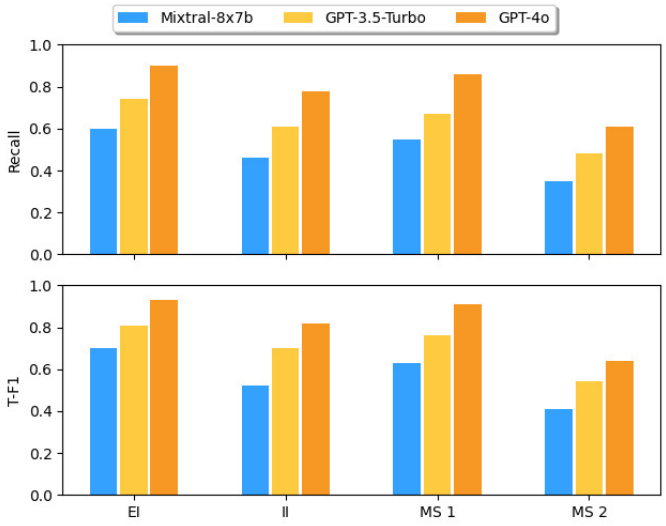

Fig. 4. Recall and TF1 for each model per phrase type under the flexible paradigm on both datasets using hand-written prompts. Symbol "EI" stands for *Explicit Information*, "II" for *Implicit Information*, "MS1" and "MS2" for *"Misleading Information types 1 or 2"*

Table 9

In-depth analysis of the "*Misleading Information type 2*" (MS2) phrase type (from Table 1). Phrases include OOD class types, basic tasks "*w/o Insert*" intent, or ad-hoc values for the target instance' relationships ("WV").

Model	OOD	w/o Insert	WV
Mixtral 8x7b-instruct-v0.1	0.00	0.28	0.57
GPT 3.5-turbo-0125	0.01	1.00	0.64
GPT-4o	**0.04**	**1.00**	**0.86**

Table 10

Results, under the flexible paradigm, of hand-written prompts by each model on the phrases that include *Employee* instances referenced by an ID or their *hasRole* relationship value instead of the *hasName* one; only available in the TH dataset.

Model	with ID reference		with *Role* reference	
	Recall	TF1	Recall	TF1
Mixtral 8x7b-instruct-v0.1	0.30	0.38	0.47	0.54
GPT 3.5-turbo-0125	0.45	0.52	0.68	0.75
GPT-4o	**0.46**	**0.54**	**0.83**	0.88

- 6% and 8% lower than for the *Employee* class. This might indicate that LLMs leverage internal knowledge for task resolution, particularly since *Employee* instances often involve familiar person names and roles, which are more likely included during LLM training, unlike the more variable nature of *Status* instances names (e.g., 'in-progress').

LLMs Appear to Adhere to the Ontology. While the results in Tables 3-8 and Fig. 4 demonstrate strong performance across various prompt levels, classes, and phrase types, suggesting that LLMs may grasp the provided ontology, closer analysis of the misleading information type 2 category (MS2) category from Fig. 4 raises concerns. This category had the lowest scores, with an average recall of 48% and an TF1 score of 53% across the three models. Although these results might seem acceptable at first glance, a deeper look at the phrase types reveals flaws in LLMs' behavior. All models performed reasonably well when encountering ad-hoc values for the target instance' relationships, reaching 86% recall using GPT-4o, as we can notice in Table 9.

However, phrases involving basic tasks without the Insert intent (e.g., 'generate all the reports you have') posed an issue for Mixtral-8x7b, which attempted to extract triples instead of outputting 'None'. The most significant

challenge was presented by out-of-distribution (OOD) class types, such as the example in last row of Table 1, where none of the models followed the prompt or ontology. Instead of verifying the detected type against the ontology and outputting 'None,' 98% of the time they incorrectly treated it as valid. This suggests that LLMs do not truly reason but are highly adept at mapping input text to target output when the cases are general enough.

Top-tier LLMs Effectively Address Grammatical Errors. Fig. 4 highlights the Misleading information type 1 (MS1) category, where phrases contain misspelled words, as shown in the third row of Table 1. While Mixtral-8x7b achieves only 55% recall and a 63% for TF1 score, GPT models handle most errors and even correct known class names (e.g., 'Porject' to 'Project'). For instance, GPT-4o reaches 86% recall and a 91% for TF1 score under the flexible paradigm.

The Underlying Semantics of Words Pose a Challenge for LLMs. The *Project* and *Employee* classes are linked through the *hasManager* relationship, and most test phrases reference the Employee's name, requiring the creation of an additional *Employee* instance, as described in subsection 3.4. Such tasks are trivial for high-performant models, as names can be linked with persons, which can be seen as a supertype for the *Employee* class. With their complex training schedule, it is highly probable their dataset contained such cases. However, when we start referencing such instances by their role (i.e. a job type), their performance starts to decline, though not drastically. As shown in Table 10, GPT-4o maintains 83% recall and an 88% TF1 score, close to its overall performance (85% recall, 90% for TF1 in Table 8, *Project* class type). However, performance drops sharply when using terms likely absent from training, such as an ID (e.g., Employee123), with GPT-4o achieving just 46% recall and 54% for TF1. This suggests that referencing class instances with unusual terms, like IDs, challenges LLMs to grasp deeper semantic relationships.

In summary, KGC remains a challenging task for LLMs under Zero-Shot prompting. As models become better, they performance tend to increase, while shifting the focus on optimizing the costs. Moreover, when checking their intermediate reasoning steps, they show lack of ability to follow the provided ontology. The open-source model has difficulties in conforming to the required output format. However, One-Shot contexts give promising results as LLMs excel in emulating a provided example. This implies that a less resource-intensive Few-Shot training approach could potentially boost performances, with a focus on techniques like Retrieval-Augmented-Generation to select more suitable examples within a given prompt. Another plus is their ability to enhance their inner knowledge to detect some implicit relationships from the input text. Nevertheless, as suggested by Fill et al. [3], presently we may use such LLMs as helpful assistants for solving such tasks, rather than ultimately faithful extractors in a pipelined system.

5. Conclusion

The proposed experiments showcases the ability of three leading LLMs, namely Mixtral-8x7b-Instruct-v0.1, GPT-3.5-Turbo-0125 and GPT-4o, in tackling the Knowledge Graph Construction task. Using both hand-written and model-rephrased prompts, we incorporated various prompt engineering techniques, such as In-Context Learning or Chain of Thought, focusing on Zero- and One-Shot contexts. Metrics measurement enabled the evaluation of the LLMs for strictly following the given prompt, as well as the their flexibility in producing useful output to be considered in post-processing steps. The results obtained from two distinct datasets tailored to various reasoning challenges highlight the LLMs strengths and weaknesses. These include their adaptability in Zero- or One-Shot scenarios and their utilization of internal knowledge to deduce implicit reasoning steps. However, they still lack self-awareness, not being able to adhere to explicit guidelines in the given prompt, or fully understand and exploit the considered ontology.

Additionally, we proposed two personalized datasets capable of assessing both the models' ability to solve the Knowledge Graph Construction task and their potential integration with task oriented dialogue systems simultaneously and a flexible measurement procedure to measure the capacity of the LLM to give logically correct results, but in an approximate format.

Future work will prioritize the integration of additional LLMs for testing, facilitated by our interface's seamless incorporation of new endpoints. Moreover, we plan to test the possible influence of placing the system prompt at the end of the message, after the input text, to mitigate long-context memory issues. Lastly, we plan to move from single turns to a dialogue context, where the extraction happens as a discussion between a user and the LLM.

References

[1] A. Bordes, N. Usunier, A. García-Durán, J. Weston and O. Yakhnenko, Translating Embeddings for Modeling Multi-relational Data, in: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, C.J.C. Burges, L. Bottou, Z. Ghahramani and K.Q. Weinberger, eds, 2013, pp. 2787–2795. https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html.

[2] H. Chen, X. Liu, D. Yin and J. Tang, A Survey on Dialogue Systems: Recent Advances and New Frontiers, *ACM SIGKDD Explorations Newsletter* **19**(2) (2017), 25–35. doi:10.1145/3166054.3166058.

[3] H. Fill, P. Fettke and J. Köpke, Conceptual Modeling and Large Language Models: Impressions From First Experiments With ChatGPT, *Enterp. Model. Inf. Syst. Archit. Int. J. Concept. Model.* **18** (2023), 3. doi:10.18417/EMISA.18.3.

[4] H. Ghanem and C. Cruz, Fine-Tuning vs. Prompting: Evaluating the Knowledge Graph Construction with LLMs, in: *3rd International Workshop on Knowledge Graph Generation from Text (Text2KG), Co-located with the Extended Semantic Web Conference (ESWC 2024), May 26–30, 2024, Hersonissos, Greece*, S. Tiwari, N. Mihindukulasooriya, F. Osborne, D. Kontokostas, J. D'Souza, M. Kejriwal, M.A. Pellegrino, A. Rula, J.E.L. Gayo, M. Cochez and M. Alam, eds, CEUR Workshop Proceedings, Vol. 3747, CEUR-WS.org, 2024. https://ceur-ws.org/Vol-3747/text2kg_paper7.pdf.

[5] S. Guo, Q. Wang, L. Wang, B. Wang and L. Guo, Jointly Embedding Knowledge Graphs and Logical Rules, in: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, J. Su, X. Carreras and K. Duh, eds, The Association for Computational Linguistics, 2016, pp. 192–202. doi:10.18653/V1/D16-1019.

[6] J. Han, N. Collier, W.L. Buntine and E. Shareghi, PiVe: Prompting with Iterative Verification Improving Graph-based Generative Capability of LLMs, in: *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, L. Ku, A. Martins and V. Srikumar, eds, Association for Computational Linguistics, 2024, pp. 6702–6718. doi:10.18653/V1/2024.FINDINGS-ACL.400.

[7] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, S. Kirrane, J.E.L. Gayo, R. Navigli, S. Neumaier, A.N. Ngomo, A. Polleres, S.M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab and A. Zimmermann, *Knowledge Graphs*, Synthesis Lectures on Data, Semantics, and Knowledge, Morgan & Claypool Publishers, 2021. doi:10.2200/S01125ED1V01Y202109DSK022.

[8] V.I. Iga and G.C. Silaghi, Leveraging BERT for Natural Language Understanding of Domain-Specific Knowledge, in: *25th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2023, Nancy, France, September 11-14, 2023*, IEEE, 2023, pp. 210–215. doi:10.1109/SYNASC61333.2023.00035.

[9] V.I. Iga and G.C. Silaghi, Ontology-Based Dialogue System for Domain-Specific Knowledge Acquisition, in: *Information Systems Development: Organizational Aspects and Societal Trends (ISD2023 Proceedings), Lisbon, Portugal, 30 August - 1 September 2023*, A.R. da Silva, M.M. da Silva, J. Estima, C. Barry, M. Lang, H. Linger and C. Schneider, eds, Instituto Superior Técnico / Association for Information Systems, 2023. doi:10.62036/ISD.2023.46.

[10] V.I. Iga and G.C. Silaghi, Assessing LLMs Suitability for Knowledge Graph Completion, in: *Neural-Symbolic Learning and Reasoning - 18th International Conference, NeSy 2024, Barcelona, Spain, September 9-12, 2024, Proceedings, Part II*, T.R. Besold, A. d'Avila Garcez, E. Jiménez-Ruiz, R. Confalonieri, P. Madhyastha and B. Wagner, eds, Lecture Notes in Computer Science, Vol. 14980, Springer, 2024, pp. 277–290. doi:10.1007/978-3-031-71170-1_22.

[11] S. Ji, S. Pan, E. Cambria, P. Marttinen and P.S. Yu, A Survey on Knowledge Graphs: Representation, Acquisition, and Applications, *IEEE Trans. Neural Networks Learn. Syst.* **33**(2) (2022), 494–514. doi:10.1109/TNNLS.2021.3070843.

[12] A.Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D.S. Chaplot, D. de Las Casas, E.B. Hanna, F. Bressand, G. Lengyel, G. Bour, G. Lample, L.R. Lavaud, L. Saulnier, M. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T.L. Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix and W.E. Sayed, Mixtral of Experts, *CoRR* **abs/2401.04088** (2024). doi:10.48550/ARXIV.2401.04088.

[13] H. Khorashadizadeh, N. Mihindukulasooriya, S. Tiwari, J. Groppe and S. Groppe, Exploring In-Context Learning Capabilities of Foundation Models for Generating Knowledge Graphs from Text, in *CEUR Workshop Proceedings*, Vol. 3447, S. Tiwari et al., ed., CEUR-WS.org, 2023, pp. 132–153. https://ceur-ws.org/Vol-3447/Text2KG_Paper_9.pdf.

[14] N. Lao and W.W. Cohen, Relational retrieval using a combination of path-constrained random walks, *Mach. Learn.* **81**(1) (2010), 53–67. doi:10.1007/S10994-010-5205-8.

[15] N. Mihindukulasooriya, S. Tiwari, C.F. Enguix and K. Lata, Text2KGBench: A Benchmark for Ontology-Driven Knowledge Graph Generation from Text, in: *The Semantic Web - ISWC 2023 - 22nd International Semantic Web Conference, Athens, Greece, November 6-10, 2023, Proceedings, Part II*, T.R. Payne, V. Presutti, G. Qi, M. Poveda-Villalón, G. Stoilos, L. Hollink, Z. Kaoudi, G. Cheng and J. Li, eds, Lecture Notes in Computer Science, Vol. 14266, Springer, 2023, pp. 247–265. doi:10.1007/978-3-031-47243-5_14.

[16] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang and X. Wu, Unifying Large Language Models and Knowledge Graphs: A Roadmap, *IEEE Trans. Knowl. Data Eng.* **36**(7) (2024), 3580–3599. doi:10.1109/TKDE.2024.3352100.

[17] F. Polat, I. Tiddi and P. Groth, Testing Prompt Engineering Methods for Knowledge Extraction from Text, *Semantic Web* **accepted** (2024).
[18] S.K.K. Santu and D. Feng, TELeR: A General Taxonomy of LLM Prompts for Benchmarking Complex Tasks, in: *Findings of the ACL: EMNLP 2023, Singapore, 2023*, H. Bouamor et al., ed., ACL, 2023, pp. 14197–14203. doi:10.18653/V1/2023.FINDINGS-EMNLP.946.
[19] M. Trajanoska, R. Stojanov and D. Trajanov, Enhancing Knowledge Graph Construction Using Large Language Models, *CoRR* **abs/2305.04676** (2023). doi:10.48550/ARXIV.2305.04676.
[20] X. Wei, X. Cui, N. Cheng, X. Wang, X. Zhang, S. Huang, P. Xie, J. Xu, Y. Chen, M. Zhang, Y. Jiang and W. Han, ChatIE: Zero-Shot Information Extraction via Chatting with ChatGPT, *CoRR* **abs/2302.10205** (2023). doi:10.48550/ARXIV.2302.10205.
[21] W. Xiong, T. Hoang and W.Y. Wang, DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning, in: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, M. Palmer, R. Hwa and S. Riedel, eds, Association for Computational Linguistics, 2017, pp. 564–573. doi:10.18653/V1/D17-1060.
[22] W. Xiong, M. Yu, S. Chang, X. Guo and W.Y. Wang, One-Shot Relational Learning for Knowledge Graphs, in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, E. Riloff, D. Chiang, J. Hockenmaier and J. Tsujii, eds, Association for Computational Linguistics, 2018, pp. 1980–1990. doi:10.18653/V1/D18-1223.
[23] J. Zhang, B. Chen, L. Zhang, X. Ke and H. Ding, Neural, symbolic and neural-symbolic reasoning on knowledge graphs, *AI Open* **2** (2021), 14–35. doi:10.1016/J.AIOPEN.2021.03.001.
[24] 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. Nie and J. Wen, A Survey of Large Language Models, *CoRR* **abs/2303.18223** (2023). doi:10.48550/ARXIV.2303.18223.
[25] Y. Zhu, X. Wang, J. Chen, S. Qiao, Y. Ou, Y. Yao, S. Deng, H. Chen and N. Zhang, LLMs for knowledge graph construction and reasoning: recent capabilities and future opportunities, *World Wide Web (WWW)* **27**(5) (2024), 58. doi:10.1007/S11280-024-01297-W.