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 studies have demonstrated that Large Language Models (LLMs) can perform various Knowledge Graph-related tasks, including Knowledge Graph Construction, even in Zero- and Few-Shot settings. However, LLMs are prone to hallucinating information and producing non-deterministic outputs, which can result in flawed reasoning, even when the answers appear to meet user expectations. This unpredictability limits their integration into automated natural language processing pipelines, such as those used in chatbots or Task-Oriented Dialogue systems. To explore the potential and limitations of LLMs in Knowledge Graph tasks, we evaluate three prominent models, namely Mixtral-8x7b-Instruct-v0.1, GPT-3.5-Turbo-0125, and GPT-4o, on constructing static knowledge graphs. Our approach uses prompts based on the TELeR taxonomy in Zero- and One-Shot scenarios, within the context of a Task-Oriented Dialogue system. Additionally, we propose a flexible evaluation framework that captures all usable information generated by the models, alongside traditional strict metrics, and introduce TODSet, a dataset tailored to gauge the performance of LLMs on knowledge graph-related tasks. Our findings suggest that, with well-designed prompts containing sufficient detail and examples, LLMs can effectively contribute to Knowledge Graph Construction tasks.

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 [8]. 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 the ability to engage in natural language dialogues with human users. Unlike chatbots, TOD systems aim to solve user-specific tasks within certain domains, while using a given ontology that serves as their task-related knowledge. Typically, such systems are composed of four modules: natural language understanding (NLU), dialogue state tracking (DST), dialogue policy (POL) and natural language generation (NLG) [2]. The first two aim at understanding the user's utterance, while the latter ones focus on generating a system response. An example of such system is presented in Figure 1. State-of-the-art (SOTA) methods leverage neural network-based models' ability to handle all tasks simultaneously in an end-to-end fashion [3, 22].

In our previous work [10], the objective was to tackle the knowledge graph construction (KGC) and reasoning (KGR) tasks by developing a TOD system that could extract relevant information from a conversation with a human user guided by a predefined ontology, and perform four key operations known as Create-Retrieve-Update-Delete

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

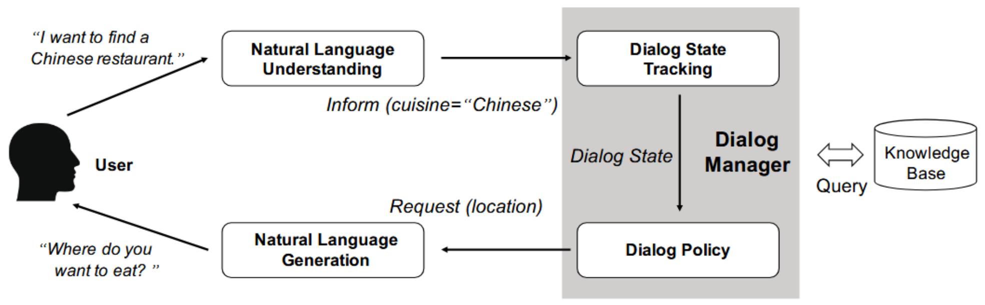
"I want to find a Chinese restaurant."
Natural Language Understanding
Inform (cuisine="Chinese")
Dialog State Tracking
Dialog State
Dialog Manager
Query
Knowledge Base
User
Request (location)
Natural Language Generation
Dialog Policy
"Where do you want to eat?"

Fig. 1. General framework of a pipeline task-oriented dialog system [26].

(CRUD) on a domain-specific KG. The acronym CRUD refers to the four basic operations that can be executed against persistent storage, such as relational or object databases, or other types of knowledge base such as KGs, to create, maintain, or update them. To keep track of the conversation's context, a discourse-specific KG was maintained, which enabled concurrent threads of conversation within a single discourse, and also acted as a proxy which validated data before persisting it into the main knowledge graph.

Nonetheless, our TOD system [10] relied on input text template-matching rules, constraining the authenticity of dialogues and hindering its adaptability to process novel concepts beyond the predefined ontology. Hence, a subsequent study [9] explored fine-tuning BERT, a pre-trained neural network model, to infer user intent and extract relevant entities directly from the input, eliminating the need for rigid templates. While integrating a deep learning model into the TOD architecture showed promising results, it did not fully overcome the earlier limitations.

Therefore, in our current work, we study the use of LLMs to solve the KGC task, in the context of a TOD system. The literature [7, 16, 17, 25, 28] identified a potential synergy between KGs and LLMs, as KGs can enrich LLMs by providing external knowledge for inference and explainability, while LLMs, in turn, can address KG-related tasks through natural language prompts. Our goal is to leverage LLMs to extract facts from natural text and automatically integrate these facts into the processing pipeline of a TOD system.

Our experiments explore LLMs for static KG contexts. Three well-established LLMs are used: Mixtral-8x7b-instruct-v0.1[1] [13], GPT-3.5-Turbo-0125[2], alongside one of the most advanced GPT models, the GPT-4o[3] version, each possessing different properties. Communicating with such models involves the use of prompts, which are instructions in natural language structured in a way that enables the model to accurately interpret the user's intent. To test LLMs capabilities of solving the aforementioned KGC task, we develop multiple hand-written prompts and also ask each model to rephrase them to increase their clarity. Each prompt belongs to a level defined by the TELeR taxonomy [19], including well-established 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 introduce TODSet, a two-fold dataset, by extracting sample phrases from the training phase of our TOD system [9]. The two splits have a different level of difficulty, such that the harder one includes test cases which require reasoning steps that are not explicitly mentioned in the prompts. This approach allows us not only to 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 [5, 7] of each LLM, 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 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 of performing such task. Performance is measured within two paradigms: strict and flexible, shedding light on the challenges encountered during post-processing. (ii) A novel and

[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

flexible metric is designed to positively evaluate any information generated by the LLM that can be automatically integrated into the TOD system pipeline with the aid of additional post-processing steps. (iii) We introduce TODSet, a customized two-fold dataset tailored to gauge the performance of LLMs on the KGC task, featuring varying levels of difficulty. (iv) The feasibility of integrating such models into a domain-specific ontology-enhanced TOD system is investigated by extracting and utilizing test phrases specific to its context.

This paper extends our previous work [11] by more precisely defining the preliminary conditions, updating the related work section to include recent and relevant studies, formally introducing the flexible measurement paradigm, and evaluating performance on an additional dataset with a more complex ontology. It also presents more detailed results, enabling deeper analysis, including insights into how well the tested LLMs handle tasks of varying 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 experiments, Section 4 presents and discusses the results, while Section 5 wraps up the paper with concluding remarks.

2. Related Work

Knowledge Graph Construction 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) co-reference resolution, and 3) relation extraction. Recent methods also include end-to-end KGC, which constructs a complete knowledge graph in a single step with the help of LLMs [17].

Many non-LLM techniques address the task by solving the first three stages rather separately. However, these stages could also be combined into a single process, known as Knowledge Graph Completion. This task aims to deduce the absent information within a specified KG [17], drawing from input text or pre-existing knowledge. Ji et al. [12] present multiple solutions for KG Completion utilizing embedding-based models like TransE [1], relation path reasoning exemplified by the Path-Ranking Algorithm [15], reinforcement-learning path finding [23], rule-based reasoning such as KALE [6], and meta relational learning [24] utilizing R-GCN or LSTM. Similar insights are shared by Zhang et al. [25], 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. [21] advocate for a multi-stage dialogue with ChatGPT to extract pertinent information from input texts, based on a predefined schema. The authors solve the KGC task by dividing it into Named Entity Recognition, Relation Extraction and Event Extraction. Zhu et al. [28] experiment with ChatGPT and GPT4 for KGC in the pipeline manner, determining that while these models 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. The authors 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. [7] 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 the key to preserve the correctness of LLMs. Khorashadizadeh et al. [14] 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. [20] experiment with a specialized pre-trained model (REBEL) and ChatGPT to automate the extraction of KGs from news articles, concluding that ChatGPT, when prompted adequately using enough information and guidelines, can solve the task with promising results. Ghanem et al. [5] evaluate various LLMs using prompts under Zero- and Few-Shot paradigms, with or without fine-tuning them beforehand. The authors report metrics including TF1, GF1, and Graph Edit Distance (GED) introduced in [7], 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, our research increases the number of textual inputs, expanding the generality of our conclusions, sharing similarities with [16, 18]. Mihindukulasooriya et al. [16] 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 an LLM performance from a different perspective, while testing models of various types and sizes. Polat et al. [18] 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 Retrieval Augmented Generation (RAG) approaches, while the evaluation of the output is done using SPARQL queries to Wikidata. In our paper, we intend to assess the performance of the LLMs on the KGC task solely based on the user's input text, without helping the LLM with additional contextual information.

Consequently, our experiments test the capacity of a proprietary LLM – namely GPT, with two versions: GPT-3.5-Turbo-0125 and GPT-4o on the KGC task. Furthermore, an open source LLM is included - Mixtral-8x7b-Instruct-v0.1 [13], to facilitate research on open-source models, given their greater adaptability and cost-effectiveness compared to proprietary alternatives. Another difference from the literature mentioned above is that our prompts are more diverse and easier to follow, ranked according to the TELeR taxonomy [19]. We introduce flexible metrics to gauge 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 throughout this paper. We describe preliminary definitions of key concepts, prompt engineering steps, the ontologies used to anchor the knowledge of the LLM, the format and distribution of the datasets, and 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 is a type of machine learning model capable of processing and understanding texts, making it highly effective for Natural Language Processing (NLP) tasks. Recently, deep learning models based on the transformer architecture have become very effective. This type of models consist of billions of parameters, trained on vast amounts of data in order to understand the semantics of words in texts. Thus, such models are classified into three types: 1) encoder-only, 2) encoder-decoder, and 3) decoder-only [17]. 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 under the format of triples that are 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. 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.

3.2. Prompt engineering

The input text that is fed to an LLM is usually referred to as a prompt. It may have several parts, such as the system message, which offers task-specific guidelines to the model, the input data that should be processed, and optionally some examples of how to solve the task against some different input text. Three important paradigms are usually employed when designing prompts [27]: Zero-, One- and Few-Shot. Zero-Shot does not add any example to the input prompt, testing the model's capability to understand and follow the provided guidelines. One-Shot includes exactly one example of how the task should be solved against some different input data, serving as a blueprint to the model. Intuitively, Few-Shot refers to the addition of multiple relevant examples such that the tested model has a wider perspective on how the task can be solved in different scenarios.

After selecting the paradigm, one should decide about the prompting technique [27]. The current work employs three main approaches, as follows: Direct prompting refers to a prompt that only comprises the task description and the input to work on, In-Context Learning adds relevant examples of solutions to the given task on different input data (One or Few-Shot), and Chain of Thought expands the prompt with a step-by-step reasoning process that breaks down the main task into smaller, more manageable ones that sequentially lead to the desired solution.

To test the capacity of a model to solve a task, our work follows the guidelines of Santu et al. [19] by assigning a level to each version of a prompt. Specifically, we utilize levels 1 through 4, while the last one is further divided into 4.1 and 4.2 to accommodate both ICL and COT variations of the prompt.

The first level prompt is exemplified in Figure 2. It sets the model's role as a KG expert, followed by instructions on the provided ontology. Subsequently, the task at hand is outlined, along with formatting guidelines. Specifically, each instance has to be identified by an ID composed of the name of its class type concatenated with the "1" digit. An alternative to this approach would be to directly use as ID the words in the text that triggered the extraction. However, this would not guarantee that the model successfully followed and understood the provided ontology, as opposed to our approach, where each instance type can be checked by analyzing its ID. In addition, the "1" digit is chosen to simplify the creation of the identifier, allowing the model to focus on the semantic content of each triple. Another important aspect mentioned within the prompt is that the model should create different instances for every detected concept and reference them through their IDs. Finally, the required output pattern is presented and the target ontology is attached. Level 2 adds a directive about the addition of the *rdf:type* relationship. Although such triples should also be extracted from the first level, this was only explicitly stated from this level to test whether model's are capable of extracting triples that are implicitly stated. It then evolves into level 3, where the only change is the text's format, as it is transformed 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 whether the target text has extractable triples or not, either an example with no output triples or one with existing golden labels is included. Examples of prompts built with each of these techniques are available in appendix A[4] in

[4]https://github.com/IonutIga/LLMs-for-KGC/tree/main/Appendixes/Appendix%20A

'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. 2. The level 1 system prompt.

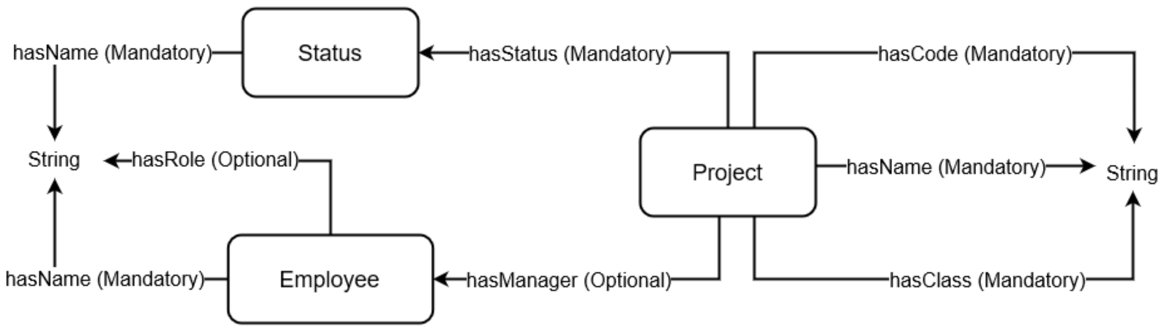

Fig. 3. TODSet ontology.

the paper's repository. Moreover, as suggested in [17], each model is asked to rephrase the provided hand-written system messages in such a way that it better understands the given guidelines. Therefore, two types of prompts are available: hand-written and model-rephrased.

3.3. Format and distribution of the datasets

To test the models capacity of solving the KGC task, the *TODSet* dataset is introduced, which features input texts with different levels of difficulty. Additionally, inspired by [16], we adapt their DBpedia-WebNLG sports ontology and dataset to our format, to further test models in a more complex scenario. Throughout the paper, this dataset will be referred to as the *FootballSet*.

TODSet base itself on the ontology introduced in our previous research [10] and described in Figure 3. 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.

The input phrases are sampled from the training schedule of the TOD system developed in [10], where the user's purpose was to solve different CRUD operations based on the concepts described in the ontology presented above. Only phrases that correspond to the intent *Create* (*Insert*) were selected, as they bring in novel information relevant for the KGC task. However, the subject of some phrases was further changed to an out-of-distribution (OOD) class type, and some phrases do not even include the intent *Insert* (labeled as *w/o Insert*). These phrases do not contain extractable triples and are further labeled as having *None* class type, to test whether a model actually follows the content of the predefined ontology. The texts are further classified based on their expressivity, as conveying: (i) *explicit information* where intent, class type, associated relationships, and values are clearly articulated and (ii) *implicit information* where additional reasoning steps are required to identify the necessary details. In addition, some phrases are labeled as misleading, where their content is altered to emulate real-world scenarios in which texts contain (iii) *grammatical errors* or (iv) unknown vocabulary words or ontology concepts (i.e. the *None* class type). Unknown vocabulary words are such constructions that have no equivalent in the real world. For example, as the relationship *hasClass* requires a programming language, we include words that do not resemble existing ones, such as Dandy, Laclut etc. These phrases are also labeled as *UVW* (which stands for Unknown Vocabulary Words).

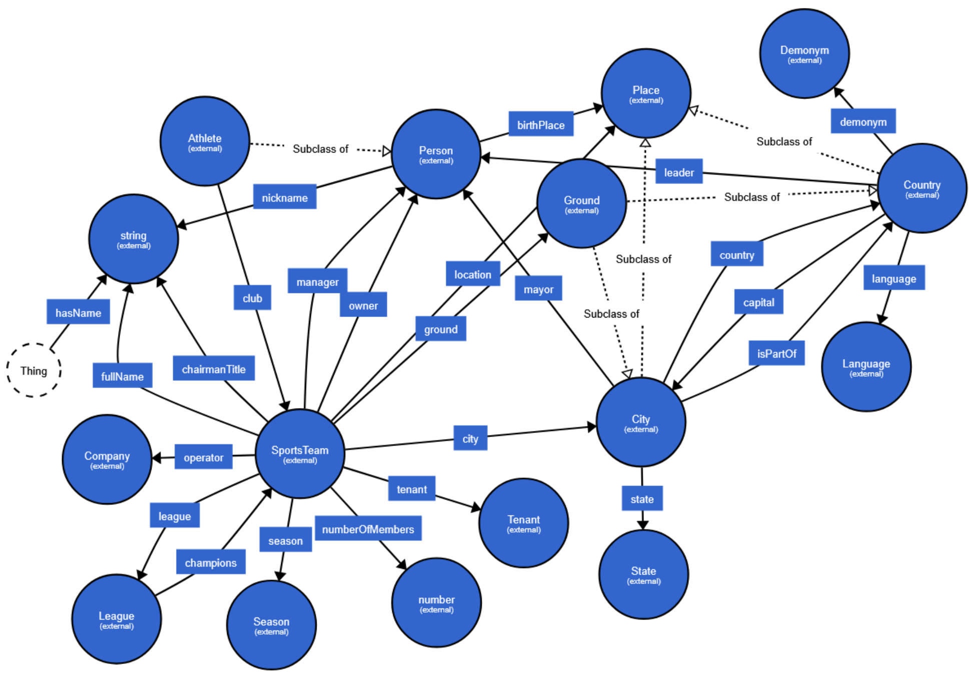

Fig. 4. FootballSet ontology.

Table 1

TODSet - input text 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)

Table 1 presents examples from each category. More details on how texts are classified are available in the project repository[5]. In the end, TODSet is divided into two: *Templates Easy (TE)* and *Templates Hard (TH)*. The first split includes explicit and misleading text types with a lower number of implicit ones, while the second one benefits of an increased overall difficulty, as well as more implicit-type texts. Table 2 presents the distribution of texts by class type, in each split.

FootballSet includes texts about football clubs, players, managers, leagues, or countries. It is based on an ontology provided by Mihindukulasooriya et al. [16], which was slightly modified to fit our framework. It comprises 14 classes and 24 relationships among them, visible in Figure 4. A number of 48 input texts were sampled from their train, test, and validation files, while the other 27 were manually created. This set only contains explicit and implicit information phrase types, as we wanted to keep as much as possible from the original set, without altering their textual content. Examples can be seen in Table 3, while Table 4 presents the distribution of texts by class type.

Each text is associated with a set of golden labels, which are the target triples that can be extracted from the input text. Additionally, we introduce two new types of triples: alternative and false positive accepted ones. Both are useful to underline the power of the flexible metrics paradigm, introduced in subsection 3.4.

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

Table 2
TODSet - distribution of texts by class type

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

Table 3
FootballSet - input text examples and their types

Input phrase example	**Input phrase type**
Peter Stöger is manager of FC Köln which has 50000 members and participated in the 2014 season.	explicit information
Part of Bundesliga, Wolfsburg is led by Oliver Glasner, under the investments of Volkswagen.	implicit information

Table 4
FootballSet - distribution of texts by class type

Datasets	Class type						Total
	SportsTeam	Person	Athlete	Country	League	Place	
FootballSet (FS)	44	10	9	5	5	2	75

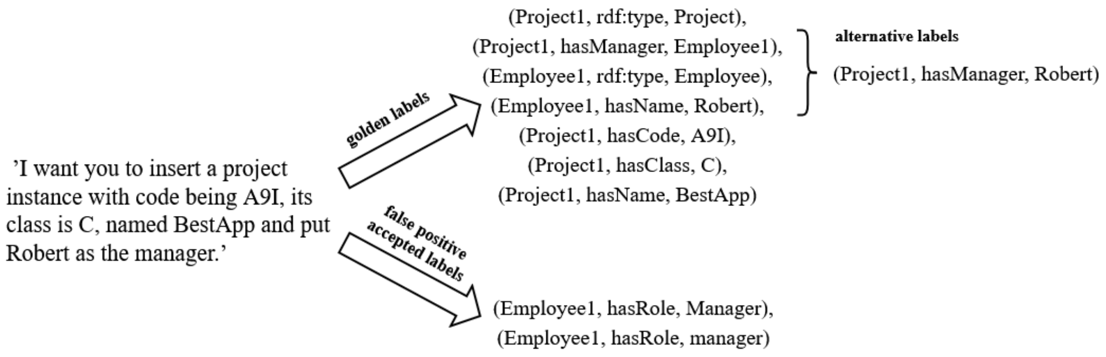
'I want you to insert a project instance with code being A9I, its class is C, named BestApp and put Robert as the manager.'
golden labels
(Project1, rdf:type, Project),
(Project1, hasManager, Employee1),
(Employee1, rdf:type, Employee),
(Employee1, hasName, Robert),
(Project1, hasCode, A9I),
(Project1, hasClass, C),
(Project1, hasName, BestApp)
alternative labels
(Project1, hasManager, Robert)
false positive accepted labels
(Employee1, hasRole, Manager),
(Employee1, hasRole, manager)

Fig. 5. An example of an input text and associated extractable triples.

The alternative triples set includes facts that convey the same semantic meaning as some target triples but do not adhere to the required format. For instance, as noted in subsection 3.2, when a relationship exists between two concepts, both must be identified by an ID. However, if the model uses text fragments from the input instead of IDs to refer to these concepts, the resulting triple can still be considered correct, since it refers to the same entities, even though it does not follow the specified format. An example can be visualized in Figure 5. The text mentions a *Project* instance that is related to an *Employee* one through the *hasManager* relationship. Therefore, the model has to create IDs for each one and construct a triple such as (*Project1*, *hasManager*, *Employee1*). However, it might only generate an ID for the main instance of the text (i.e. the project) and link it with the employee using the given name, such as (*Project1*, *hasManager*, *Robert*). Semantically, both triples encapsulate the same information, and with the right post-processing steps, one can accept both forms of extraction.

The false positive accepted set allows a predicted triple that is outside the golden labels (i.e. a false positive) to be considered as correct if it follows the provided ontology and no additional background information contradicts it. For example, in Figure 5, the prompt does not specifically mention what is Robert's role (i.e., job), but one could infer from the input text that he is the manager, as he is part of the *hasManager* relationship. 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 LLMs to exhibit such variability.

3.4. Metrics

Several metrics are used to measure a model's performance, such as: precision, recall, and F1 score as TF1 (Triple-matching F1) [5, 7]. Our evaluation is conducted under two paradigms: strict and flexible metric measurements. The following subsection outlines the specific metrics used in this study.

For each text, the LLM produces a list of m triples. Let us denote it by $PT = \{s_i,\ i = \overline{1,m}\}$. 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 strict criterion, metrics are calculated in a standard text extraction manner, by counting how many predicted triples from PT are among the golden ones GL, adhering to the provided guidelines. This approach enables the assessment of the model's ability to exactly follow the given prompt and process the input text so that its results can be directly used in subsequent pipelines.

However, LLMs may not follow the required format, which directly discards its output under strict criterion without any evaluation of the predicted triples content. Moreover, the KGC task is inherently difficult, as the extraction of triples from a given text necessitates a sequence of reasoning steps such as the identification of possible entities and connections between them, while some information may not even be explicitly mentioned. Additionally, as described in subsection 3.3, some triples may be deemed true if no background information is provided, and others might be counted as correct even though they do not follow the required format, but encapsulate similar semantic information with some of the target ones. Therefore, we introduce the flexible metric paradigm that allows the LLM to produce mistakes which can be corrected in post-processing steps, while imposing certain user-defined penalties. Using this approach, models that might not be as precise as more elaborate ones but require fewer resources can be positively evaluated and taken into consideration to solve the KGC task.

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 $h_i = 0$ if the triple is not found in GL or is unusable in further post-processing steps.

Under the flexible measurement paradigm, certain penalties are included 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.

It is important to note that any penalties considered in this work fit our specific prompt formulation. We emphasize that neither the penalties nor their values are fixed and can be adapted according to a user's specific needs.

Format Penalties at the level of the whole output. The user may request that the LLM final output follows a global format. In our case, the prompt demands a reply that is composed solely of a list of triples. Therefore, we consider a penalty of 2.5% for outputs with multiple lists and a penalty of 7.5% if the LLM produces additional text that was not requested.

Format Penalties at the triple level. The prompt asks not to include the full IRI of an entity (i.e. without the namespace) to reduce the number of tokens and the possibility of confusion between the same concepts, shifting the focus to the semantic content of the output. Each addition of IRIs is penalized with 1%. Finally, the prompt specifies that the output format of each triple is a JSON object with three properties, thus for any triple that does not follow the required format the model is penalized with 10%.

Content Penalties refer to penalties related to the information encapsulated within a triple. Specifically, the model is asked to construct a simple ID for each extracted entity - the capitalized name of its class concatenated with "1". A deeper analysis of models output underlined a tendency to replace the number "1" with another single digit. Thus, if altering the final digit of a predicted identifier to "1" signifies correctness of the whole 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 be rewarded for its partial correctness. However, this flexibility only applies to the validity of a constructed ID(s). Any other type of error is not allowed since it would alter the factuality of the implied information.

As previously noted, certain alternative triples to the designated correct ones can be regarded as valid. Specifically, in Figure 5, concerning the relationship marked as *hasManager* between a *Project* and an *Employee* instance, if a model predicts that the value of the object is directly the employee's name, instead of creating an *Employee* instance and assigning its type and name, it will count as correct. However, there is an implicit penalty, as the model did not output two other necessary triples, deviating from the prescribed ontology and guidelines. Furthermore, some false positive triples can be deemed true in the absence of background knowledge (*false positive accepted* triples), such as inferring the role of an employee as a manager from the relationship *hasManager*. The flexible paradigm will not consider them wrong during the calculation of the model's precision, thus increasing its value. Any of these triples may be further penalized for having content mistakes, as mentioned above.

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 to the above-presented equations are more optimistic, allowing one to better assess the usefulness of the LLM output for subsequent processing steps.

4. Results and discussions

This section presents and discusses the obtained results, aligned with the research questions guiding this study.

The experiments were carried out on Google Colab, using a virtual machine equipped with two Intel Xeon CPU 2.20GHz processors. Three models were tested, namely Mixtral-8x7b-Instruct-v0.1, GPT-3.5-Turbo-0125 and GPT-4o[6]. Mixtral is an open-source model, leveraging the Mixture of Experts [13] 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 one of OpenAI's best performing model. 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, regardless of the dataset. On TODSet, each run lasted approximately 120 minutes, with Mixtral consuming about 50% of the experimentation time. A total of 6750 prompts were sent in all runs to the three models. On FootballSet, an interation also took around 120 minutes, comprising a total of 3375 prompts. Each set of predictions can be loaded, tested and visualized from the paper's repository, available at https://github.com/IonutIga/LLMs-for-KGC.

[6]Experiments were conducted in October 2024, when GPT-4o was OpenAI's most advanced model. As of May 2025, GPT-4.1 was also evaluated; however, it showed only minor improvements over the previously reported GPT-4o metrics, so we chose to retain the original experiments.

During the evaluation, an extra post-processing step was needed for the GPT models. Due to their ability to generate JSON formatted output, the response was surrounded 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, the tag is still added. Thus, to ensure a fair analysis solely based on the output text, the tag is removed in a post-processing step.

Regarding TODSet, tables 5 to 8 display the results per prompt level and model, considering both strict and flexible metrics measurement paradigms. The first two tables focus on the Templates Easy dataset, while the latter ones on the Templates Hard dataset. Tables 5 and 7 display the results for the hand-written system prompts, while in tables 6 and 8, each model had to rephrase the system prompts beforehand. Tables 9 and 10 present class-wise performance across both dataset splits, based solely on metrics from hand-written prompts, which outperformed model-rephrased alternatives, as shown in former tables. Figure 6 displays the recall and TF1, under the flexible paradigm for each model per phrase types as described in Table 1, on TODSet dataset, using hand-written prompts. Table 13 highlights an in-depth analysis of the "*MS 2*" category from Figure 6, given three phrase sub-types. Table 14 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.

Regarding the FootballSet, table 11 shows the metrics per prompt level and model, considering both strict and flexible metrics measurement paradigms, using hand-written prompts. Table 12 presents the class-wise performance.

The best result for any model per level or class is highlighted in **bold**, while the best overall result among the models is in *italics*. The best overall level/class and model are underlined.

4.1. Disscusion about the influence of prompt engineering

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 improve performance. Level 3 prompts, when rigorously evaluated, exhibit a decrease in recall and TF1 scores compared to levels 1 and 2. When evaluated using more flexible metrics, the discrepancy diminishes. This can be attributed to the inclusion of explanatory text in responses, as models tend to replicate the input text which is more elaborate, thus not following the provided guidelines under the strict evaluation. Interestingly enough, on Templates Hard, Mixtral-8x7b yields its best scores at the level 1 prompt, when strictly measured.

ICL and COT Prompting Techniques Lead to the Best Results. Most of each models best results happened when prompted at levels 4.1 and 4.2, no matter the dataset or prompt 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 the literature [19] suggests. However, since Mixtral-8x7b sometimes provided explanations for its output, erroneous reasoning steps are still noticeable, especially in cases where the input text contains a class type that is not present in the ontology. Thus, despite the fact that GPT models exhibit this behavior less frequently, LLMs still have significant room for improvement in terms of reasoning capabilities.

Asking Models to Rephrase the System Prompt Might Generally Be a Good Idea for Mixtral-8x7b. Some experiments in the literature [17] ask 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. Compared to hand-written prompts, on the TODSet, Mixtral-8x7b benefits the most under rigorous evaluation, with an average increase of 7% for each recall and TF1 score. GPT-3.5-Turbo appears to conserve its behavior, signaling a decrease of only 2%. Surprisingly, GPT-4o exhibits a significant decrease in performance when it paraphrased the input prompts. On average, it reduced its performance by 33% for both metrics, with third-level prompts being the worst affected. Nonetheless, it is promising to see the open-source model enhancing its output by closely adhering to the provided system prompt.

4.2. Discussion about the influence of texts and ontologies structure

Implicit Reasoning Poses Challenges for LLMs. Template Hard dataset contains text cases that require the LLM to discover implicit connections between the mentioned entities. As concluded by the results presented in the

Table 5
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 | 0.56	0.47 | 0.57	0.73 | 0.75	0.48 | 0.63
2	Recall	0.19 | 0.49	0.45 | 0.51	0.63 | 0.85	0.42 | 0.62
	TF1	0.18 | 0.49	0.52 | 0.60	0.64 | 0.87	0.45 | 0.65
3	Recall	0.19 | 0.44	0.37 | 0.44	0.71 | 0.86	0.42 | 0.58
	TF1	0.20 | 0.50	0.48 | 0.58	0.72 | 0.88	0.47 | 0.65
4.1	Recall	**0.25** | 0.63	**0.88** | **0.88**	***0.89*** | ***0.91***	0.67 | 0.81
	TF1	**0.25** | 0.62	**0.88** | **0.88**	***0.89*** | ***0.91***	0.67 | 0.80
4.2	Recall	0.19 | **0.69**	0.85 | 0.87	***0.89*** | ***0.91***	0.64 | 0.82
	TF1	0.19 | **0.69**	0.85 | 0.87	***0.89*** | ***0.91***	0.64 | 0.80
Total Recall		0.21 | 0.54	0.59 | 0.63	0.75 | 0.83	0.52 | 0.67
Total TF1		0.22 | 0.57	0.65 | 0.72	0.78 | 0.86	0.55 | 0.71

Table 6
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 | 0.58	0.48 | 0.57	**0.73** | 0.75	0.54 | 0.63
2	Recall	0.15 | 0.37	0.45 | 0.47	0.36 | 0.85	0.32 | 0.56
	TF1	0.17 | 0.44	0.54 | 0.58	0.36 | 0.86	0.36 | 0.63
3	Recall	0.20 | 0.42	0.40 | 0.46	0.07 | 0.77	0.22 | 0.55
	TF1	0.22 | 0.48	0.50 | 0.59	0.07 | 0.64	0.26 | 0.57
4.1	Recall	0.19 | 0.58	***0.85*** | ***0.89***	0.66 | **0.90**	0.57 | 0.79
	TF1	0.19 | 0.57	***0.85*** | ***0.89***	0.66 | **0.86**	0.59 | 0.77
4.2	Recall	**0.42** | **0.74**	0.84 | 0.89	0.31 | 0.87	0.52 | 0.83
	TF1	**0.42** | **0.72**	0.84 | 0.89	0.31 | 0.82	0.52 | 0.81
Total Recall		0.27 | 0.52	0.59 | 0.63	0.40 | 0.81	0.42 | 0.65
Total TF1		0.28 | 0.56	0.65 | 0.71	0.43 | 0.80	0.45 | 0.69

tables from 5 to 8, under flexible metrics, Mixtral 8x7b achieves at best a recall and TF1 scores of 56% on the more difficult dataset, which is 17% 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%. The same behavior is observed with GPT-4o, as it falls from 91% recall and TF1 score to around 76% and 74%, respectively. Figure 6 displays the differences in a compact form, on the TODSet dataset, based on each phrase's type, under the flexible paradigm. 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 example, all models reduce their recall, on average, with 12%, and their TF1 score with 13%.

LLMs Only Appear to Adhere to the Ontology. While the results in Tables 5-10 and Figure 6 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 (MS2) category from Figure 6 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 may seem acceptable at first glance, a deeper look at phrase types reveals flaws in LLMs behavior. All models performed reasonably well when encountering UVW text types, reaching 86% recall

Table 7
Results on Template Hard (TH) dataset, using hand-written prompts

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

Table 8
Results on Template Hard (TH) dataset, using model rephrased prompts

Model		Mixtral	GPT-3.5-Turbo	GPT-4o	Total
Level	Metric	strict ǀ flexible	strict ǀ flexible	strict ǀ flexible	strict ǀ flexible
1	Recall	**0.33** ǀ 0.43	0.37 ǀ 0.42	**0.51** ǀ 0.54	0.40 ǀ 0.46
	TF1	**0.37** ǀ 0.49	0.45 ǀ 0.52	**0.60** ǀ 0.64	0.47 ǀ 0.55
2	Recall	0.14 ǀ 0.35	0.43 ǀ 0.47	0.28 ǀ 0.72	0.28 ǀ 0.51
	TF1	0.15 ǀ 0.39	0.50 ǀ 0.56	0.28 ǀ 0.73	0.31 ǀ 0.56
3	Recall	0.12 ǀ 0.39	0.41 ǀ 0.45	0.09 ǀ 0.66	0.20 ǀ 0.50
	TF1	0.13 ǀ 0.43	0.50 ǀ 0.55	0.09 ǀ 0.56	0.23 ǀ 0.51
4.1	Recall	0.07 ǀ 0.47	***0.71*** ǀ ***0.75***	0.55 ǀ **0.76**	0.44 ǀ 0.66
	TF1	0.07 ǀ 0.45	***0.71*** ǀ ***0.75***	0.55 ǀ **0.73**	0.44 ǀ 0.64
4.2	Recall	0.31 ǀ **0.56**	0.70 ǀ 0.74	0.18 ǀ 0.73	0.40 ǀ 0.68
	TF1	0.31 ǀ **0.56**	0.70 ǀ 0.74	0.18 ǀ 0.69	0.40 ǀ 0.66
Total Recall		0.19 ǀ 0.44	0.52 ǀ 0.57	0.32 ǀ 0.68	0.34 ǀ 0.56
Total TF1		0.20 ǀ 0.46	0.57 ǀ 0.63	0.34 ǀ 0.68	0.37 ǀ 0.59

using GPT-4o, as can be noticed in Table 13. 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 OOD class types, such as the example in the 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 the target output when the cases are general enough.

Elaborate Ontologies Are More Difficult to Grasp. The TODSet dataset is based on a simple ontology of only three concepts and six relationships. However, the FootballSet dataset constructs its texts on a more elaborate one, comprising 14 concepts and 24 relationships among them. This leads to significantly lower results for all tested models. Under the flexible paradigm on FootballSet, the overall recall was 40% and the TF1 score was 45%, with 26.5% lower than the metrics on the TE split of TODSet. When comparing the top scores on both sets, recall and TF1 reduced from 91% to 65%, highlighting the increased difficulty of understanding more elaborated ontologies.

Table 9
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 | 0.63	0.67 | **0.76**	0.83 | 0.92	0.57 | 0.77
E	Recall	**0.25** | **0.68**	**0.63** | 0.66	***0.85*** | ***0.89***	0.58 | 0.74
	TF1	**0.23** | **0.66**	**0.68** | 0.72	***0.88*** | ***0.92***	0.60 | 0.77
S	Recall	0.18 | 0.47	0.53 | 0.62	0.60 | 0.68	0.44 | 0.59
	TF1	0.17 | 0.43	0.55 | 0.65	0.64 | 0.72	0.45 | 0.60
N	Recall	0.09 | 0.10	0.43 | 0.43	0.38 | 0.43	0.30 | 0.32
	TF1	0.09 | 0.10	0.43 | 0.43	0.36 | 0.43	0.29 | 0.32
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 | 0.72

Table 10
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** | **0.55**	**0.71** | **0.76**	***0.81*** | ***0.89***	0.56 | 0.73
E	Recall	0.13 | 0.38	0.43 | 0.51	0.27 | 0.40	0.28 | 0.43
	TF1	0.13 | 0.42	0.49 | 0.60	0.30 | 0.43	0.31 | 0.48
S	Recall	0.07 | 0.29	0.56 | 0.60	0.40 | 0.48	0.34 | 0.46
	TF1	0.07 | 0.29	0.61 | 0.65	0.42 | 0.50	0.37 | 0.48
N	Recall	0.03 | 0.03	0.19 | 0.20	0.18 | 0.22	0.13 | 0.15
	TF1	0.03 | 0.03	0.19 | 0.20	0.19 | 0.22	0.13 | 0.15
Total Recall		0.11 | 0.39	0.54 | 0.58	0.62 | 0.70	0.42 | 0.56
Total TF1		0.12 | 0.42	0.59 | 0.65	0.64 | 0.72	0.45 | 0.59

Complex Class Types Do Not Imply More Difficult Reasoning. Analyzing both Tables 9 and 10, all models seem to perform better on the *Project* type, compared to the other three classes. It may be attributed to the inclusion of more difficult phrase types for this class, combined with a notable lower number of examples for the other three. Despite 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 75% for the *Project* class, while for *Employee*, the models only achieve 59% and 63%. 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 54%, respectively - 6% and 9% 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').

The same behavior is seen on the FootballSet ontology, as presented in Table 12. The ontology's most complex class is *SportsTeam*, part of 14 relationships, followed by *Athlete* and *Country* with 7, *Person* with 6, and *Place* and *League* with 2. *SportsTeam* has the best results compared to any other type, with a recall of 46% and TF1 score of 52%, under flexible paradigm. Moreover, it seems that all results are in accordance with the class complexity, with only one exception, as the *Athlete* concept registered lower metrics than *Person*, although it has it is part of

Table 11
Results on FootballSet (FS) 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	0.05 | 0.10	0.16 | 0.20	0.37 | 0.38	0.19 | 0.23
	TF1	0.06 | 0.14	0.23 | 0.28	0.46 | 0.47	0.25 | 0.30
2	Recall	0.26 | 0.39	0.26 | 0.28	0.67 | 0.67	0.40 | 0.45
	TF1	0.28 | 0.43	0.33 | 0.36	0.68 | 0.69	0.43 | 0.49
3	Recall	0.09 | 0.20	0.19 | 0.22	0.47 | 0.52	0.25 | 0.31
	TF1	0.12 | 0.27	0.27 | 0.31	0.53 | 0.59	0.31 | 0.39
4.1	Recall	**0.36 | 0.49**	***0.64 | 0.64***	**0.59 | 0.60**	0.53 | 0.58
	TF1	**0.38 | 0.54**	***0.65 | 0.66***	**0.60 | 0.61**	0.54 | 0.60
4.2	Recall	0.00 | 0.11	0.61 | 0.62	0.27 | 0.53	0.30 | 0.42
	TF1	0.00 | 0.17	0.63 | 0.65	0.27 | 0.53	0.31 | 0.45
Total Recall		0.15 | 0.26	0.37 | 0.39	0.48 | 0.54	0.33 | 0.40
Total TF1		0.17 | 0.31	0.43 | 0.45	0.52 | 0.58	0.37 | 0.45

Table 12
Results on FootballSet (FS) dataset, using hand-written prompts, on class types. Symbol "ST" stands for *SportsTeam*, "P" for *Person*, "A" for *Athlete*, "C" for *Country*, "L" for *League*, and "Pl" for *Place*

Model		Mixtral	GPT-3.5-Turbo	GPT-4o	Total
Class	Metric	strict | flexible	strict | flexible	strict | flexible	strict | flexible
ST	Recall	**0.19 | 0.31**	**0.43 | 0.46**	***0.54 | 0.61***	0.39 | 0.46
	TF1	**0.21 | 0.37**	**0.50 | 0.54**	***0.57 | 0.65***	0.43 | 0.52
P	Recall	**0.19 | 0.31**	0.37 | 0.38	0.44 | 0.53	0.33 | 0.41
	TF1	**0.21 | 0.37**	0.45 | 0.47	0.48 | 0.58	0.31 | 0.50
A	Recall	0.05 | 0.12	0.23 | 0.25	0.37 | 0.40	0.22 | 0.27
	TF1	0.05 | 0.15	0.27 | 0.31	0.41 | 0.45	0.24 | 0.31
C	Recall	0.01 | 0.15	0.32 | 0.33	0.51 | 0.56	0.28 | 0.35
	TF1	0.01 | 0.20	0.37 | 0.38	0.53 | 0.59	0.30 | 0.39
L	Recall	0.07 | 0.14	0.23 | 0.25	0.25 | 0.25	0.18 | 0.21
	TF1	0.09 | 0.20	0.29 | 0.32	0.33 | 0.33	0.24 | 0.28
Pl	Recall	0.16 | 0.21	0.17 | 0.19	0.27 | 0.30	0.20 | 0.23
	TF1	0.17 | 0.22	0.17 | 0.20	0.29 | 0.32	0.21 | 0.25
Total Recall		0.15 | 0.26	0.37 | 0.39	0.48 | 0.54	0.33 | 0.40
Total TF1		0.17 | 0.31	0.43 | 0.45	0.52 | 0.58	0.37 | 0.45

Table 13
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 unknown vocabulary words ("UVW").

Model	OOD	w/o Insert	UVW
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**

more relationships. A deeper analysis reveals that the *Athlete* class had more difficult texts types, which led to lower scores.

The Underlying Semantics of Words Pose a Challenge for LLMs. The *Project* and *Employee* classes are linked

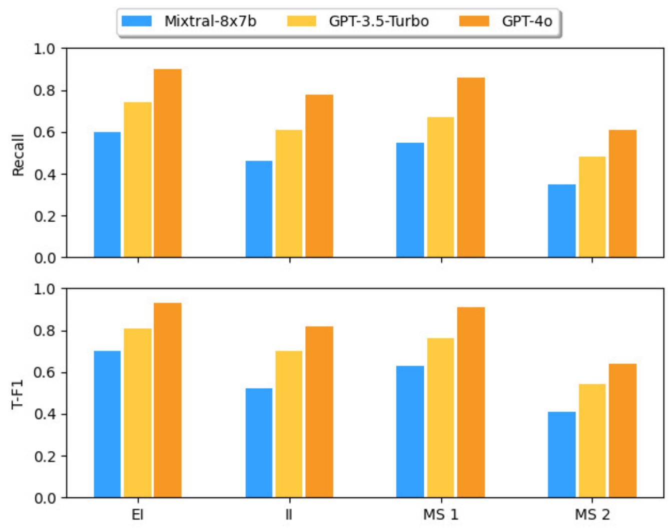

Fig. 6. Recall and TF1 for each model per phrase type under the flexible paradigm on TODSet dataset using hand-written prompts. Symbol "EI" stands for *Explicit Information*, "II" for *Implicit Information*, "MS1" and "MS2" for *"Misleading Information types 1 or 2"*

Table 14

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

through the *hasManager* relationship, and most test phrases reference an employee by 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, although not drastically. As shown in Table 14, GPT-4o maintains 83% recall and an 88% TF1 score, close to its overall performance (85% recall, 90% for TF1 in Table 10, *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.

4.3. *Discussion about LLMs performance*

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 the full IRI of an entity or explanatory text, as evidenced by the 0 scores at the 4.2 level in Tables 7 and 11, scores that otherwise notably increase when formatting mistakes are allowed.

Top-tier LLMs Effectively Address Grammatical Errors. Figure 6 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.

GPT-4o is More Consistent and Performant, While GPT-3.5-Turbo Achieves the Best Results. Despite showing fluctuations in results when it rephrased the prompts, GPT-4o was the best overall model. Based on Table 5, on the TE dataset, under strict measurements, it had 75% recall and 78% TF1 score, almost four times more than Mixtral-8x7b and with 13.5% more than GPT-3.5-Turbo. The same behaviour can be seen on the FootballSet dataset, where GPT-4o had 48% recall and 52% TF1 score, with 34% more than Mixtral-8x7b and 10% more than GPT-3.5-Turbo. It can be interpret 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 both TH and FootballSet datasets, as it can be observed in Table 7 and Table 11. Depending on the user's objectives, while considering the model's costs, the choice of the final model could vary.

In summary, KGC remains a challenging task for LLMs under Zero-Shot prompting. As models become better, their performance tend to increase, while shifting the focus on optimizing the costs. Moreover, when checking their intermediate reasoning steps, the LLMs 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. [4], 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 showcased the ability of three leading LLMs, namely Mixtral-8x7b-Instruct-v0.1, GPT-3.5-Turbo-0125 and GPT-4o, to tackle the Knowledge Graph Construction task. The two proprietary models produced great overall results, as GPT-4o was more consistent, while GPT-3.5-Turbo achieved the best metrics on two out of the three datasets. Moreover, both of them effectively addressed input texts with grammatical errors, underlining their inherent capabilities of processing natural language texts. On the other hand, Mixtral-8x7b rarely followed the required output format, however, it benefited the most from rephrasing the system prompt. Another important aspect is that none of them was able to correctly handle out-of-distribution class types, where models should have not output any triples, thus all models only appeared to adhere to any ontology.

The variety of prompt engineering techniques used throughout the study highlighted the significance of tailoring input text to suit both the task and the specific model. Simply increasing the number of instructions did not consistently enhance performance; instead, incorporating examples through techniques like in-context learning or chain-of-thought proved more effective. However, our experiments had a fixed example for every prompt, which may not always be sufficiently relevant for the model, limiting the benefits of such additions. Moreover, asking models to rephrase the system prompt showed potential for improving task comprehension in some cases, although it occasionally led to performance declines in other models. This approach should therefore be applied with care.

Key challenges arose from the structure of the input texts and ontologies. As anticipated, texts that demanded additional reasoning steps proved difficult for all models tested. Interestingly, the primary difficulty with the ontology stemmed not from the structure of its concepts but rather from the overall size of the ontology, which posed significant challenges for the models.

The flexible metrics evaluation paradigm proposed in this study enables a favorable assessment of models that produce errors amenable to correction through subsequent post-processing, thereby placing greater emphasis on the semantic quality of the generated output. Various penalties were applied depending on the type of error, related to the format or content. This perspective supports the adoption of LLMs that, while potentially less precise than others, offer the advantage of reduced resource requirements. However, penalty values were customized to our task-specific needs, restricting their broader applicability.

Finally, a two-fold customized dataset was proposed, namely the TODSet, including texts derived from the training schedule of a TOD system [9]. It includes phrases grouped into various categories that reflect different levels of difficulty and linguistic characteristics. Each text is annotated with a set of golden labels representing the target extractable triples. Additionally, two supplementary sets of triples, alternative and false positive accepted ones, were included to account for valid deviations that convey similar semantic content. This design facilitates the flexible metrics measurement paradigm. However, human experts are required to formulate such triples, limiting the size of the dataset, as it is a costly process.

Future work will prioritize the integration of additional LLMs for testing, facilitated by our interface's seamless incorporation of new endpoints. Moreover, models will be tested with longer input phrases and more complex ontologies, closer to real-world scenarios. The flexible metrics measurement paradigm needs to have a smooth way of integrating any type of penalties, while the creation of a dataset will incorporate methods that automatically create any type of triple.

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] Z. Chen, L. Chen, B. Chen, L. Qin, Y. Liu, S. Zhu, J.-G. Lou and K. Yu, UniDU: Towards A Unified Generative Dialogue Understanding Framework, in: *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, O. Lemon, D. Hakkani-Tur, J.J. Li, A. Ashrafzadeh, D.H. Garcia, M. Alikhani, D. Vandyke and O. Dušek, eds, Association for Computational Linguistics, Edinburgh, UK, 2022, pp. 442–455. doi:10.18653/v1/2022.sigdial-1.43.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] F. Polat, I. Tiddi and P. Groth, Testing Prompt Engineering Methods for Knowledge Extraction from Text, *Semantic Web* **accepted** (2024). https://www.semantic-web-journal.net/content/testing-prompt-engineering-methods-knowledge-extraction-text-0.

[19] 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.

[20] 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.

[21] 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.

[22] C. Wu, S.C.H. Hoi, R. Socher and C. Xiong, TOD-BERT: Pre-trained Natural Language Understanding for Task-Oriented Dialogue, in: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, B. Webber, T. Cohn, Y. He and Y. Liu, eds, Association for Computational Linguistics, 2020, pp. 917–929. doi:10.18653/V1/2020.EMNLP-MAIN.66.

[23] 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.

[24] 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.

[25] 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.

[26] Z. Zhang, R. Takanobu, Q. Zhu, M. Huang and X. Zhu, Recent advances and challenges in task-oriented dialog systems, *Science China Technological Sciences* **63**(10) (2020), 2011–2027. doi:10.1007/s11431-020-1692-3.

[27] 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.

[28] 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.